COMPUTATION OF INVISCID COMPRESSIBLE FLOWS ABOUT ARBITRARY GEOMETRIES AND MOVING BOUNDARIES

Volume Two

by

Sami Alan Bayyuk

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Aerospace Engineering)
in The University of Michigan
2008

Doctoral Committee:

Professor Kenneth G. Powell, Co-Chairman
Professor Bram van Leer, Co-Chairman
Professor Philip L. Roe
Professor Grétar Tryggvason
Professor Marsha Berger, Courant Institute,
New York University
Dr. August Verhoff, Boeing Technical Fellow,
The Boeing Corporation

CHAPTER VII

Cell Merging

Cell merging plays an essential, enabling role in the method developed and studied in this work. This chapter describes that role and how it is called upon, presents the algorithm used to merge cells, presents the theorems on which this algorithm is based, and describes the disadvantages and limitations of cell merging. Also addressed here are the computational requirements for cell merging. Possible improvements to the current merging algorithm are proposed, along with ideas on extending cell merging to more general situations, particularly to three-dimensional moving-boundary computations. Other than in the latter regard, the presentation is strictly in a two-dimensional context.

7.1 Merging and Composite Cells

Cell merging is the combining of individual cells from the subregion set [1] to form **composite cells** that can be *treated as* individual cells in the flow-solver. Whether the cells are also combined into one geometric unit is a secondary, implementational detail.

Any composite cell must at least satisfy the definition of an individual subregion; in particular, it must be simply-connected, implying that no member of a multi-

[1] The terms "subregion" and "subregion set" are defined in Chapter IV

member composite cell may be simultaneously disjoint from all the other members. Convexity is sometimes not necessary, but almost always desirable. The additional special requirements that composite cells must meet for cell merging to fulfill its role are purely geometric; they are discussed in section 7.3.

7.2 Definitions I: Topologic and Geometric Invariance

In this section, all **points**, **line segments**, and **polygons** refer to geometric elements of the Cartesian Grid and are therefore stationary.

A point or vertex is **topologically invariant** for a motion step (of the internal boundaries) iff its topologic status (that is, its state among "in", "on", or "out") at the start of the motion step is identical to its topologic status at the end of the motion step with respect to *every* closed internal boundary in the computational region. A line segment or closed polygon is **topologically invariant** iff the set of topologic states taken by the set of points in the segment or polygon at the start of a motion step is identical to that at the end of the motion step with respect to *all* the internal boundaries in the computational region.

A curved or straight line segment is **geometrically invariant** iff its intersection pattern with *all* the internal boundaries of the computational region at the start of a motion step is "similar" to its intersection pattern with *all* the internal boundaries at the end of the motion step. Intersection patterns are similar iff any intersection points in the segment at most only translate along the segment and without any change of order, where order refers to both intra-boundary and inter-boundary order along a segment. A closed polygon is **geometrically invariant** iff each of its edges is geometrically invariant.

In contrast to the case with topologic invariance, geometric invariance of the

edges of a polygon is necessary for geometric invariance of the polygon.

For any polygon or line segment, geometric invariance implies topologic invariance. If boundaries are stationary, every line segment (hence every polygon) is geometrically (hence also topologically) invariant.

7.3 The Role of Cell Merging

The grid generation scheme chosen in this work is characterized by the use of a stationary and non-boundary-conformal grid, even for unsteady flows and moving boundaries. Such a choice requires confronting the following two sources of numerical and algorithmic difficulties, and the success of the overall method hinges on how well they are dealt with:

1. change in the topologic status or intersection pattern of cells between the start and end of a motion step; and

2. arbitrary intersection of cells on internal boundaries.

Each of the above sources of difficulty is encountered with several cells in figure 7.1. The figure shows the initial and final positions across one motion step of an internal boundary that undergoes deformation and translation. The figure also shows the **motion envelope**, defined as the union of all cells that are intersected by the interface at least at one of the (here, two) discrete positions in the motion step. Only the cells in the motion envelope are drawn, and for simplification, they all have the same refinement level in the quadtree.

Each of the two sources of difficulty listed above presents three challenges to the solution algorithm in updating the state vector of an affected cell:

- maintaining stability;

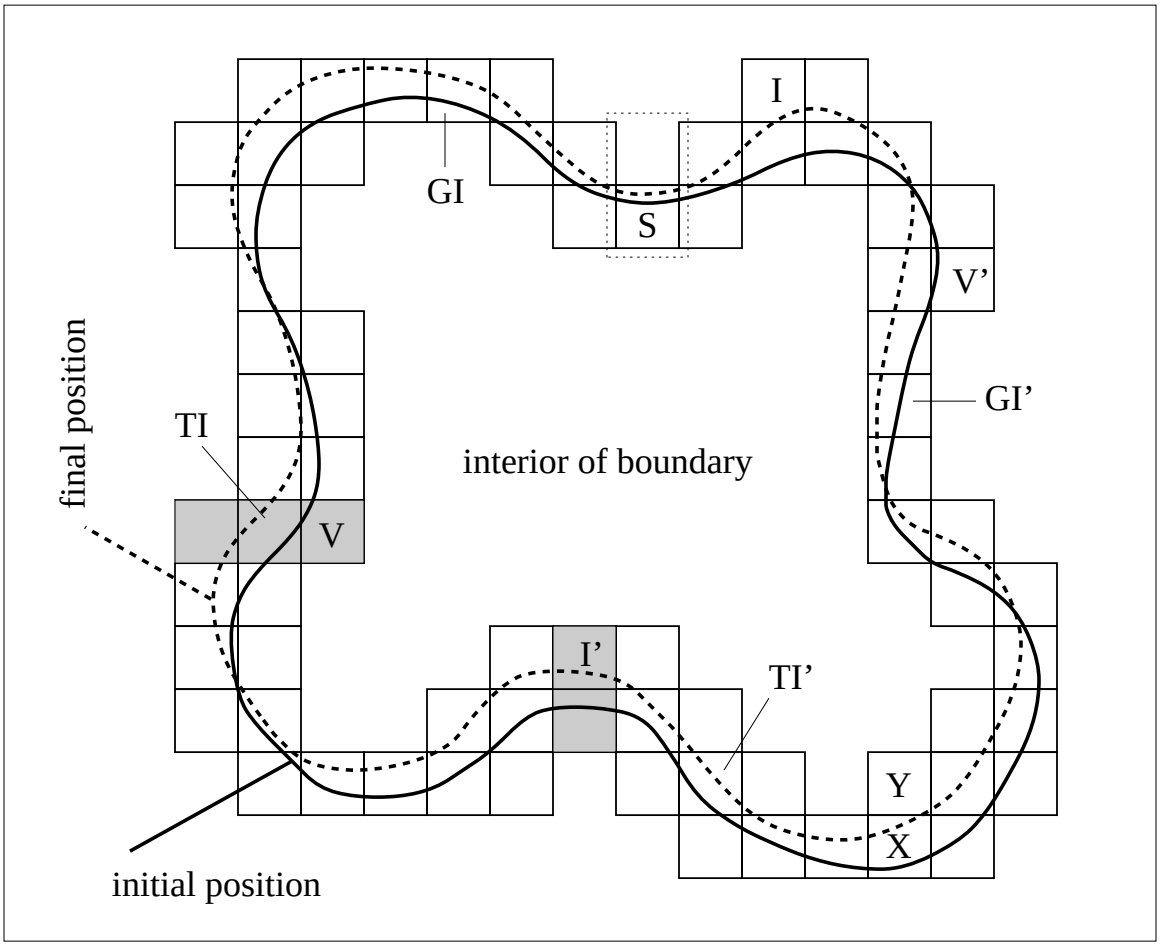

Figure 7.1: The problems solved by cell merging: cells that undergo change in intersection pattern or topologic status, and cells that have small area.

- maintaining conservation; and
- accurate computing of the fluxes between the cell and all its neighbors, leading to an accurate update of the affected cell and all its neighbors.

This work uses cell merging to simultaneously eliminate both sources of problems. While the use of cell merging to overcome the second source of difficulties is not new [79, 74, 176], what is new in this work is its use in a general way to *also* overcome the first source of problems.

7.3.1 Changes in Intersection Pattern and Their Treatment

Changes in topologic status or intersection pattern arise only if there is boundary motion. If the maximum displacement of any point in the boundary is kept to less than the smallest *local* cell dimension, the set of possible changes is reduced; in particular, change in topologic status without change in intersection pattern cannot occur, as discussed in more detail below. Removing the restriction on boundary displacement does not introduce any additional computational challenges; in practice, the only side-effect is an increase in the size of composite cells required. For simplification, and without loss of generality, the rest of this section assumes that the above restriction, justified in Chapter VI, is in effect.

Table 7.1 classifies the admissible types of change in intersection pattern into eight categories, based on the computational problems encountered, citing an example of each from figure 7.1. As the figure shows, there are pairs of cells for which the types of change experienced by the interior and exterior states are reversed. This duality is indicated by symbols that differ only by a "prime". Because of it, the eight categories can be reduced to four fundamental ones.

Categories 1, $1'$, 2, and $2'$ involve topologic transformations. Note that, under the restriction on boundary displacement mentioned above, the only possible changes in topologic status are of the type "in" to "on" and vice-versa (boundary invasion and boundary vacation, respectively), and "out" to "on" and vice-versa (boundary invasion and boundary vacation, respectively); direct change in either direction between the "on" and "out" states in the same motion step is precluded. The other types of change in intersection pattern (categories 3, $3'$, 4, and $4'$) do not involve a change in topologic status. In particular, categories 4 and $4'$ maintain geometric invariance. Changes involving topologic transformations present the most challenging problems

Type	Geometric Description	G. I. [a]	Topologic Transfor--mation	T. I. [b]	Computational Challenge		Example cell [c]
					interior state	exterior state	
1	boundary invasion	no	out $\longrightarrow$ on	no	t.i. [d]	p.e. [e]	I
$1'$	boundary invasion	no	in $\longrightarrow$ on	no	p.e.	t.i.	I'
2	boundary vacation	no	on $\longrightarrow$ in	no	p.i. [f]	t.e. [g]	V
$2'$	boundary vacation	no	on $\longrightarrow$ out	no	t.e.	p.i.	V'
3	boundary re-intersection	no	on $\longrightarrow$ on	yes	p.i.	p.e.	TI
$3'$	boundary re-intersection	no	on $\longrightarrow$ on	yes	p.e.	p.i.	TI'
4	boundary re-intersection	yes	on $\longrightarrow$ on	yes	p.i.	p.e.	GI
$4'$	boundary re-intersection	yes	on $\longrightarrow$ on	yes	p.e.	p.i.	GI'

[a] Maintenance of geometric invariance.
[b] Maintenance of topologic invariance.
[c] In reference to figure 7.1.
[d] Total introduction.
[e] Partial elimination.
[f] Partial introduction.
[g] Total elimination.

Table 7.1: Classification of the types of change in intersection pattern. "Geometric Description" and "Topologic Change" are with respect to the affected cell.

from the computational point of view, since only these involve total elimination or introduction of states.

Before discussing the role of merging, the computational difficulties presented by the transformations shown in table 7.1 are discussed.

7.3.1.1 Total Elimination of States

Only categories 2 and $2'$ in table 7.1 are subject to total elimination, that is, the complete removal of a state across a motion step.

Consider cell V in figure 7.1. Its vacation by the boundary physically causes the exterior "part" of the cell, and the associated state, to "vanish". The only outcome of an update for the exterior state that does not violate the conservation principle is the vacuum-state. However, basing the fluid-dynamic update on the edge lengths at the initial position, and on the fluxes given by the initial state vectors could result in a state that is either (i) non-physical, or (ii) physically admissible but conservation-violating (in the sense that the exterior part of the cell will not necessarily be evacuated of the exterior fluid). Also, this method would not generally "distribute" the conserved quantities to the neighboring cells accurately, that is, in accordance with the relative motion trajectories of the intersection points along the edges of the cell, nor would it allow accurate application of boundary conditions. The conservation problem can be viewed as a consequence of the accuracy problem. However, it is infeasible to enforce the conservation property here by seeking improvements in the accuracy.

The conservation and accuracy problems described in the above paragraph can both be alleviated but neither of them can be eliminated by interpolating to find the "stage" of the motion step at which the boundary leaves the cell and using that in an explicit scheme to partition the global time-step into sub-steps, each corresponding to one of the different intersection patterns of the affected cell that arise during the motion step. Also, since the stages of the motion step at which change of status occurs may differ arbitrarily among cells, this procedure would at least double the computing effort for geometric calculations.

7.3.1.2 Total Introduction of States

Only categories 1 and $1'$ in table 7.1 are subject to total introduction, that is, presence of a state at the end of a motion step that was absent at the start of the motion step.

Consider cell I' in figure 7.1. Its invasion by the boundary physically causes the introduction of an exterior "part" and an associated state. Violations of the conservation property are easy to prevent in this case; the main difficulty is how to accurately evaluate the state vector.

Extrapolation of the state from neighbors on the exterior side of the boundary generally leads to violations of conservation. Using the initial, "vacuum", state vector over the entire motion step leads to the vacuum state in the exterior part of the cell at the end of the motion step. Although conservation is retained with this approach, the accuracy is unacceptably low for all types of interface, and invalidatingly so for massless interfaces. Splitting the update by the use of sub-steps within the motion step to account for "switching" of intersection points between different sets of edges of the cell will improve the accuracy of the boundary procedure and the update of all affected cells but generally requires computation of several intermediate gasdynamic states and, as explained above, requires additional geometric computations; therefore, it is not an attractive option.

7.3.1.3 Partial Elimination or Introduction of States

Partial elimination and partial introduction refer to a reduction or an increase in the corresponding partial area of a cell, respectively (rather than a reduction or increase in the corresponding conserved quantities). Only categories 3, $3'$, 4, and $4'$ in table 7.1 are subject to only partial elimination or introduction.

Consider the exterior state of cell TI in figure 7.1. The cell does not change its topologic status, so conservation can readily be maintained. However, accurate computation of fluxes and accurate application of boundary conditions presents similar, albeit more manageable, difficulties as those described for fully eliminated and fully introduced states because of the migration of intersection points across edges. Note that this situation cannot arise unless there are neighboring cells that are either invaded or vacated. All cells that are topologically but not geometrically invariant, that is, all those falling in categories 3 and $3'$ present the same difficulties as cell TI.

Cell GI in figure 7.1 also experiences partial elimination and partial introduction. However, there is an important difference between it and cell TI, as discussed in the next subsection.

7.3.1.4 The Special Case of Geometric Invariance

Consider the exterior state of cell GI in figure 7.1. Conservation can readily be maintained, just as for cell TI. However, there is no migration of intersection points across edges and each edge of the cell is geometrically invariant. This allows second-order accuracy in the flux computation to be obtained simply and without considering any intermediate stage in the motion step. In particular, on each edge, the flux used in the second-order version of the update in this work implicitly evaluates to the length-weighted average of the initial and final fluxes, that is, it can be expressed by the approximation

$$\widetilde{\vec{\mathbf{F}} \cdot \vec{\mathbf{ds}}} = [(\vec{\mathbf{F}} \cdot \vec{\mathbf{ds}})_{inital} + (\vec{\mathbf{F}} \cdot \vec{\mathbf{ds}})_{final}]/2, \tag{7.1}$$

where $\tilde{(.)}$ is the time-average of $(.)$ over a motion step.

Geometric invariance of the edges of cell GI similarly enables boundary conditions

to be simply applied with accuracy that is nominally up to second-order.

The discussion and conclusions for cell GI clearly extend to all cells falling in categories 4 and 4′, since the determining characteristic is geometric invariance. Under conditions of interface motion on a stationary grid, geometric invariance is the most favorable change in intersection pattern possible.

The use of geometrically invariant cells across which boundaries move in a Finite-Volume-based solver does not present any significant difficulties. This is because the Finite-Volume method is readily applicable to subregions of arbitrary shape, while for moving boundaries, the method readily admits a straightforward and accurate formulation for subregions with variable area (or volume) and translating boundary conditions.

Note that cells that remain unintersected throughout a motion step, including those well away from all motion envelopes are all, vacuously, geometrically invariant.

7.3.1.5 The Unifying Remedy: Cell Merging

The previous subsection demonstrates that geometric invariance is superior to topologic invariance which in turn is superior to topologic non-invariance as a requirement for computational cells with moving boundaries. This section shows how cells can be merged together to eliminate the need to deal with topologic or geometric non-invariance.

Consider the portion of a motion envelope shown in figure 7.2. Segments of the boundary are shown at the start and the end of the motion step. The interior/exterior orientation is irrelevant in what follows and is therefore not shown. The complete intersection pattern with the "dotted" cells is also not shown in order to leave, for subsequent discussion, undetermined possibilities.

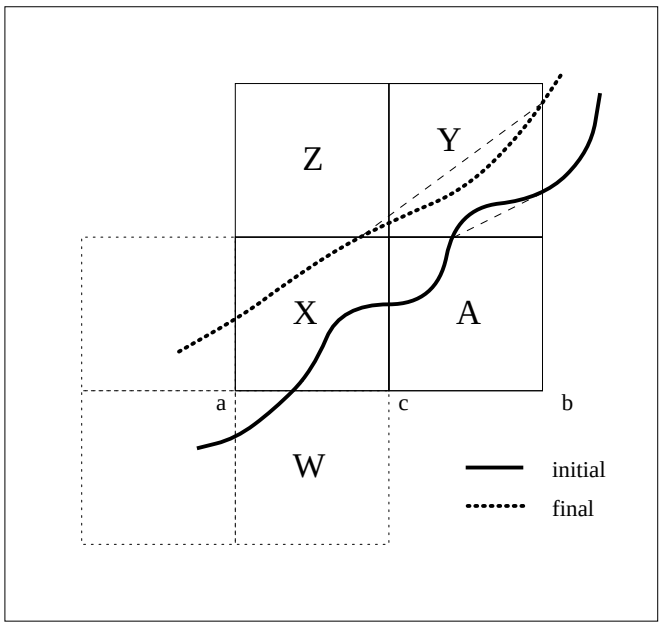

Figure 7.2: Topologic and geometric invariance.

Cells A, Z, and W undergo change in topologic status, while cells X and Y do not. None of the mentioned five cells is geometrically invariant.

Suppose cell Y is merged with cell Z. The resulting composite cell Y $\cup$ Z is geometrically invariant; if it were used as an individual cell in the flow-solver, this merging would eliminate all the problems arising from the geometric non-invariance of cell Y and the topologic non-invariance of cell Z. This conclusion is valid for any sensible representation of boundaries within composite cells, provided the actual intersection points with the cell edges are preserved. The figure shows a "linear-truncation representation" of boundaries within the cell Y $\cup$ Z.

Now suppose cell A is merged with cell X. The cell A $\cup$ X is topologically but not geometrically invariant (because its status is "on" at both positions of the boundary, but the segment *acb* is not geometrically invariant). While this merging eliminates the problems associated with the change of topologic status of cell A, the resulting

composite cell still presents the problems of partial elimination and partial introduction of states.

Consider merging cells A, X, and the left neighbor of X. Depending on whether the boundary at the final position intersects the bottom edge of the left neighbor of cell X, merging that neighbor with cells A and X may also not result in a geometrically invariant composite cell.

Using only the information shown in the figure, the two composite cells Y $\cup$ A and Z $\cup$ X $\cup$ W provide an appropriate combination that satisfies geometric invariance and includes all five labeled cells.

Another two examples of suitable merging selections producing geometrically invariant composite cells are indicated by the shaded rectangles in figure 7.1. Note how each composite cell must include at least two problematic cells if there are changes in topologic status. For the composite cell containing cell V, the exterior part contracts across the motion step, while for the composite cell containing cell I', the exterior part expands across the motion step.

An important question is whether the above examples of successful merging apply to any geometry and discretization. The answer is given in section 7.10 where it is shown that provided certain conditions hold, all problematic cells can be merged into geometrically invariant composite cells. Given that this is so, the purpose of cell merging with regard to handling changes in intersection pattern can now be precisely defined as the combining of cells in categories 1, 1′, 2, 2′, 3 and 3′ with appropriate neighbors to form composite cells that fall only into categories 4 and 4′. If, instead of geometric invariance, the inferior criterion of topologic invariance is used to drive the cell merging process, composite cells falling in categories 3 and 3′ also become admissible. Table 7.2 lists in detail how the categories are collapsed under each of

the two criteria.

Even if only topologic invariance were ultimately sought, basing a merging algorithm on the stronger condition of geometric invariance is computationally advantageous because it can be shown to require examination of smaller "stencils" of cells to form each composite cell, and to require less frequent re-merging. Both of these factors lead to greater computational efficiency. For example, a topologic invariance criterion would allow the merging of cells A and X in figure 7.2. Depending on how the boundary at the final position intersects the extension of the segment *bca*, this may preclude the possibility of finding a convex, topologically invariant composite cell that covers the cells indicated by dotted lines. If that were to happen, re-merging the cells in that neighborhood becomes necessary. On the other hand, a geometric invariance criterion would allow only the composite cells A $\cup$ Y and W $\cup$ X $\cup$ Z to be formed. Several other examples where topologic invariance would require examination of larger stencils can be seen in figure 7.1. One of them occurs around cell V: if that cell were merged with only the cell immediately to its left, no topologically invariant combination could be found for the rest of the cells in the lower left part of the motion envelope.

7.3.2 Arbitrary Cell-Intersections and Their Treatment

The second problem introduced by the grid generation approach adopted in this work is often called the "small-cell problem". This problem has already been discussed in Chapter VI, and described in detail in Appendix A.1. It is a consequence of only the non-conformality and may be encountered with or without boundary motion. As explained in Appendix A.1, for a Finite-Volume solver the concerns raised by this problem almost entirely revolve around any stability constraints imposed as

Category of Problematic Individual Cell	Category of Resulting Composite Cell[a]	
	Toplogic Invariance Merging Criterion	Geometric Invariance Merging Criterion
1	3 or 4	4
$1'$	$3'$ or $4'$	$4'$
2	3 or 4	4
$2'$	$3'$ or $4'$	$4'$
3	3 or 4	4
$3'$	$3'$ or $4'$	$4'$
4	4	4
$4'$	$4'$	$4'$

[a] This refers to the composite cell formed to include at least the problematic cell as a member.

Table 7.2: Unification of intersection-pattern-change categories by cell merging under the criteria of (a) topologic invariance, and (b) geometric invariance.

a consequence of the cell area being too small.

Consider example-cell S in figure 7.1. It is geometrically invariant, but its initial and final exterior areas are both small. Cell merging, the approach adopted in this work, would combine this cell with the one immediately above it (which is not drawn since it does not lie in the motion envelope) to form the composite cell that lies within the dashed rectangle. The indicated composite cell has acceptable initial and final exterior areas, eliminating entirely any unnecessary stability constraints for a Finite-Volume solver. The ratio of the above two areas is also important, and in this case is acceptable.

Any computational cell, including a composite cell, may possibly have inappropriate areas. In addition to the geometric invariance condition, composite cells must therefore also be required to satisfy constraints on their areas to avoid the small-cell problem. For example, the composite cell $X \cup Y$ in figure 7.1 satisfies the geometric invariance condition but has too small an exterior final area and too large a ratio of

initial to final exterior areas; the problem is resolved by merging with a third cell below X. The initial and final areas are acceptable for the two composite cells shown covering the cells V and I' in figure 7.1.

The area constraints are imposed on all computational cells, in the following form:

$$A_i^e \leq f_i^e \mathcal{A}_r, \tag{7.2}$$

and

$$A_f^e \leq f_f^e \mathcal{A}_r, \tag{7.3}$$

where the A_i^e and A_f^e are the initial and final exterior areas of the composite cell respectively; f_i^e and f_f^e are locally determined variables satisfying $0 \leq k_{min}^e \leq f_i^e, f_f^e \leq k_{max}^e$, with k_{min}^e, k_{max}^e computation-dependent global constants (typically satisfying $0.5 \leq k_{min}^e \leq k_{max}^e \leq 2$, but preferably slightly less than 1); and $\mathcal{A}_r$ the area of an uncut Cartesian cell at refinement level r, where r is the *highest* refinement level in the locality of the composite cell.

A similar set of constraints are imposed for the interior portion of each composite cell, with A_i^i, A_f^i replacing the corresponding exterior areas in equations 7.2 and 7.3. Also f_i^i, f_f^i, k_{min}^i, k_{min}^i must replace the corresponding terms in the above equation, but may possibly take different values. For simplicity, the f's will be assumed global constants (like the k's are) throughout the rest of this chapter.

The above constraints imply other constraints on the minimum and maximum value of the *ratio* of initial to final areas, but this ratio may also additionally be explicitly constrained, and in practice this is necessary to directly control merging-dependent oscillations in the vicinity of boundaries if large local differences in velocity exist between boundaries and the surrounding fluid (for example, during violent

accelerations). These constraints are enforced in the following form:

$$\nu^e_{min} \leq \nu^e = \frac{A^e_i}{A^e_f} \leq \nu^e_{max}, \tag{7.4}$$

and

$$\nu^i_{min} \leq \nu^i = \frac{A^i_i}{A^i_f} \leq \nu^i_{max}, \tag{7.5}$$

where the superscripts and subscripts have the same meanings as they do in equations 7.2 and 7.3 and where ν^e_{min} and ν^i_{min} need not be equal, but preferably between 0.25 and 0.75, and ν^e_{max} and ν^i_{max} need not be equal, but preferably between 1.25 and 4.0. The ad-hoc values given above were found to be a suitable compromise for the typical computation shown in this work between the quality (that is, smoothness) of the solution on interfaces and the displacement with which a boundary could be moved during each iteration (which largely determines the overall processing time for the computation). For simplicity, the various ν values will be assumed global constants in the rest of this chapter.

7.3.3 Admissibility Conditions for Composite Cells

The complete set of conditions that a composite cell must satisfy in addition to the basic, defining, ones given in section 7.1 can now be clearly be stated:

1. geometric invariance; and

2. satisfaction of constraints on the initial and final areas and their ratio, for both the interior and exterior sides.

7.3.4 Single-Sided Merging

For generality, this section has so far assumed that fluid states exist on both sides of an interface and that calculations were necessary in both, the interior and exterior

of every boundary. Whenever this is not the case, as, for example, when there are interfaces separating fluid from solid states, the only additional simplification that can be introduced is elimination of the constraints on the areas and areas ratios in the region(s) where computations are not required; the geometric invariance condition cannot be relaxed in any way.

7.4 Updating Composite Cells

A gasdyanmics update is performed on a composite cell as a whole. After completion of the motion step, the conserved quantities of the exterior and interior states, if applicable, are projected back separately to each of the composite cell's constituent cells in accordance with the final topologic states and partial areas of these cells, and, for higher order accuracy, in accordance with the gradients in the "interior" and "exterior" parts of the composite cell.

7.5 Definitions II: Grid-Line and Edge Types

A **pre-existing grid-line** is a straight line segment that coincides with the edges of a nonempty set of cells that already exist in the discretization. In figure 7.5, the line segments sw and $dgkmprt$ are two examples of such grid-lines.

A **fictitious grid-line** is a straight line segment that coincides with the edges of a nonempty set of cells that do not *all* exist but such that any "missing" cells may be created by refinement of the current grid up to *at most* the preset maximum refinement level. In figure 7.5, the line segments oq and $dgkmprtv$ are two examples of such grid-lines. Assuming that the smallest cells shown in the figure correspond to the maximum allowed refinement level, then any perpendicular bisector of oq cannot be a fictitious grid-line.

For a computational region centered at the origin, with side-length $2l$, and for a preset maximum refinement level of n (where $n = 1$ refers to the original square computational region), fictitious grid-lines may have x or y co-ordinate values only equal to $\pm r\frac{l}{2^{n-2}}$ for some integer r, $0 \leq r \leq 2^{n-2}$, where the condition $n \geq 2$ must be satisfied since no fictitious grid-lines can exist when $n = 1$.

An **admissible grid-line** is one that is either pre-existing or fictitious. In figure 7.5, the line segment rq extended to the meet segment sn is an example of an admissible grid-line, and so is every solid and dotted grid-line shown there.

For any polygon that is intersected by an internal boundary, an **entry** edge is one through which the tangent to the boundary is directed into the polygon's interior, while an **exit** edge is one through which the tangent is directed into the polygon's exterior. In this work, the tangent direction is defined in accordance with an anti-clockwise looping direction around each boundary, but using the opposite convention would have been just as valid.

7.6 The Merging Algorithm: Outline

The conditions that a composite cell must satisfy in order for cell merging to overcome the two problems identified in section 7.3 are presented above. In section 7.10, a theorem is presented that shows that cell merging can be performed on any Cartesian adaptive grid with smooth internal boundaries to form composite cells that satisfy those conditions, and that cover the entire Computational Region.

The merging theorem stops short of proving the existence of a satisfactory total cover for non-smooth internal boundaries (that is, those with tangent vector discontinuities). However, a merging algorithm based on the theorem can be supplemented by heuristic methods for handling such features, making it intuitively possible to

obtain such a cover for any boundary geometry.

The composite-cell cover for a subregion set is not unique, and can be formed on a local basis. This section presents the algorithm that constructs such a cover for a motion step given: (i) the complete geometric specification of the boundaries of the computational region at the start and end of a motion step; and (ii) the subregion set at either the start or the end of the motion step.

The main steps of the algorithm are as follows:

1. **Motion-Envelope Merging:** The cells in the motion envelopes of internal boundaries are merged, one envelope at a time, as follows:

   (a) **Selection of an Initiation Site:** An intersected, geometrically-invariant, pre-existing grid-line that connects the inner and outer boundaries of the current motion envelope, and which is suitable for launching the envelope merging is found.

   (b) **Enforcement of Geometric Invariance:** The initiation segment found in the previous sub-step is examined and possibly modified for greater conformity with the local geometry. Using the possibly modified initiation segment as an entry edge, a geometrically-invariant polygon that straddles the boundary and that has all its edges coincident with *admissible* grid-lines is constructed.

   After completion of the first polygon, a second, intersected, geometrically-invariant polygon with all edges coincident with admissible grid-lines is constructed, using the exit edge of the previous (first) polygon as its (geometrically-invariant) entry edge.

   The formation of geometrically-invariant polygons is repeated as many

times as necessary, advancing along the boundary with each new polygon, until the initiation site is re-encountered, signaling completion of the loop around the boundary.

(c) **Enforcement of Area and Area-Ratio Constraints:** Each geometrically-invariant polygon generated in the above sub-step is expanded or contracted (while maintaining geometric invariance) along edges shared by other polygons to satisfy as far as possible the constraints specified for minimum and maximum areas and for the ratio of these extremes. If expansion or contraction cannot be accomplished without violating geometric invariance, combination with other polygons, and local re-merging are both also attempted.

(d) **Data-Structural Merging:** After the polygons covering the motion envelope have been formed, all the cells that lie within each of these polygons are identified and then merged together. If an edge of a polygon is found to align with a fictitious grid-line, there will be at least one cell that is partially overlaped by the polygon. Any such cell is refined as many times as necessary to eliminate any partial overlap, modifying the original grid and the original motion envelope in the process.

2. **Non-Envelope Merging:** All cells outside the motion envelopes of all internal boundaries are "self-merged". These cells are the ones left un-merged by step 1. This step does not require any geometric computations since any cell outside the merging envelopes is, by definition, geometrically invariant. The process amounts to direct data-structural reconfiguration of each such cell as a single-member composite cell. The non-envelope cells are merged to allow uniform

treatment in the flow-solver, but, otherwise, the procedure is unnecessary.

Sub-step 1 (b) is the most software- and computationally-intensive part of the merging algorithm. As evident from the above description, its sole purpose is to subdivide the current motion envelope by geometrically invariant polygons, ensuring that all parts of the boundary translate only within geometrically invariant polygons. Note that each polygon is formed using only local information. The most frequently used polygon is the rectangle, but non-convex polygons may optionally be allowed as well.

In order to satisfy the geometric invariance condition as strongly and flexibly as possible, sub-step 1 (b) purposely does not construct polygons with edges taken only from segments of pre-existing grid-lines, but instead selects segments from *all* admissible grid-lines. This choice has the following advantages, explained in detail in the next section:

1. the area, shape, and orientation of polygons can be better suited to the local boundary geometry at the start and end of the motion step since the set from which segments can be selected is larger and has a finer "ruling";

2. if the refinement level varies around a boundary, the formation of convex, geometrically invariant polygons requires examination of wider stencils of grid-lines if only pre-existing grid-lines can be used to select sub-segments; and

3. it readily allows identification of the least set of modifications to the local discretization to satisfy failed requirements of the merging theorem, and allows these modifications to be conveniently and controllably introduced.

Sub-step 1 (c) independently satisfies the area and area-ratio requirements for the geometrically invariant polygons formed in sub-step 1 (b) by modifying any

unsatisfactory one. The motivation for separation of sub-steps 1 (b) and 1 (c) is that a conflict often arises between these two sub-steps across small gaps between boundaries; the conflict is best resolved after first satisfying geometric invariance.

In sub-step 1 (d), the merging amounts to the creation of a data-structure that is to retain the extrinsic attributes of the composite cell and to establishing a link between (setting a pointer from) each of the member cells to that structure. The basic tree data-structure is not modified and all member cells retain their individual tree-inherited identities. The member cells are not geometrically combined. This approach is clearly the better alternative under the highly dynamic conditions encountered with unsteady flows, solution adaptation, and moving boundaries.

The formation of each composite cell in step 1 is performed in two separate stages; namely, (i) formation of a geometrically invariant polygon (in sub-steps 1 (b) and 1 (c)); and (ii) the identification, and then combination of the cells lying within that polygon, possibly with modification of the grid (in sub-step 1 (d)). The prime motivation for this separation is that it allows the isolation of the mostly geometric operations (of stage (i)) from the mostly data-structural operations (of stage (ii)), leading to a more modular implementation, at the expense of a slight decrease in computational efficiency.

If a topologic transformation of internal boundaries occurs during a motion step (for example, if there is coalesce or break-up), the boundary definition must be modified and the merging must account for this in a manner that maintains conservation and accuracy. This topic is discussed in more detail in Section 7.9.

7.7 The Merging Algorithm: Details and Implementation

This section presents the details of the methods used in each of the steps given in the outline of the merging algorithm. A briefer overview of the capabilities and the operation of the algorithm may be extracted from section 7.8.

7.7.1 Initiation of Motion-Envelope Merging

An initiation segment must be an intersected, geometrically invariant sub-segment of a pre-existing (not a fictitious) grid-line, and must lie wholly within the motion envelope. In figure 7.1, the left and right edges of cell S are both suitable initiation segments, and so is the left edge of the shaded rectangle enclosing cell I'.

In addition to the above mandatory conditions, other optional ones are also applied to avoid initiation sites that could introduce complications and undesirable effects at a subsequent stage in the merging process. One such condition allows simplification of the test for the wrap-around condition required for termination of step 1 (b). Another set of conditions is aimed at avoiding composite cells with an unnecessarily large number of member cells; in particular, parts of boundaries in the vicinity of tangent vector discontinuities, or where the local displacement is large, or where the local variation in refinement level is large are all avoided.

7.7.2 Enforcement of Geometric Invariance

As explained in the previous section, the first step in the formation of a composite cell cover within motion envelopes determines the most appropriate shapes and locations for composite cells, first by forming polygons that straddle the boundary and only satisfy the geometric invariance condition, and then, by modifying these polygons to satisfy the area and area ratio constraints.

7.7.2.1 Formation of Geometrically Invariant Rectangles

The purpose of this procedure is to construct "optimal" geometrically-invariant rectangles whose edges fall on admissible grid lines. An optimal rectangle is one that has the simplest possible intersection pattern with all boundary positions in the motion step, has the smallest possible area, the least *unnecessary* incompatibility with the existing local discretization, and that is constructed with the least amount of geometric computations.

The rectangle formation procedure is localized, and based on projection and examination of "trial" line-segments coincident with pre-existing or fictitious grid-lines to determine suitable, geometrically invariant edges. These edges are eventually assembled into a closed, geometrically invariant rectangle. Decisions in the process are guided by heuristics seeking to satisfy the geometric invariance condition as strongly as possible while minimizing the overall anticipated number of geometric calculations.

The selection of the length of each projected trial edge accords with a length-scale drawn from the local refinement-level pattern. The selection of the projection direction of each edge, and the selection of the grid-line with which it aligns accord with the local intersection pattern and the local displacement of the boundary across the motion step.

The most significant steps in the formation algorithm for geometrically invariant rectangles are described with reference to figure 7.3 which shows the motion envelope and the initial and final positions for a moving and deforming boundary. The arrowed circular arcs indicate the geometrically invariant rectangles that are constructed, as well as the construction "direction". Arrows indicating the projection direction for edges are also shown, but only for the first rectangle constructed.

Assume that the segment ayb is either the exit edge for a previously-formed rectangle or the initiation segment for the current envelope, and assume that the merging length-scale throughout is equal to the edge-length of each of the cells shown in the figure.

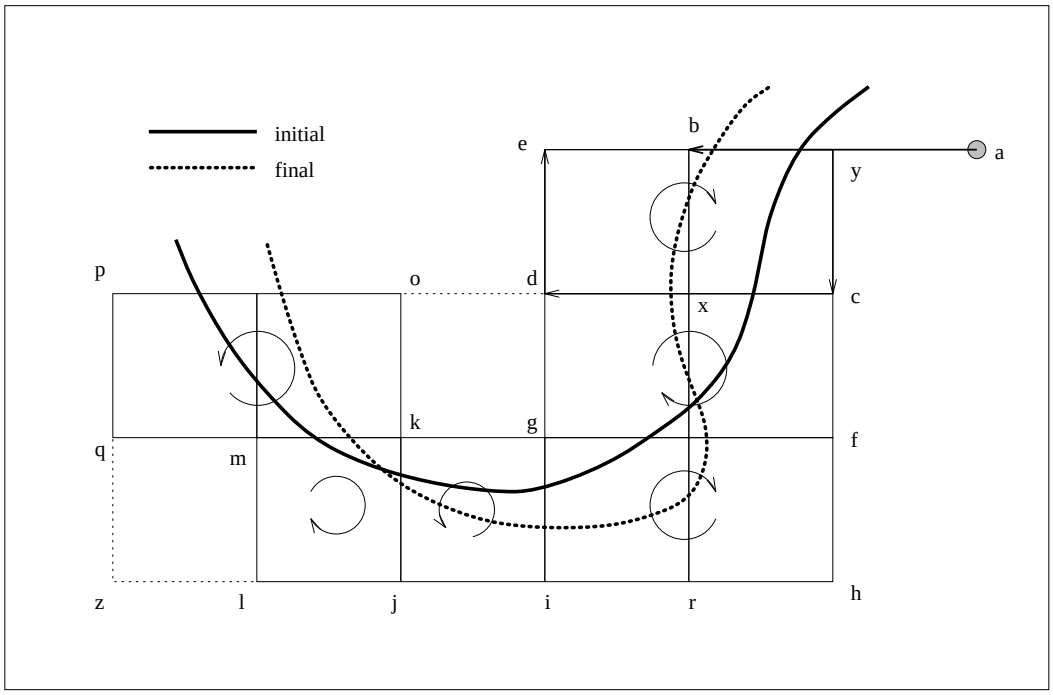

Figure 7.3: Construction of geometrically invariant rectangles.

Step 1: This step examines the suitability of a geometrically invariant edge as an entry edge, and possibly modifies it. In the example of figure 7.3, since both intersection points in ayb lie within its sub-segment yb, and since the local length scale equals the length of that sub-segment, the entry edge for the next geometrically-invariant rectangle to be constructed is re-defined to be the segment yb. The length-scale associated with an entry edge must always be equal to the dimension of an admissible cell in the quadtree, and is most often equal to the dimension of the largest cell enclosed by the previously-formed rectangle.

Depending on the length scale and intersection pattern, a line segment may be expanded, contracted, shifted along the grid-line it falls on, or left unmodified before re-using it as an entry edge. An example of each of these situations will arise during the course of this chapter.

Step 2: This step determines the formation loop direction (that is, the construction direction) for a rectangle. This is done by examination of the local intersection pattern, the curvature of the boundary spline(s) at the local intersection points, and the positions of points further along the boundary spline(s) in the merging direction. For the example at hand, the original direction will be reversed to be from b to y, although ultimately this reversal will not alter the resulting rectangle.

At the end of step 2, the first geometrically invariant edge for the rectangle is considered tentatively defined.

Step 3: This step projects a second tentative edge, with length equal to the local merging length scale, from one of the end-points of the edge defined in step 2. The end-point and the projection direction are both selected according to the formation loop direction. Any such projected edge is guaranteed to coincide with an admissible grid-line. In the example of figure 7.3, the projection will be from point y toward point c, defining the segment yc. The projected edge is examined for geometric invariance (by forming ordered sets of any intersection points at the initial and final boundary positions and comparing these sets).

If the geometric invariance condition is satisfied, the edge is tentatively admitted.

If the condition fails because the edge is multiply intersected, the edge is deleted and the algorithm returns to the previous (first) edge and attempts to extend that edge in the formation loop direction while preserving its geometric invariance. If the attempt succeeds, the projection attempt for the second edge is repeated. If

the attempt fails, the the formation loop direction is reversed and step 2 started anew. If no progress can be made beyond step 2, the algorithm returns to define a more suitable entry edge, possibly re-defining the previously-formed rectangle in the process.

If the geometric invariance condition for the second edge fails because exactly one of the boundary positions fails to intersect it, the edge is extended and re-examined, one local length scale at a time, up to a prespecified number of extensions (dependent on the geometry, but typically 3). If the condition still fails, the formation loop direction is reversed and an edge is projected from the opposite end of the redefined entry edge. The alternative edge is re-examined and extended if needed, again up to a prespecified number of extensions. If the modified alternative edge still fails the condition, the original projected edge is re-extended and re-examined as many times as necessary, but now up to a second prespecified number of extensions (typically 7). If the condition still fails, the alternative projected edge is re-extended as many times as needed up to the second threshold of extensions. If the geometric invariance is still unattained, the algorithm returns to the redefine a more suitable entry edge.

In order to minimize the number of intersection evaluations, any new intersection point encountered during extension of a segment is stored in case it is needed again during a re-examination.

For the example of figure 7.3, the edge yc satisfies the geometric invariance condition trivially since it contains no intersection points.

At the end of step 3, the first two geometrically invariant edges are considered tentatively defined.

Step 4: This step projects the third trial edge of the rectangle from the "free" end of the second trial edge in the direction given by that of the formation loop, and with

length equal to the local length scale. In the example at hand, this procedure defines the directed segment cx. The projected segment is re-extended and re-examined as needed to satisfy geometric invariance or until thresholds on the number of extension attempts are encountered, or until the segment becomes multiply-intersected, just as done in step 3. If the first such threshold is encountered, or if the edge becomes multiply-intersected, the algorithm returns to the previous (that is, the second) edge and attempts to extend that before returning to the third edge. The local back-tracking may go as far as redefining the entry edge and reversing the formation loop direction. In the case shown, the segment cx requires only one extension (to cxd) to satisfy geometric invariance.

In addition to being geometrically invariant, the third trial edge must also be no less in length than the first (entry) edge. Otherwise, closure of the rectangle being formed cannot be satisfied. Failure of this condition is remedied by re-extension and re-examination, with possible back-tracking as needed, following the procedure described above.

At the end of step 4, the first three geometrically invariant edges are considered tentatively defined.

Step 5: This step projects the fourth trial edge of the rectangle in accordance with the formation loop direction. The local length scale is over-looked in this step since, for closure of the rectangle, the length of the fourth edge must be equal to that of the second edge.

For the case shown, this defines the directed segment de. The projected trial segment is examined for geometric invariance, and if the condition fails, the trial edge is deleted, and the algorithm returns to the previous (third) edge to re-attempt satisfaction of geometric invariance by extension. The back-tracking may again go

all the way back to redefinition of the entry edge. In the example at hand, the edge *de* satisfies the geometric invariance condition and is admitted as valid.

At the end of step 5, all four geometrically invariant edges of the bounding rectangle are considered tentatively defined.

Step 6: This step re-examines the entry edge for closure. If the rectangle is not closed, the algorithm attempts to extend the entry edge without violating its geometric invariance. Failure to achieve this will result in deletion of the fourth edge and initiation of descending back-tracking, starting from the third edge, possibly all the way back to redefinition of the entry edge.

In the example of figure 7.3, the left-sided extension *eb* satisfies the closure requirement without violation of geometric invariance.

After closure is established, the number of cells falling within the rectangle is evaluated. If the number of enclosed cells is found to be excessive, or if all cells enclosed by the rectangle are only partially overlaped, step 6 causes a change in the merging length-scale and returns to step 1 (to re-build a new rectangle with the new length-scale) after deletion of the rectangle just formed. This procedure is described in detail in the next sub-section.

Once the sixth step is passed, a geometrically invariant rectangle (*ycxdeby* in the example shown) would have been formed and the algorithm is ready to proceed to the next rectangle, after completion of the seventh and eighth steps, unless the process is terminated in step eight.

Step 7: This step reconfigures the exit edge (*cxd*, in the example shown) of the rectangle just formed as an entry edge for the next rectangle.

Step 8: This step checks whether slight dilation of the rectangle just formed results in overlap with the first rectangle formed on that boundary. This is the

preliminary test used to determine completion of the rectangle formation loop around the boundary.

If the result is positive, the starting and current arc-length parameters along the spline are compared to determine whether the overlap is caused by boundary "folding" rather than by completion of the loop around the boundary. If wrap-around has occured, the local boundary geometry and the rectangles are examined to determine how best to perform the "closure" operation. In some cases, the initial and final invariant rectangles already align along a shared grid-line and no further action is necessary. In others, the rectangles overlap. Overlap is eliminated either by fusion of the two overlapping rectangles (if the resulting union is rectangular and does not exceed a certain area threshold), or, by deletion of the overlapping rectangles and reconstruction of one or more rectangles to cover the "non-merged" part of the motion envelope. Both methods use only the basic edge operations already described.

Once any necessary wrap-around operations are performed, the rectangle formation loop for the current boundary is terminated. If the wrap-around condition is not satisfied, formation of the next rectangle is initiated, starting with step 1 again.

For the example of figure 7.3, the next rectangle formation will start with edge cxd. In this case, the direction of the edge will first be reversed (to dxc) to obtain a clockwise formation direction for the rectangle. Then, the segment cf, which is geometrically invariant, is projected. Then, the first half of segment fg is projected and tested, and, as a result, it is extended to the geometrically invariant segment fg. Then, the edge gd is projected successfully, and the closed rectangle $dxcfgd$ is admitted.

The clockwise formation loop direction is retained for the next rectangle resulting in $gfhrig$. This is followed by reversal of the formation direction to give rectangle

igkji, followed by another reversal, resulting in rectangle *kjlmk*, and by another reversal to give *mkopqm* as the last rectangle in the example.

Now that the overall rectangle formation procedure has been described, the importance of selecting the "better" formation loop direction from among the two possibilities (in step 2) can be well-understood. In some cases, the wrong selection can lead to non-optimal rectangle shapes or sizes. In the extreme, it may be impossible to satisfy geometric invariance. Figures 7.3 and 7.4 provide examples of both cases.

In figure 7.4, assume that the entry edge for the next rectangle to be formed is *ab*. The smallest geometrically invariant rectangles that result with a clockwise and with an anti-clockwise formation loop are both indicated; these rectangles can be mentally constructed by following steps 1-8 for the given geometry. The clockwise loop will ultimately result in the composite cell A ∪ Z ∪ Y ∪ X ∪ W; the anti-clockwise one will result in the composite cell A ∪ V. If the final boundary position is deformed upwards so that it never intersects the extension of segment *bc*, the clockwise direction will fail to define a geometrically invariant rectangle.

In rectangle *mkopqm* in figure 7.3, depending on whether the final boundary position intersects the extension of the segment *lm*, and on how far along the extension any such intersection first occurs, the formation direction would be crucial. For all the other rectangles shown in that figure, the formation direction happens to have little or no effect on the resulting rectangles.

The merging theorem in section 7.10 shows that, provided the boundary is sufficiently smooth, if one of the formation directions fails to form a geometrically invariant rectangle, the other *must* succeed.

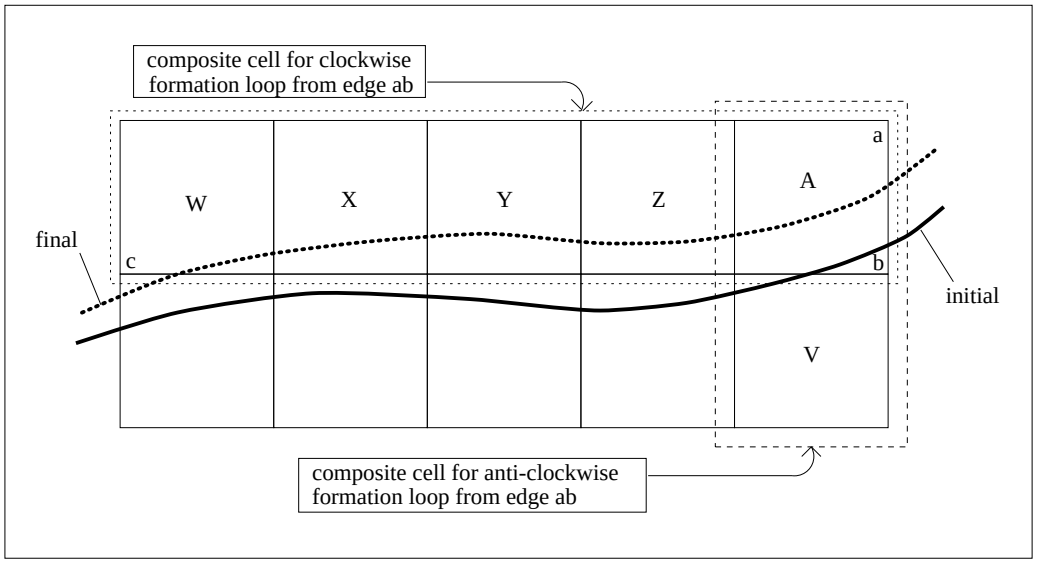

Figure 7.4: The importance of correct choice of the formation loop direction.

7.7.2.2 Formation of Geometrically Invariant Rectangles with Refinement Level Variations

In the previous subsection, the procedures used to form geometrically invariant rectangles were given and illustrated by example for the special case of uniform refinement in the motion envelope. In the actual algorithm, most of the above steps involve additional, subsidiary ones to enable the handling of refinement variations that are arbitrary, save the one-refinement-level-difference-rule between edge neighbors.

The three most important mechanisms for dealing with the additional complications introduced by non-uniform refinement are:

1. increase or decrease (always by a factor of two) in the length-scale used in forming rectangles;

2. reversal of the formation loop direction; and

3. re-alignment of segments, either by "sliding" them along grid-lines, or by shifting them to different, parallel grid-lines.

The methods used to effect the most important of these actions are presented through examples arising in the portion of the motion envelope shown in figure 7.5 in which the selected rectangles and formation directions are indicated by arrowed arcs.

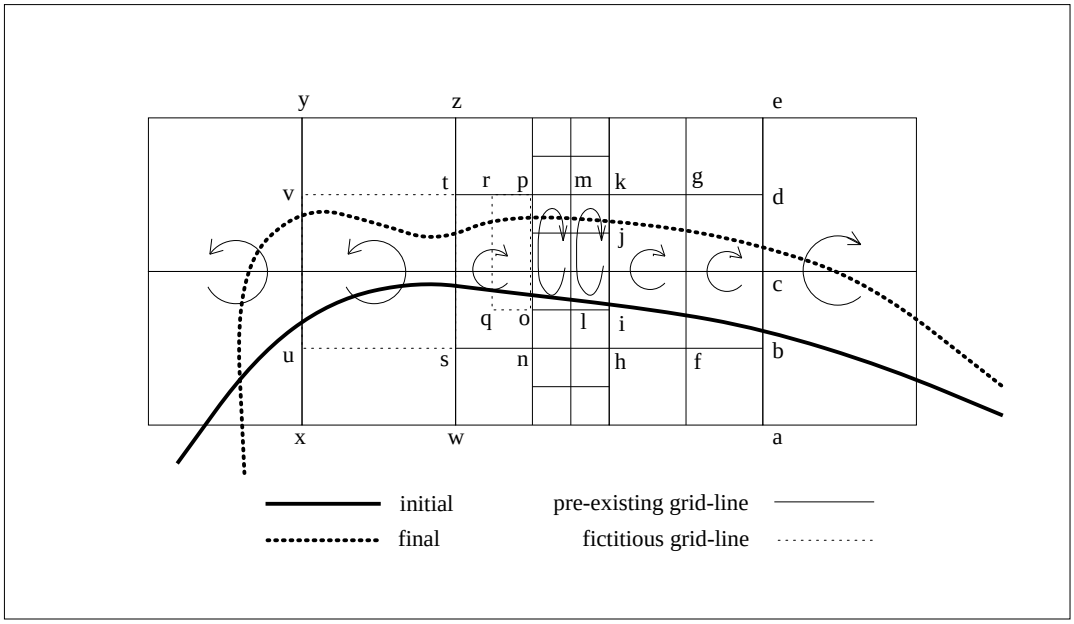

Figure 7.5: Construction of geometrically invariant rectangles with change of length-scale, with reversal of formation-loop direction, and with line-segment re-alignment.

Starting from the right of figure 7.5, assume that the first invariant rectangle formed is that having edge *abcde* on its left, with length-scale equal to the dimension of each of the cells it encloses. Since the initial value of the length-scale of a new rectangle is equal to that of the previously-formed one, the next tentative rectangle to be formed, given the geometry, is the one coincident with the reflection of the first rectangle in edge *abcde*. This rectangle encloses eight cells.

As explained briefly in step 6 of the previous subsection, before any rectangle is finally admitted, all cells overlapping it are identified in order to determine whether a change in length-scale is advisable.

If the area of a rectangle or the total number of cells it overlaps exceed prespecified thresholds, and if the magnitude of the boundary displacement, and the simplicity of the local intersection pattern allow it, the following sequence of events occurs:

1. the rectangle just created is deleted;
2. the length-scale is reduced (by a factor of two);
3. if necessary, the length of the entry edge is changed to be more compatible with the new length-scale (during step 1 of the rectangle formation process); and
4. another rectangle using the new length-scale is formed.

In the example at hand, the number of cells enclosed by the rectangle just formed (eight) exceeds the typical threshold (4-6 cells). This triggers a reduction in the merging length-scale. The rectangle just formed is deleted, and the length-scale reduced. Step 1 of the previous sub-section causes a contraction of the original entry segment, *abcde*, to the segment *bcd*, since this is the smallest geometrically invariant intersected segment that has end-points corresponding to the refinement level associated with the new merging length-scale. Following steps 1-8 of the previous section, the rectangle formation process proceeds to yield the rectangle $bfgdcb$. The exit edge fg subsequently becomes an entry edge, and the next rectangle to be formed is $gfhijkg$, with no change of length scale.

The next rectangle formation similarly initiates a reduction in length-scale after the first trial rectangle, $hnopmkjih$, is found to overlap eight cells. The exit edge $hijk$

is contracted to ijk and the next two rectangles to be formed with the new length-scale are $ilmkji$ and $lopml$. Note that in these two rectangles, the restriction on the maximum local boundary displacement is violated, but the merging still succeeds.

If the number of cells that are strictly partially-overlapped by a rectangle exceeds a certain ratio of the total number of overlapped cells, an increase in the merging length-scale is triggered. This occurs during the attempt to form the next rectangle in the example of figure 7.5, when the trial rectangle $oqrpo$ is found to only partially overlap two cells. In such a case, the following sequence of events occurs:

1. the rectangle just formed is deleted;
2. the length-scale is increased (by a factor of two);
3. if possible, the entry edge is redefined (by re-alignment, or by expansion or contraction of the segment at one or more of its ends) in accordance with the new length-scale; and
4. another rectangle using the new length-scale is formed.

In the example at hand, the smallest geometrically invariant entry edge derived from the given exit edge and compatible with the new length scale is pn. The rectangle formed with entry edge pn and the new length scale is therefore $ponstrp$.

Note that the only way to resolve partial overlap without increasing the merging length-scale is to refine the cells that are partially overlapped. In the example at hand, refining each of the two cells that are partially overlapped by $oqrpo$ gives eight cells, three of which lie completely within the rectangle. In general, this is not desirable, since it is best to let the refinement level be dictated only by the geometry and solution refinement requirements. Moreover, allowing such unnecessary refinement, even locally, may propagate by a large number of cells around a boundary.

The next entry edge is now ts. The rectangle that would be formed with that entry edge, and the given geometry and length-scale, is $suvts$, indicated in figure 7.5 by dotted lines. This rectangle only partially overlaps two cells, and its admission would force their unnecessary refinement to obtain total overlap, as explained above. In this case, eight new cells will be created, four of which lie completely within the rectangle $suvts$.

As explained above, the merging length-scale is doubled and the rectangle $suvts$ is deleted. However, the vertices of the smallest invariant segment with the new length-scale, ts, do not fall on grid-lines corresponding to the new length-scale, so partial overlap would still not be prevented. This situation is handled by re-alignment of edge st (in step 1) by translating it downwards to the nearest vertex that corresponds to the new length-scale, that is, point s becomes coincident with w.

This need for re-alignment of entry edges arises only during the transition to lower refinement levels. Without this re-alignment, large portions of the original merging envelope could be unnecessarily modified. Note, however, that re-alignment is not always necessary. For example, there was no need to re-align the vertex p of the entry edge po for rectangle $ponstrp$, since it already falls on a grid-line compatible with the new length-scale.

After re-alignment of edge ts in the example of figure 7.5, step 1 causes upward expansion of the segment by one length-scale, to wz. The new geometry dictates a reversal in the formation direction and the next rectangle formed will be $ztswxuvyz$.

The final rectangle to be formed in the figure is that coincident with the two cells to the left of the segment $xuvy$.

The above discussion described the main methods of handling arbitrary variations in refinement around boundaries. The role of the formation loop direction and its

reversal has already been discussed. The need to shift edges to parallel grid-lines arises very rarely in practice, but the idea there is to translate the entry edge to a more favorable position and then perform local remerging. The occasional deletion and reconstruction of rectangles required with refinement level variations is a penalty that should be willingly paid in exchange for the flexibility allowed by the loose coupling between the rectangle formation process and the local discretization.

Several examples with large differences in refinement level can be seen in the grid plots in Chapter VIII.

7.7.2.3 Formation of Non-Convex Geometrically Invariant Polygons

So far, the discussion in this section has addressed only rectangular geometrically-invariant polygons. Since all edges of a polygon must be aligned with grid-lines, non-rectangular shapes must also be non-convex. The algorithm optionally allows the formation of non-convex geometrically invariant polygons, and this is accomplished primarily by trying several projection directions for each new edge before choosing the one that appears to minimize the area of the polygon, and by relaxing the restriction on the number of edges, and on the lengths of the third and fourth ones. These restrictions are not removed completely, however, in order to retain control, through user-specified parameters, on the severity and frequency of non-convexity. Several examples of non-convex merging appear in figure 7.6.

Allowing non-convexity reduces the average polygon size, as shown in figure 7.6. However, the algorithmic complexity and the computational cost both increase because of the large increase in the number of alternatives that must be examined.

If non-convexity is allowed, most instances of it are usually introduced not during the creation of geometrically invariant rectangles, but as a result of polygon "fusion"

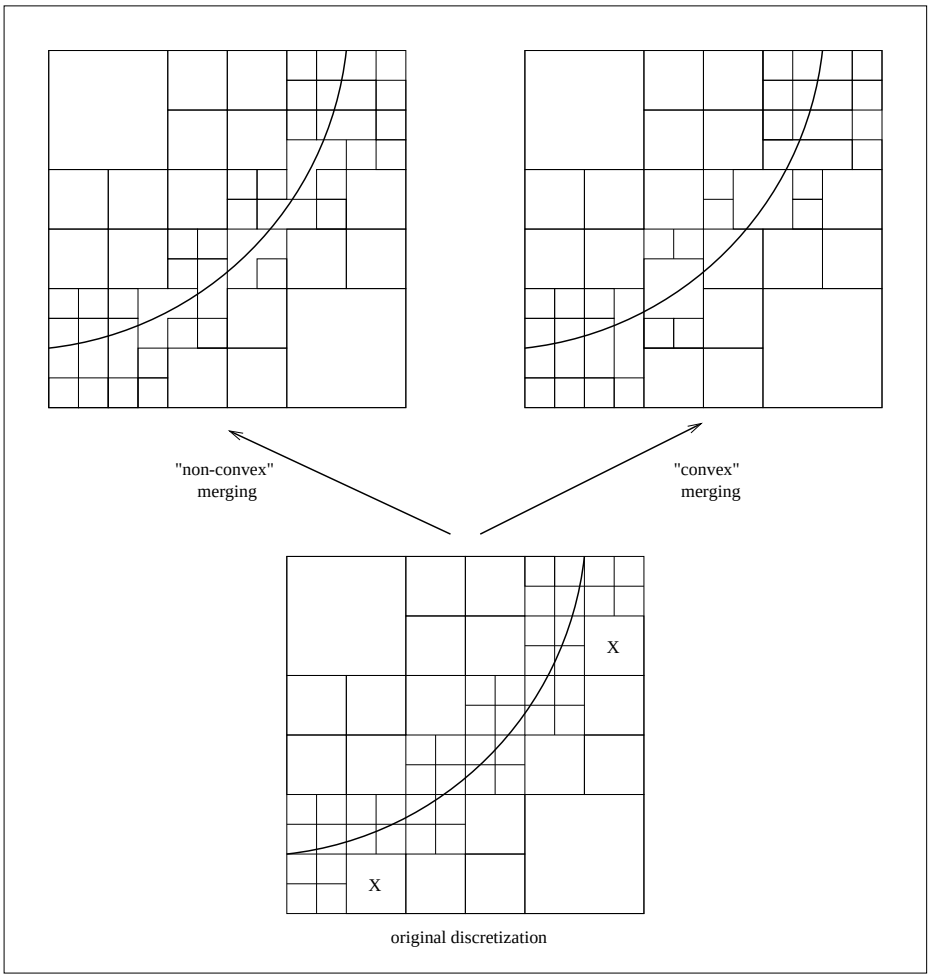

Figure 7.6: Convex vs. non-convex merging for the same discretization; all parameters of the merging algorithm are identical, with uniform $f_i^e = f_f^e = 0.9$, and uniform $f_i^i = f_f^i = 0.2$.

during their modification to satisfy the area and area-ratio constraints. For this reason, this topic is further discussed in the next sub-section.

Although the reduction in average polygon area that accompanies non-convex merging may have a slight advantage in a first-order-accurate calculation, it is normally disadvatageous to choose non-convex computational-cell shapes with a higher-order scheme because the gradient limiting would typically reduce the reconstruction order from first (or higher) to zeroth order. In situations where accuracy is not impor-

tant and only a coarse-grid calculation is desired, allowing non-convexity to overcome local violations of the theoretical restrictions on boundary shape and displacement could be helpful.

7.7.3 Enforcement of Area and Area-Ratio Constraints

The constraints of equations 7.2, 7.3, 7.4, and 7.5 are satisfied by modifying the geometrically invariant polygons covering the current motion envelope which have already been generated in sub-step 1 (b). Three methods of modification are attempted:

1. fusion of contiguous polygons;

2. expansion of polygons along intersected edges; and

3. local re-merging of two or more polygons.

Regardless of the method used, any modification must not introduce geometric non-invariance.

Fusion of contiguous polygons is used when it is not possible to expand polygons outward (because all their exterior sides are intersected) or inward (because all their interior sides are intersected). Expansion of polygons by outward or inward relocation of unitersected edges is used when there are such edges and when this would not result in overlap with other polygons (whether belonging to the current boundary or a different one). If expansion is not possible without overlap, then the polygons that overlap may be fused together to define a new single polygon. This is often necessary in the vicinity of approaching or "folding" boundaries. The algorithms used to fuse, to expand, or to re-merge are mostly based on heuristics.

Unlike the geometric-invariance requirement, satisfaction of the area and area-ratio constraints is not essential. Failure to achieve it will at most temporarily restrict the next global time-step (through the stability constraints of the flow-solver). For this reason, the attempt to satisfy these constraints is made separately from and only after a complete, geometrically-invariant cover of the motion envelope has been constructed.

The extent of fusion and outward expansion are controlled mostly through the area and area-ratio constraint parameters (of equations 7.2, 7.3, 7.4, and 7.5). Figure 7.7 shows the composite cells that are generated after satisfying the area and area-ratio constraints with and without the non-convexity option. The two cases have identical geometries and all other setting in the merging algorithm are the same. The differences between convex and non-convex merging are much less pronounced for lower f^e and f^i.

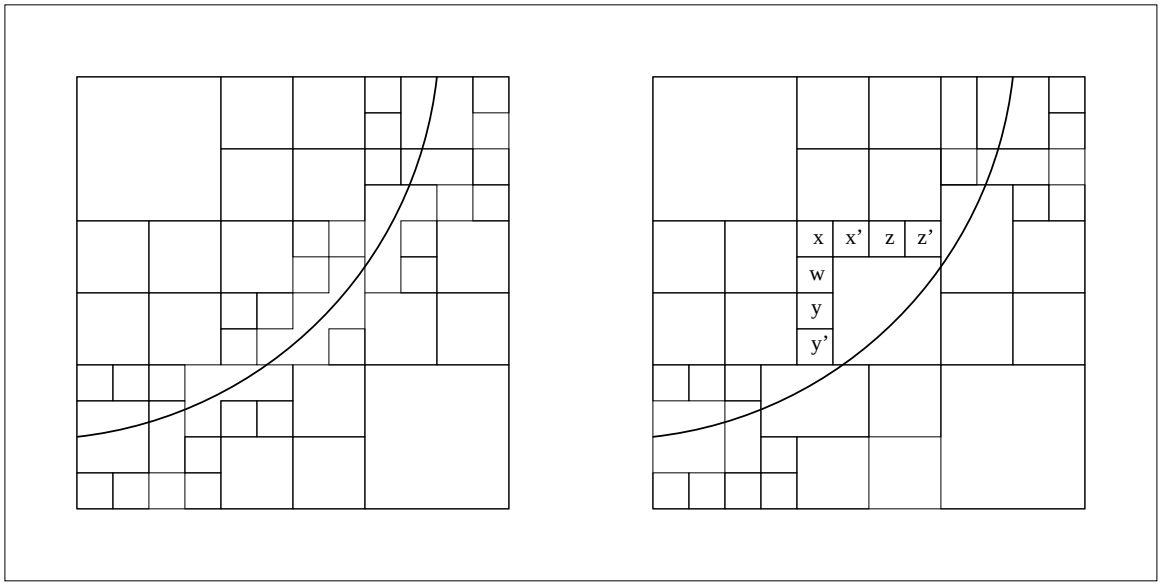

Figure 7.7: The effects of increasing f with convex and non-convex merging; here from the uniform values of the previous figure to the uniform values $f^e = 1.5$, $f^i = 0.3$.

7.7.4 Treatment of Stationary Boundaries

For stationary boundaries, all cells in the subregion set are, by definition, geometrically-invariant polygons. This makes sub-step 1 (b) redundant, since automatically reconfiguring each cell as a composite one would accomplish the sub-step's objectives. Nevertheless, it is still necessary to construct the geometrically polygons that cover the "motion" envelope and to identify the precise intersection patterns of these polygons since the polygons are needed as the starting point for enforcement of the area and area-ratio constraints in motion envelopes.

7.7.5 Determination of Composite Cell Members

As described above, the formation of geometrically invariant polygons is guided most strongly, as it should, by the local geometry and local intersection pattern, rather than by the local grid refinement pattern. Once a polygon has been defined, the next step is to determine all the cells falling within it and then to merge them together into one composite group of cells.

Identification of all the cells that overlap a polygon (whether partially or wholly) is efficient and straightforward, especially if the polygon is rectangular. It is possible to use the quadtree data-structure itself to perform an efficient search for all such cells starting from the root node, but the algorithm instead follows a method that localizes the search as much as possible, for the greatest possible savings in computational effort. This is done by using mostly neighbor-finding operations and using the cell from which the boundary emerged from the previous invariant polygon as the locus of the search. This "tracker" cell is updated after every composite group is formed. Also, to improve efficiency, a list of neighbors is maintained and updated only as necessary to minimize the number of invocations of neighbor finding operations as

the marching proceeds around the boundary.

Due to the construction method of the geometrically-invariant polygons, cases of partial overlap are possible. Cell *suvts* in figure 7.5 shows a typical example. Whenever partial overlap occurs, it can always be eliminated by refinement since there must exist a fictitious grid-line that coincides with every edge of every geometrically-invariant polygon. In the example mentioned above, refinement would divide each of the two partially overlapped cells, creating eight cells, four of which lie completely on or within the rectangle *suvts* and so would be merged together in one group. In addition to situations which require refinement, there are also situations which call for local coarsening of a grid, but these arise only during re-merging.

Refining a grid to eliminate partial overlap implies that the merging process may modify the original discretization, and hence the original motion envelope. This is to be avoided since the refinement distribution should be determined as largely as possible by the geometric and gas-dynamic refinement criteria. For this reason, the formation of invariant rectangles has a "feedback loop", as described above, that greatly reduces unnecessary modification of the discretization. The two most important mechanisms of providing this feedback are through modification of the local length scale, and through re-alignment of edges and the associated process of contraction or expansion during re-definition of entry edges; these situations are discussed above and illustrated in figure 7.5. In the case of rectangle *suvts*, re-alignment eliminates the need to refine the overlapped cells as explained previously.

Once all the overlapping cells are identified and any necessary refinement has been carried out to eliminate partial overlap, the actual data-structural merging is performed. In this work, the data-structural merging amounts to the allocation of a data-structure that will hold the group's extrinsic variables and to setting the

"merged group" pointer of each of the members of the group to this data structure. Initialization of the extrinsic properties of the composite cell occurs later in the process.

In this work, no geometric merging is performed. Each cell retains its independent identity in the tree. As mentioned in the introduction, geometric merging is not essential. For a grid based on a tree data-structure and when most operations utilize that structure, there is no advantage to geometric merging. Even if no tree data-structure is being used, any minor potential advantages for steady flows will be detrimental to efficiency in the rapidly changing grid and merging patterns required for unsteady flows and moving boundaries. Thus, the original Cartesian cells fully retain their identity, and the composite cell configurations are built and destroyed without altering these original cells (other than through reseting their "merged group" pointers to the appropriate composite cell to which they belong).

The bounding rectangles are retained in memory until completion of step 1 (d). After that they are destroyed, leaving only the composite cell data-structures. The total memory requirement increases slightly during merging to store the data-structures for the bounding polygons.

7.7.6 Non-Envelope Merging

The merging for these cells is trivial since each cell is un-intersected at each stage of the motion step and is therefore geometrically invariant, regardless of whether there is boundary motion. No geometric computations are necessary, and the process amounts to "self-merging". Since step 1 may result in modification of the original motion envelopes by refinement or coarsening, the non-envelope cells cannot easily be determined before completion of the envelope merging.

7.8 An Example of Envelope Merging

The companion figures 7.8 and 7.9 illustrate the main steps in and the key features of envelope merging for a typical, realistic geometry. This example is taken from an actual case. Note that the sharp corner violates the restrictions required by the merging theorem but the algorithm successfully handles it. The two figures are best examined together. In figure 7.9 the non-envelope cells are indicated by dotted lines and the numbers appearing in figure 7.8 have been suppressed for clarity.

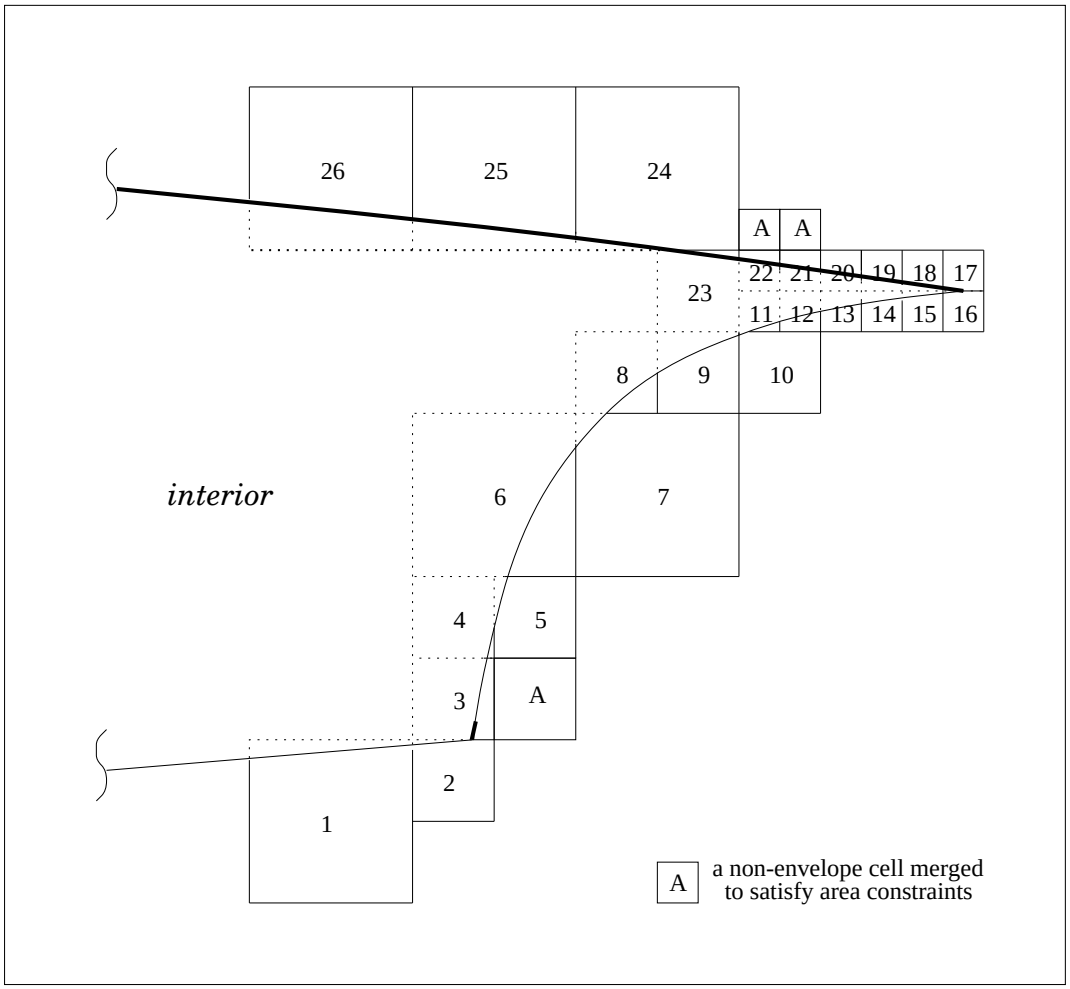

Figure 7.8: Cell merging in the "motion" envelope of a stationary boundary.

Figure 7.8, which shows a portion of a stationary boundary gives an example

of the envelope merging procedure. The cell numbered 1 may be assumed to be the initiation site. Since the boundary is stationary, every intersected cell is by definition geometrically invariant and so will itself define a single-member composite group. Since the boundary exits from cell 1 into cell 2, the latter will be the next cell to be checked for geometric invariance and merged with itself. This process advances along the boundary from cell to cell as described above, all the way to cell number 26. Note how the size of composite cells changes in accordance with the refinement around the boundary. This is dicussed in more detail below.

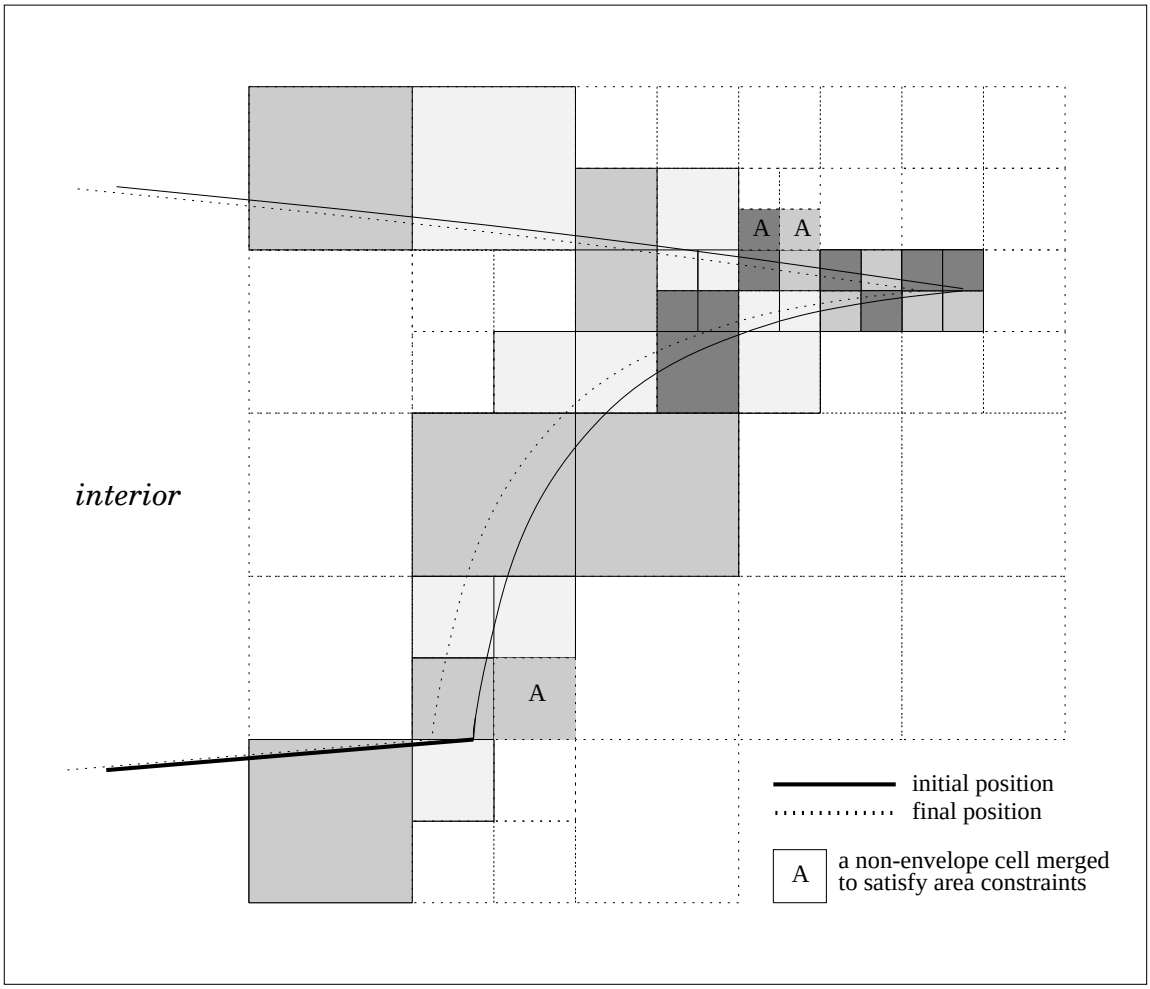

Figure 7.9: Cell merging in the motion envelope of a moving boundary.

For the moving-boundary case shown in figure 7.9, starting at the shaded cell corresponding to cell 1 in figure 7.8, the local length scale is set to the dimension of cell 1 and the looping direction is set to clockwise. Starting from the top edge, each edge of the cell is found to be geometrically invariant and therefore the cell need not be combined with any other. The exit edge of the composite (single-cell) cell leads to the definition of an entry edge for cell 2. Using the length-scale of the previous composite cell results in a rectangle that includes three unintersected cells, in addition to cell 2. This signals the need to reduce the length-scale to the dimension of cell 2. After this is done, forming a composite group starting at the exit face of cell 1 leads to a composite group containing only cell 2 since all edges of that cell are geometrically invariant. The same follows for cell 3 but this time, it is not necessary to change the length-scale again.

Unlike the case for the stationary boundary, the East face of cell 4 is no longer geometrically invariant in the moving-boundary case. The sequence of events for the moving-boundary is as follows: starting from the entry edge (the Southern face of cell 4), and going in a clockwise loop, inspection of the Eastern edge reveals geometric variance. The algorithm returns to the Southern edge and extends it by the current length scale. The second edge will now coincide with the (geometrically invariant) Eastern face of cell 5. The third edge will be the (geometrically invariant union of the North edge of cell 4 and the North edge of cell 5) while the fourth edge will coincide with the Western edge of cell 4. The rectangle defined in the above manner is found to contain cells 4 and 5 so these are merged together to form the fourth composite group. Similarly, cells 6 and 7 are merged together. Cell 8 must be merged with a cell to its West, while cell 23 must be divided into four and its lower two leafs must be merged with cell 9. Cells 10, 11, and 12 are merged together, while cells 13,

14, 19, and 20 are self-merged. Cells 15 and 16 must be merged together to retain geometric invariance on the North edge of the composite cell. For a similar reason, cells 17 and 18 must be merged together. Cell 24 must be divided and its lower right leaf merged with the two top leafs of cell 23. The rest of the mergings are rather clear. All the member cells of a composite cell are given the same shading in figure 7.9.

In the above examples (of figures 7.8 and 7.9), assuming that the minimum initial and final areas of composite cells are required to be no less than half the area of the smallest un-cut cell, several composite cells must be modified to satisfy these constraints. In particular, cell 3 will be combined with its shaded neighbour (marked "A") while each of cells 21 and 22 will be combined with the shaded cell directly above it (which are also marked "A"). In the case of cell 3, the combination with the non-envelope cell is accomplished by outward expansion of the invariant rectangle of cell 3 in the direction of the East edge of cell 3. In the cases of cells 21 and 22, the combinations are accomplished by outward expansion in the direction of their Northern edges.

7.9 Merging with Topologic Transformations in Boundaries

During any topologic transformation, it is impossible to satisfy the condition of geometric invariance for all cells in the merging envelope(s) concerned. In any motion step in which a boundary changes its topology, the topologic changes are performed on a local basis at the beginning of the step. The parts of the boundary or boundaries that are not affected by these changes maintain the trajectories that were presecribed for them at the beginning of the motion step, and the merging is performed for the new geometry and the corresponding new motion envelope.

All topologic transformations are simulated by instantaneous events that occur at the beginning of a motion step. If cells that are initially of the "out" or "on" type become of the "in" type as a result of a topologic transformation, then the associated state vectors are determined by extrapolation of the available neighboring states within the interior of the boundary. Similarly, cells that transform from the "in" or "on" type to the "out" type as a result of a topologic transformation have their state vector determined by extrapolation of the available surrounding exterior cells. Clearly, this procedure is neither accurate nor conservative.

7.10 The Merging Theorem: Existence of a Composite-Cell Cover

The merging theorem establishes the geometric conditions under which the motion-envelope envelope of a boundary can be completely subdivided into geometrically-invariant composite cells. The significance of the theorem is that Step 1 (b) of the merging algorithm can be accomplished provided these geometric conditions are met. The theorem can be proved by geometric construction in which a boundary meeting the stated conditions is made to curve on a grid in the most unfavorable manner for merging. The geometric conditions that must be satisfied are found from this constructive proof to be as follows:

- $\forall p \in B, d(p_i, p_f) \leq D$, where B is a boundary, $d(.,.)$ is the Eucledian norm in $\mathbf{R^2}$, p_i and p_f are the initial and final positions respectively of point $p \in B$, and D is the dimension of the smallest cell within which p_i and p_f lie;

- $\forall p \in B, \rho_p \leq 5D$, where D is as above and ρ_p is the radius of curvature at point p; and

- $(\forall p, q \in B)[(s(p,q) \geq (10 arcsin(1/5))D) \Longrightarrow (d(p,q) \geq 2D)]$ where D and $d(.,.)$ are as above, and $s(.,.)$ is the arc-length along the curve defining the boundary B.

The most important observations regarding the theorem is that the constraints are entirely expressible in terms of a comparison between the length-scale of the local grid discretization on the one hand, and the length-scales that describe the local geometry and motion of the boundary on the other hand. According to these constraints, cell merging will be possible if the temporal refinement over the motion step everywhere adequately resolves the motion on the local grid and if the local spatial grid refinement everywhere in the vicinity of a boundary adequately resolves the local geometry of the boundary. The third constraint ensures that any narrow "necks" in the boundary are adequately resolved by the local grid refinement.

7.11 Computational Cost of Cell Merging

Examination of section 7.7 shows that Step 1 (b) is the the most computationally-intensive part of the merging algorithm. For spline segments with chord-lengths comparable to the local cell dimensions, the complexity of this step is $m \times \log n$, where m is the total number of single spline segments in the entire set of Parametric Composite Cubic Splines present, and n is the total number of cells in the grid. The omitted constant in the above expression depends on the product $a \times b$, where a is the operation count for finding an individual cell in the computational domain that overlaps the bounding rectangle of a given single spline segment, and b is the operation count for computing a spline-rectangle intersection problem (since this computation need only be performed for the cells that overlap the bounding rectangle of the spline). Since this is the most expensive part of the procedure, the intersection

points and the invariant polygons are stored till the end of the merging procedure to eliminate the need for any repeated calculations that may be required during the expansion or remerging of the geometrically-invariant polygons.

7.12 Shortcomings and Limitations of Cell Merging

Despite the enabling role of cell merging for the technique studied in this work, the procedure has several shortcomings. The major ones are as follows:

- the imposition of an additional source of grid-alignment dependence of the solution (especially with highly elongated composite cells);
- the introduction of non-physical propagation of waves and state properties and an increase in the total numerical dissipation (during merging and unmerging);
- the introduction of variations in the areas and area-change ratios of adjacent composite cells, leading to oscillations in the solution around boundaries. These are most noticeable while the near-boundary flow is accelarating rapidly relative to the boundary. Although these are minimized by enforcing tighter upper and lower bounds on the areas and area ratios, it is not possible to completely eliminate these variations around a boundary; and
- the introduction of asymmetry in the grid, and hence in the solution, even for symmetric problems.

The shortcomings described above are, however, confined to the merging envelopes around boundaries, and their spatial extent decreases with increasing grid resolution around boundaries.

7.13 Summary and Concluding Remarks

This chapter defined merging and composite cells, and described the role of cell-merging in enabling the Lagrangian tracking of boundaries and interfaces on a stationary Cartesian grid. The procedure and the requirements for cell merging were described in detail. For stationary boundaries, the cell-merging approach developed in this work can be regarded as a generalization of the Method 2 of Appendix A.2.

Throughout this chapter, it was assumed that only two positions of a boundary (the starting and ending ones) need to be considered across each motion step. This is always the case if a two-step solution procedure is used. However, if there are more than two positions (as in the application of this algorithm to [8, 9]), then the definitions of geometric and topologic invariance must be extended to cover all the positions that occur during the motion step. The actual algortihm checks for an arbitrary number intermediate positions. Note that only the discrete intersection states are relevant for the purposes of merging - whether the continuous boundary motion violates the invariance conditions is irrelevant.

All the discussion in this chapter was strictly in the context of two-dimensional space. Extension of some of the ideas presented here to three-dimensional space is discussed in Chapter IX. Although cell merging is a computationally intensive procedure, the additional expense can be justified in terms of enabling economical and efficient grid generation and adaptation procedures in two-dimensional space. However, because of the increased emphasis on computational efficiency, the Cartesian approach with cell merging will be less attractive in three-dimensional space than it is in two-dimensional space.

CHAPTER VIII

Verification, Validation, and Demonstration

This chapter presents and discusses several of the computations that were performed with the algorithm developed and studied in this work. The two most important considerations in selecting or designing the test cases for the computations were as follows: (i) their suitability for the verification and validation of the overall methodology; and, (ii) the extent to which they enabled the demonstration or the investigation of the important capabilities and features of the methodology, and its strengths and weaknesses. Emphasis is placed throughout on test cases that contain flows with discontinuities or stagnation points, since the correct and accurate treatment of these flow features is usually considered the most critical and demanding criterion for numerical schemes for the solution of The System of Euler Equations, as described in Chapters II and III. The quality of the solutions obtained for the various test cases is assessed, general observations about the performance and the computational properties of the new methodology are given, and conclusions are drawn about its relative capabilities, and about potential improvements to it. The effectiveness of the grid-adaptation scheme at making optimal use of the computational resources is also discussed. The requirements of the new methodology in terms of storage space and computational effort are also discussed, and compared with those of alternative

techniques, such as the ones discussed in Chapter I.

8.1 Organization and Description of the Computational Results

The computations presented in this chapter are separated into three main classes, according to the type of flow problem computed, as follows: (i) computations of steady flows (with stationary boundaries); (ii) computations of unsteady flows with stationary boundaries; and, (iii) computations of unsteady flows with moving boundaries. The third class also includes the sub-class of moving-boundary problems with topologic transformations in the boundaries. Each class has an entire section of this chapter devoted to it, and wherever a sub-class is identified, a separate sub-section of the corresponding section is wholly devoted to the sub-class, and at least one example from that sub-class is shown or discussed in that sub-section. Within each class or sub-class presented in this chapter, the computations are arranged roughly in order of increasing physical complexity, or in order of increasing number and complexity of the algorithmic capabilities utilized.

The classification described above was chosen in preference to a classification of the problems according to whether they primarily serve a verification, validation, or demonstration purpose, because it is more intuitive, and because several of the test cases actually serve dual or multiple roles. The three classes (and their sub-classes) identified in the preceding paragraph cover the entire range of problem types for which the new computational methodology was developed, as described in detail in Section 1.5.

For each test case, the physical problem involved and its important features are described. The corresponding computational results are also presented and de-

scribed, with the emphasis being placed mostly on the correctness of the computational solution, and on the resolution of the geometric and solution features of the problem. Where appropriate, point values at important locations and contour-line plots in important regions of the computational solution are given. The quality of the computational solution and its correspondence to the physical problem involved is assessed as far as possible, and the role served by the test case for any verification, validation, or demonstration purposes is indicated, at least implicitly. For computations that are used for validation or verification testing, the computational results are compared directly with the corresponding analytical or experimental results, and the implications of the comparison are discussed. Otherwise, special flow-features of specific problems are only discussed if their physical significance warrants it.

For each computation, the numerical treatment is described in as much detail as practicable. In particular, the Boundary Conditions and, if appropriate, the Initial Conditions used are usually fully specified. However, since only one type of Boundary Condition is appropriate at impermeable boundaries in inviscid flow, as described in Chapter III, explicit specification of this Boundary Condition is normally omitted. The chief options or parameters chosen for the geometric representation, for the solution algorithm, and for the adaptation procedure are given as appropriate. For example, since the discrete solution sometimes depends strongly on the one-dimensional flux calculation procedure on cell faces (following Godunov's method, as described in detail in Chapter III), the specific numerical flux function(s) used in the computation are usually specified. Similarly, the specific gradient-reconstruction and gradient-limiting procedures chosen from among the possible options are specified. In addition, any special parameter settings or any special procedures or modifications to the standard formulations adopted or used, as described in Chapter III, are given.

Unless otherwise stated, the fluid in all the computations shown in this chapter is assumed to be an ideal, perfect gas with $R = 287.0$ and $\gamma = 1.4$. Any special capabilities or techniques that are not part of the solution algorithm but that were also utilized to perform a computation are described. However, if these capabilities or techniques are beyond the scope of the work, or are not directly relevant to it, they are described only in as much detail as is relevant to the work, or as is necessary for the purposes at hand.

The level of detail to which the various test cases are discussed varies greatly: typically, those test cases which serve a validation and verification purpose are discussed in detail, while those test cases which serve mostly a demonstration purpose are discussed briefly. The organization and the arrangement of the description and discussion also varies from test case to test case, since no single uniform arrangement seems to be simultaneously the most suitable for all the test cases.

For those test cases in which corresponding sets of contour-line and grid plots are given, the corresponding plots are chosen to cover identical spatial regions. The exceptions to this convention are committed, for example, when it is more appropriate to render the finer cells in the grid more visible. In all the contour-line plots presented in this chapter, the contour level values increase in uniform increments between the minimum and maximum values registered over the entire Computational Region, even if the plot shows the contours in only part of the Computational Region.

8.2 The Role of Verification, Validation, and Demonstration

Verification, validation, and demonstration [6] are three related, important processes that are used to establish the fidelity, accuracy, effectiveness, usefulness, efficiency, robustness, and reliability of a new methodology for computational simula-

tion.

Verification is the process of establishing that an algorithm executes in the manner it was intended to and designed for (without regard to the correspondence to physical reality of the consequences of those intentions). Equivalently, verification is the process of ensuring that there are no "unintended errors" in an algorithm. Verification is typically carried out through simple tests on individual components of the algorithm, usually by comparing the outputs obtained to the corresponding known, expected values. Verification tests are rarely discussed or presented since they are usually considered part of the detailed implementation procedure. This convention will be largely followed in this work.

Validation is the process of establishing the extent to which the adopted mathematical model and the adopted numerical solution scheme together generate computational solutions that reproduce the physical phenomena being simulated. Validation is typically carried out through comparison with experimental data or analytical solutions. For The System of Euler Equations, there is a small, but useful body of exact and approximate analytical solutions for classical problems involving smooth flows as well as flows with discontinuities, as described, for example, in Section 2.3 and the references therein (including [5], [7], and especially [75]). These analytical solutions, however, are mostly for flows with simple geometries and simple Boundary and Initial Conditions. There is also a wide variety of standardized, benchmarked experimental test cases for The System of Euler Equations, as described, for example, in [5, 7]. More recently, comparisons with numerical results from validated, benchmarked algorithms have been also been accepted for validation purposes, the process then being specifically called "cross-validation".

For solutions of The System of Euler Equations, an important outcome from a val-

idation program, and a chief requirement for assessing the successfulness or quality of a new scheme, is the determination of how accurately and reliably the algorithm being tested preserves into the discrete solution all the fundamental analytical features and properties that The System of Euler Equations possesses, as given in Chapter II, and especially in Section 2.2. This is particularly the case with properties across discontinuities because of the strong nonlinearities and the far-reaching global effects associated with these features, and in stagnation regions because of the difficulty of obtaining accurate, converged numerical solutions for high-speed flows there. Special features to be examined carefully in validating a computational scheme for The System of Euler Equations include the rate of numerical diffusion of vortex sheets, and the width of shock and contact waves which are oblique to the grid lines.

Demonstration is the process of establishing the types and the ranges of problems to which an algorithm is applicable, and exhibiting the quality of the results that can be obtained. Demonstration is typically carried out through simulation of realistic problems requiring the special features or capabilities that are to be established or exhibited, and is achieved only if accurate or plausible results are obtained.

Because of the crucial importance and the central role of intersected cells and of cell merging in this work, much of the verification and validation efforts must be directed towards establishing that the use of intersected cells and cell merging does not significantly degrade the quality of the solution or the order of accuracy of the solution algorithm, and to quantifying any negative side effects that are caused by use of intersected cells or cell merging. Analogously, much of the demonstration efforts must be directed towards establishing the additional capabilities that become feasible, and the new advantages that can be realized with the use of intersected cells and with the use of cell merging. In addition, part of the demonstration effort should

seek to quantify the additional computational costs involved in using intersected cells and cell merging.

8.3 Steady, Stationary-Boundary Computations

Because the computational methodology developed in this work is of the Arbitrary-Lagrangian-Eulerian type, it is formally applicable to all problems involving steady-state solutions (of inviscid, compressible flows of ideal gases) about stationary boundaries. It should therefore be verified, validated, and demonstrated for such problems. However, any steady-state solution for a flow-field about a rigid obstruction in the Reference Frame of the obstruction is invariant to any Galilean Transformation of the velocities of the far-field and the obstruction. Therefore, any steady-state, stationary-boundary problem can be re-formulated and treated as an unsteady, moving-boundary problem, and may indeed be computationally simulated as such. Nevertheless, it is useful and important to be able to solve steady-state, stationary-boundary problems in their special form (that is, without transforming them to the equivalent unsteady, moving-boundary problems), even when the capabilities of an Arbitrary-Lagrangian-Eulerian solution methodology are available. This is because the special form enables a significant reduction in the required computational resources, and offers the advantages of modeling simplification, ease of visualization of the results, and ease of data extraction for validation purposes. Therefore, it remains necessary and useful to verify, validate, and demonstrate the new methodology for the class of steady-state, stationary-boundary problems in their special form. For this reason, several test cases from this class were studied and explored. The remainder of this section is devoted to presenting two of the most popular of those test cases.

8.3.1 Confined Collision of Two Supersonic Streams

In this popular [277] benchmark test case, a uniform stream with a Mach Number of 2.378 is injected at an angle of -10.940378 degrees to the x-axis, into another uniform stream which is directed along the positive x-axis with a Mach Number of 2.9.

In the specific geometric and modeling arrangement adopted here, the horizontally-directed stream is injected along the axis of a channel of finite length with parallel, horizontal, infinitesimally-thin sides, and the inclined stream is injected into the horizontal stream from the upper side of the channel, as shown in Figures 8.1 and 8.2.

The first interaction between the two streams results in the formation of a single, straight oblique shock wave which discretely separates the two uniform streams, connecting the lower and upper sides of the channel. This oblique shock wave is also reflected from the lower side of the channel towards the upper side of the channel. The flow pattern, including the geometry of the shock-wave reflection pattern, and the locations and spread of the expansion fans are shown in Figures 8.1 and 8.2, which clearly show how the resulting shock system separates three regions of uniform flow.

For the given flow configuration and conditions [431], and for a channel with width 1 unit, the leading shock wave should intersect the lower side of the channel at the downstream distance $x = 1.8040477$, and should re-intersect the upper side (after reflection) at the downstream distance $x = 2.3243076$. The reflected shock angle should be 23.279098 degrees to the x-axis. The stream emerging from the reflected shock should have a Mach Number of 1.940.

As indicated in Figures 8.1 and 8.2, the analytical results described above are reproduced to the third decimal place by the computational solution. The maximum

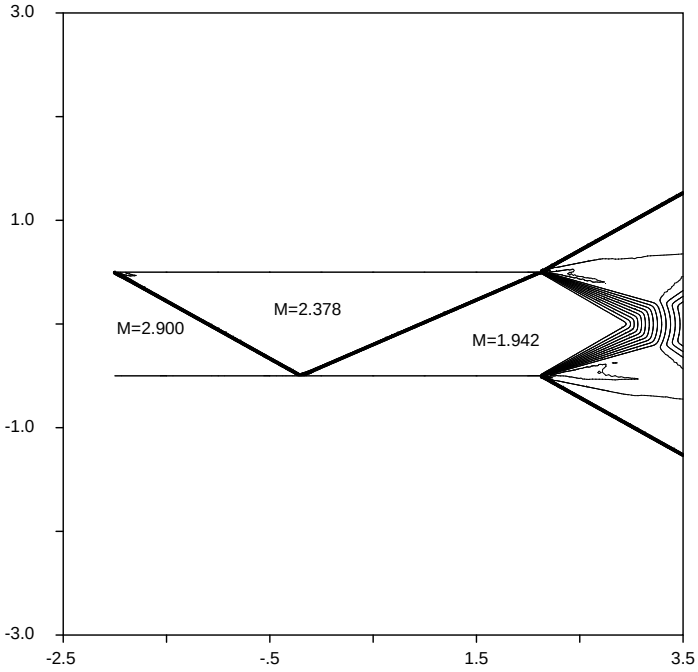

Figure 8.1: Contour-line plot of the Mach Number (with 100 contours), showing the point values in the regions of uniform properties, and the largely monotonic solution variation across the discontinuities.

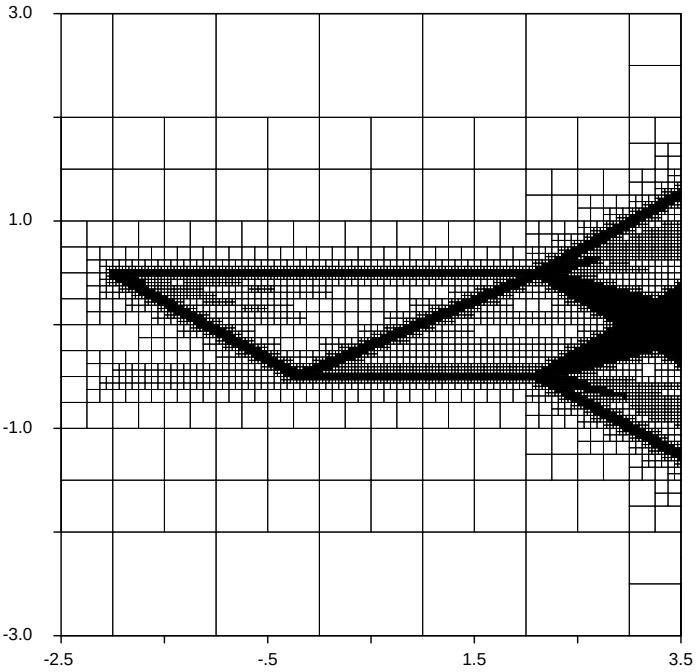

Figure 8.2: Grid plot, showing the solution-induced refinement around the stationary shock waves and in the regions occupied by the expansion fans, and the solution-induced coarsening in the regions of uniform properties.

refinement level was increased from the levels used in displayed solutions, first by 1, and then by 3, and the only observable effect of this increase was to spatially sharpen the shock system, and to make the solution smoother in the expansion fan region. The contour-line plot of Figure 8.1 shows how the regions of uniform flow are monotonically and accurately captured, and how the solution algorithm accurately handles preservation of the jump conditions across an oblique shock, and how it accurately handles the application of the boundary conditions at impermeable boundaries, even in local regions in which there are reflected shock waves (and hence two or three different uniform states).

The computations for this test case were performed with the Exact-Riemann solver, with Least Squares Gradient reconstruction, and with the modified form of the Barth Limiter described in Chapter III. The solution adaptation was based on all four Adaptation Indicators described in Chapter VI, and especially in Section 6.5, even though some of them could have been suppressed.

The effect of solution adaptation in reducing the number of computational cells is made evident in Figure 8.2, which clearly shows that only a small proportion of the total number of cells fall in the uniform flow regions. The total number of cells in Figure 8.2 is about 37,000 cells, and most of those cells fall in the expansion fan region.

A small pocket of non-uniformity (corresponding to a variation of approximately 1.2% in the Mach Number) is observed just behind the leading shock wave near the leading edge of the upper side of the channel, and this is caused by heavy limiting of the reconstructed gradient at the solid boundary at that location.

8.3.2 Subsonic Flow About A Cylinder

In this test case, an impermeable, rigid, circular boundary is immersed in a uniform free-stream of ideal gas having $\gamma = 1.4$. The far-field Mach Number, M_∞, is set to 0.2, with the following specific choice for the free-stream primitive-variable vector: $\{\rho_\infty, u_\infty, v_\infty, p_\infty\} = \{1.0, 74.417569, 0.0, 98,892.406\}$.

No exact analytical solution for the problem of flow of an inviscid compressible fluid over a right cylinder appears to have been derived. Exact solutions in terms of infinite series, however, have been obtained using the Hodograph Method for shapes approximating that of a right cylinder [71]. It would be possible to automate the solution process developed in [71] and demonstrated by an example in [72] to obtain semi-analytical solutions for a right cylinder, but this is beyond the scope of this work. Nevertheless, as described in the next paragraph, even without knowledge of its exact solution, the test problem studied in this sub-section can be effectively used in validating a computational scheme, and in assessing many of its critical and important features.

Because of the relatively low speed of the free-stream in this problem, no flow separation (whether inviscid, or induced by numerical diffusion) is observed. The resulting flow-field is only weakly compressible, to the extent that the flow remains subsonic (and hence smooth) everywhere, and the solutions for the velocity field (and hence also for the streamlines) and for the pressure field closely approximate the corresponding classical solutions for incompressible, inviscid flow of a uniform free-stream about an infinite cylinder [30]. In particular, the maximum possible point differences between the compressible and incompressible variants of this specific problem, which can be closely estimated from the Mach Number [431], must everywhere be less than about 4% for both the pressure and the velocity values, and

in practice are even less than that. The density in the compressible variant of this problem, however, varies by almost 7% from its free-stream value (and hence also by the same amount from the uniform-density value of the incompressible variant). Furthermore, the solution at most of the critical points in the flow-field for the compressible variant can be closely estimated using various analytical and semi-analytical methods [325, 431].

Twenty-four different computations were performed for this test case, using Computational Region dimensions of 16x16, 32x32, and 64x64 units (which here correspond to meters). These three sizes of Computational Region will hereinafter be respectively named S (for small), M (for medium or intermediate), and L (for large). For each size of Computational Region studied, four successive maximum refinement levels were used. In particular, the three sets of four different maximum refinement levels used for the 16x16, 32x32, and 64x64 Computational Region sizes were respectively $\{8, 9, 10, 11\}$, $\{9, 10, 11, 12\}$, and $\{10, 11, 12, 13\}$. For each combination of Computational Region size and maximum refinement level value, two different computations were performed: one using a local time-stepping procedure, and one using a time-accurate, global time-stepping procedure in which the solution was evolved to the relatively large integration time of 5s. In all of the computations, the circle was centered at the origin, and was assigned a radius of 1 unit (which here corresponds to 1 meter). Also in all of the computations, the far-field flow was directed purely along the positive x-axis, as implied in the choice of the free-stream primitive-variable vector given above.

The numerical differences (of either 1 or 2) between corresponding elements of the three sets of maximum refinement level given in the preceding paragraph are such that for all three sizes of Computational Region, the same ratio is maintained

of the width of the finest-cell to the radius of the circle with each group of three corresponding elements. In particular, the four maximum refinement levels in each set and its associated Computational Region size correspond to the following four ratios of the width of the finest-cell to the radius of the circle: 0.125000, 0.062500, 0.031250, and 0.015625.

Figures 8.3 and 8.4 below show the geometry involved in this test case, specifically for the Computational Region with size M, and show the cell size distribution for the third highest maximum refinement level (namely, 11) used for this size. The coarsest cells in Figure 8.3 have a refinement level of 6, while the finest cells have a refinement level of 11. However, due to the limits of plotting resolution in the figure, the cells at refinement levels 10 and 11 are indistinguishable in that figure, but are clearly distinguishable in Figure 8.4. Figure 8.4 also clearly shows how the cell intersection at the circular boundary results in a very large variation in the areas of the affected cells.

Because of the relationship explained above between the different maximum refinement levels for the different sizes of Computational Region, the cell density, and even the specific cell cutting pattern around the circular boundary depends only on the sequential order of the maximum refinement level (which ranges from 1 to 4, as indicated above). In particular, the number of cells cut by the boundary of the circle ranges from about 60 (for the lowest maximum refinement level used), to about 500 (for the highest maximum refinement level used), and these numbers are the same for all three sizes of Computational Region. Thus, the zoomed grid plot of Figure 8.4 has the same appearance for all three sizes of Computational Region used when each size is meshed with its third highest maximum refinement level.

No solution-adaptation was used for this test case, in order to enable a more

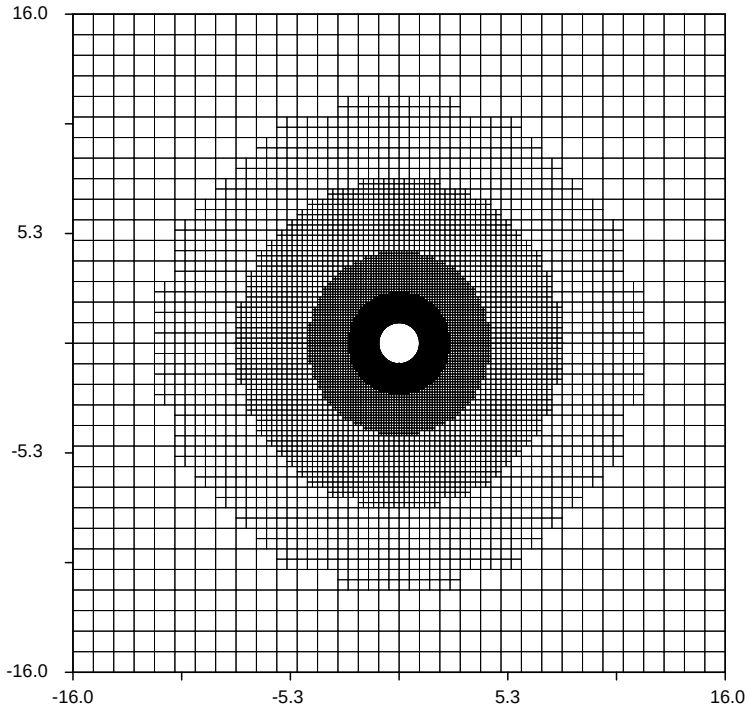

Figure 8.3: Grid plot showing the geometry and the location of the Computational Region of size M and of the cylinder, and showing the cell size distribution across the former with its third highest maximum refinement level (11).

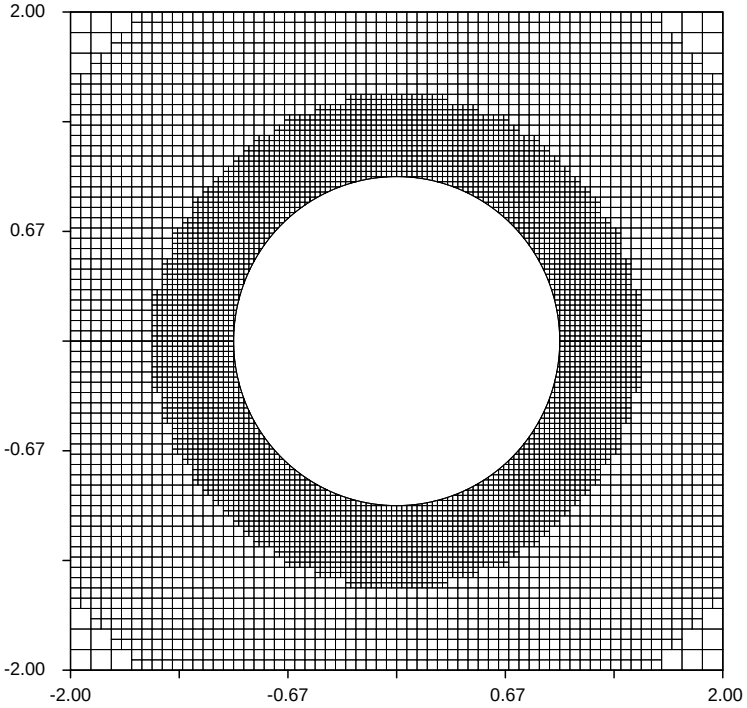

Figure 8.4: Zoomed grid plot showing the cell size distribution around and in the near neighborhood of the cylinder with the third highest spatial resolution.

easily and precisely controlled grid-refinement study to be performed, and in order to avoid any disturbances to the high degree of smoothness of the solution. Such small disturbances invariably accompany refinement and coarsening operations, and are normally inconsequential, especially in time-accurate calculations with discontinuities in the solution. Instead of using solution adaptation, the cells were refined according to their radial proximity from the surface of the cylinder, to form predetermined concentric annular bands of uniformly-refined cells, as shown in Figures 8.3 and 8.4 below. Another, far more attractive, possibility would have been to use error estimates from the exact solution for the corresponding incompressible flow to form an adaptation criterion to be used to create a fixed adaptation pattern.

All twenty-four calculations were performed using the Exact Riemann Solver for computing the fluxes, using the Least-Squares algorithm for computing the 1-exact gradient reconstruction, and using no limiting of the reconstructed gradients. The limiting was not necessary because there are no discontinuities in the solution and therefore no risk of generating non-physical states or oscillations in the solution. Elimination of the limiting also eliminates artificial clipping of the extrema, and leads to the most accurate solution possible with the given spatial discretization.

Because of the bluntness of the boundary geometry in this test case, consideration must be given to the enforcement of the Kutta Condition, as described in Chapter II. Indeed, if the exact circular geometry described above is used as it is, the trailing-edge stagnation point and (to a lesser extent) the leading-edge stagnation point are both observed to slowly drift and oscillate along the boundary of the cylinder about their theoretical locations of respectively $(1, 0)$ and $(-1, 0)$ (with a magnitude of up to several computational cells in extent for the higher maximum refinement levels), causing the drag and lift forces on the cylinder to also oscillate.

This occurs whether local or global time-stepping is used, but in the case of local time-stepping, this oscillation also completely stalls the convergence. Another observation associated with the Kutta Condition was the increasing total integration time required with increasing maximum refinement level for the solution to converge to its "time-average". This is probably because of the increasing difficulty in establishing the appropriate Kutta Condition at the trailing edge of the cylinder with decreasing numerical dissipation.

The effects and difficulties described in the preceding paragraph were largely eliminated by attaching to the cylinder a small spike at the trailing-edge. The spike was directed along the positive x-axis, and was given a length of 0.1 units and a thickness of 1.0e-08 units. This modification effectively enforces the Kutta Condition by moving the trailing-edge to a sharp geometric feature, and forcing the flow to leave the tip of the spike tangentially to the x-axis, in accordance with the expected role played by such a sharp feature, and in accordance with the phenomenon of inviscid separation, as described in Chapter II. As would be expected, another implied benefit of adding the spike was improved symmetry about the x-axis of the computational solutions. It was also found that increasing the length of the spike from that described above, or adding a second spike at the leading-edge of the cylinder were both found to produce little additional improvement in the convergence rate, the extent of symmetry of the solution about the x-axis, or the overall quality of the solution. All results presented and discussed in this sub-section were obtained with this spike added to the original geometry of the exact circle, instead of with the exact cylinder described above. However, the thickness of the spike, as given above, is relatively so small that the spike can only be seen in grid or contour-line plots with considerable magnification, and is therefore not visible in the plots shown

in this sub-section.

Comparing solutions on grids having equal maximum spatial resolution shows that the computed pressure and velocity fields vary by less than about 5% everywhere if the Computational Region size is varied from S to M. The corresponding variation in the pressure and velocity solutions drops to less than about 1% everywhere if the Computational Region size is varied from M to L. This pattern of decreasing differences indicates that the free-stream boundary conditions can be considered to be applied sufficiently far away from the circular obstacle in the Computational Regions with sizes M and L, and that computations performed with both of these sizes of Computational Region should be considered sufficiently accurate, at least with regard to the correct imposition of the far-field boundary conditions.

Comparing solutions obtained on identical grids (and hence also on identical Computational Regions) shows that the computed pressure and velocity fields vary by less than about 1% everywhere when the time-integration scheme is varied from local time-stepping to global time-stepping with a total integration time of 2s. After a total integration time of 5s for the global time-stepping scheme, the corresponding differences between the solutions obtained with the two time-integration schemes drop to less than about 0.2% everywhere. Furthermore, the differences are smaller for the smaller Computational Region sizes, and larger for the larger Computational Region sizes, presumably because it takes increasingly longer for the "inflow" to convect any perturbations or deviations from the infinite-time solution out of the Computational Region as the size of the latter increases.

Because of the relative smallness of the differences described in the preceding two paragraphs, and in order to present the results and the important conclusions compactly, it was chosen to confine all plots of computed contour and surface data

presented in this sub-section to those taken from the time-accurate calculations performed on the Computational Region with size M. Any significant deviations observed relative to these baseline results for calculations performed with local time-stepping, or with the other two sizes of Computational Region used are described only verbally.

Figures 8.5, 8.6, 8.7, and 8.8 respectively show the contour-line plots for the distributions of the density, the pressure-coefficient, the total-velocity, $q = \sqrt{u^2 + v^2}$, and the entropy increase around the cylinder for the Computational Region with size 32x32 units (size M). The first two of these figures are for solutions with the third highest refinement level for this size of Computational Region; namely, 11, while the last two are for solutions with the highest maximum refinement level for this size of Computational Region; namely, 12. The reason for displaying plots with different maximum refinement levels is to illustrate the key features of the computational solution with the minimal number of plots and space, as will become evident through the succeeding several paragraphs.

The contour-line plots for the computed solutions of the density, the pressure-coefficient, and the total velocity are all smooth throughout the flow-field, and exhibit the correct overall patterns and behavior throughout, including the critical points. As would be expected for a correct compressible-flow solution with a free-stream Mach Number $M_\infty = 0.2$, the distributions for the total velocity and the pressure coefficient are close [431, 325] to the corresponding ones for the incompressible-flow solution.

The quantitative behavior at the critical points, such as the leading-edge and trailing-edge stagnation points, and the top and bottom maximum-velocity points, agrees closely with the expected behavior. For example, as shown in Figures 8.6 and

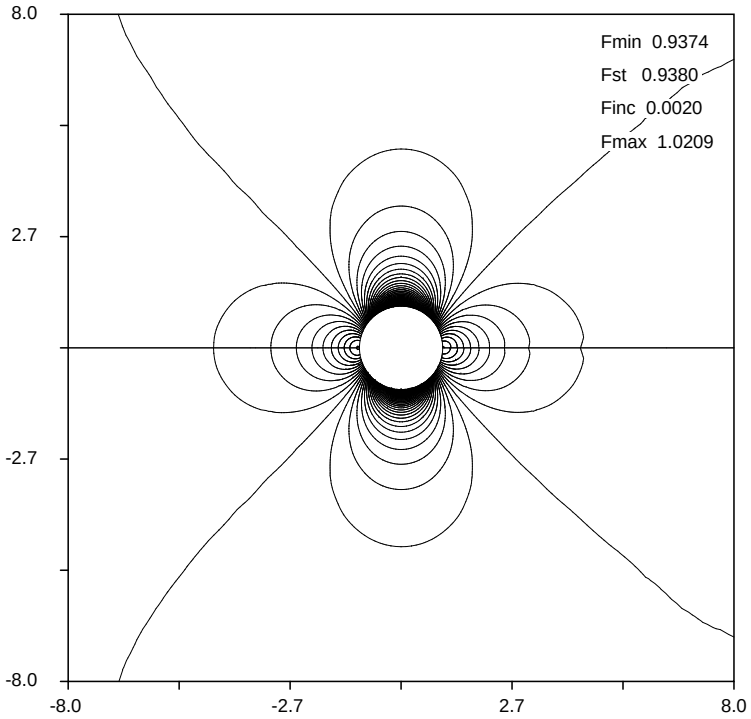

Figure 8.5: Contour-line plot of the density distribution around a cylinder in compressible flow with $M_\infty = 0.2$.

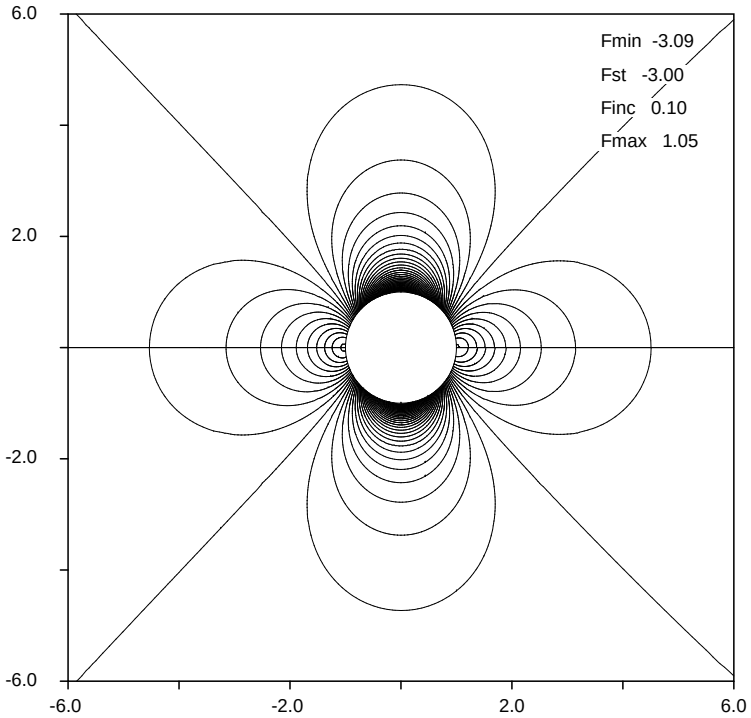

Figure 8.6: Contour-line plot of the distribution of the pressure coefficient around a cylinder in compressible flow with $M_\infty = 0.2$.

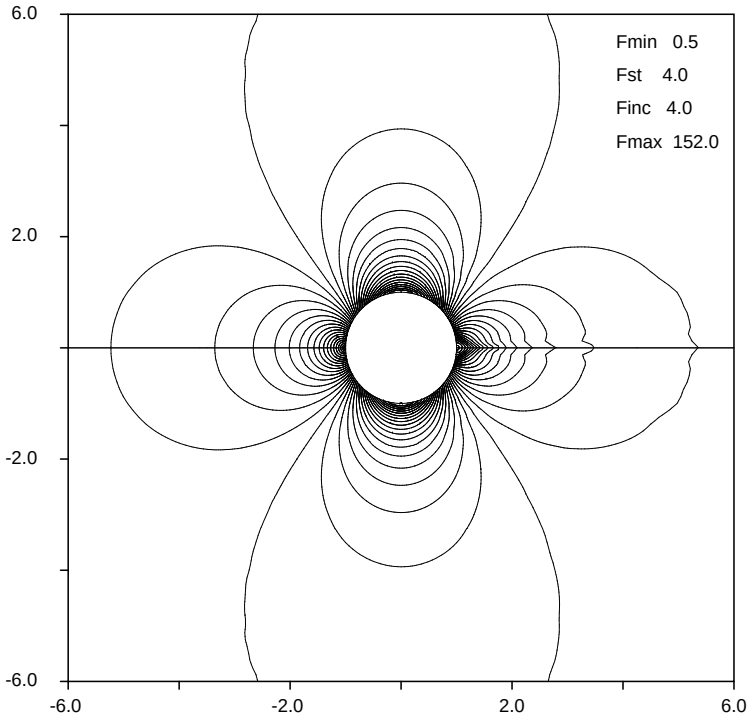

Figure 8.7: Contour-line plot of the total-velocity distribution around a cylinder in compressible with $M_\infty = 0.2$.

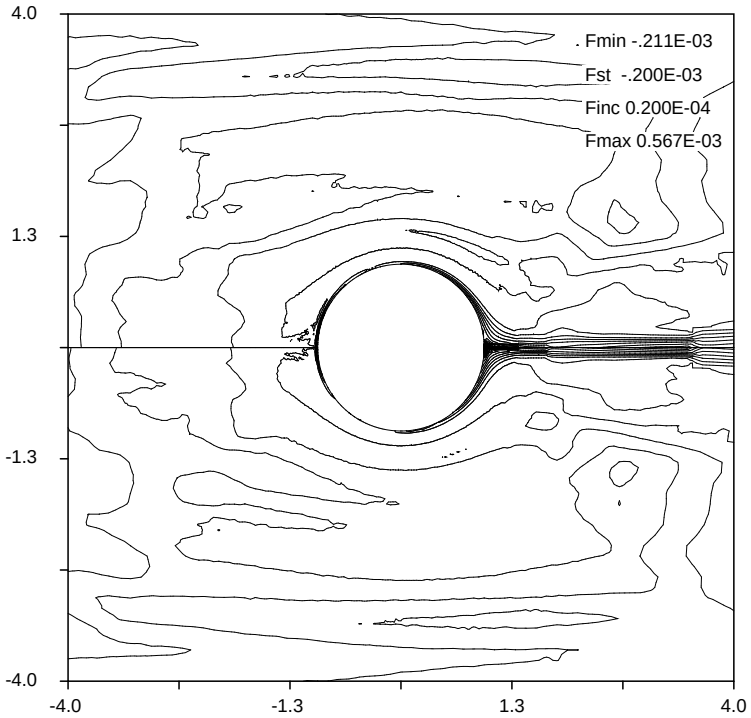

Figure 8.8: Contour-line plot of the entropy distribution around a cylinder in compressible flow with $M_\infty = 0.2$.

8.7, the total velocity approaches zero quite closely at the leading-edge and trailing-edge stagnation points, and the minimum pressures and the maximum total velocities occur very close to the $\theta = \frac{\pi}{2}$ and $\theta = \frac{3\pi}{2}$ points, where θ is the angle along the arc of the circle, measured in an anti-clockwise convention from the trailing-edge stagnation point. The behavior of the computational solution at the stagnation points and at the maximum-velocity points is discussed in more detail below.

Close examination of all four contour-line plots in Figures 8.5, 8.6, 8.7, and 8.8, however, shows that while the computational solution exhibits a relatively high degree of symmetry of about the x-axis, it exhibits an appreciable degree of asymmetry about the y-axis. This asymmetry can be seen most clearly by comparing the shapes of the outermost contour lines and the locations of their intersection points with the square, symmetrically-centered boundaries of the plotted region. For example, in Figure 8.5, the outermost contour lines to the left of the y-axis bend away from the x-axis and intersect the top and bottom edges of the boundary of the plotting region, while the outermost contours to the right of the y-axis bend toward the x-axis and intersect the left edge of the boundary of the plotting region.

The reasons for the symmetry about the $x$axis are the physical symmetry of the problem, the correctness and consistence of the application of the far-field and impermeable-interface boundary conditions, and the correctness of the overall numerical scheme, at least for subsonic flow.

The reason for the asymmetry about the y-axis is the loss in stagnation pressure and the generation of entropy due to numerical dissipation in the computational scheme, especially in the vicinity of the cylinder, and especially in the intersected cells on the boundary of the cylinder. This increase in entropy is clearly shown in Figure 8.8. The numerical dissipation causes the temperature and the pressure

to respectively increase and decrease relative to the isentropic solution as the flow rounds the cylinder, causing the trailing-edge stagnation point to attain a lower pressure coefficient and a lower density that the corresponding properties at the leading-edge stagnation point. Without this effect, and without any asymmetries introduced by cell-merging or by geometric misalignments, the solution should be highly symmetrical about the y-axis. As would be expected with any consistent computational scheme, the degree of symmetry is found to increase with increasing maximum refinement level throughout the range of 4 different maximum refinement levels examined in this test case, except for an anomalous disturbance introduced by cell merging, as explained further below. Indeed, the distortions described above in the outermost contour lines in Figure 8.5 are appreciably lower for the solution employing the highest maximum refinement level, 12, for this Computational Region size. Conversely, these distortions are more severe for the two lower maximum refinement levels. This improvement in the extent of symmetry about the y-axis with increasing maximum refinement level is shown and discussed in more detail below, especially in plots of the distributions of properties on the surface of the cylinder.

The asymmetry about the y-axis is most noticeable for the density, total-velocity, and entropy-increase plots, and least noticeable for the plot of the pressure-coefficient. Thus, for example, the bending and asymmetric intersection of the outermost contour lines occurs to a much smaller extent in Figure 8.6 than in Figure 8.5. For the density and total velocity plots, this greater proportional sensitivity to losses in stagnation pressure or increases in entropy is because of the relatively low free-stream Mach Number, $M_\infty = 0.2$. For the entropy increase plot, this asymmetry is a natural outcome of the cumulative build-up of any increases in entropy (or any losses in stagnation pressure) along streamlines, as described in more detail below and in

Chapter II.

In addition to the pronounced asymmetry about the y-axis, the density and the total-velocity plots also clearly exhibit the effects of the loss of stagnation pressure through the appearance of "wiggles" in the contour lines downstream of the trailing-edge stagnation point, indicating fluctuations in the local solution in those regions. In the pressure-coefficient plot, these wiggles are hardly discernible. In the density plot, these wiggles cause a localized and relatively minor distortion in the affected contour lines. In the total-velocity plot, these wiggles distort the contour lines for several radial lengths downstream of the trailing-edge stagnation point, and to such an extent that the affected contour lines suffer a loss in their circular shape in some regions. It should be noted that this occurs even for the computation with the highest maximum refinement level, as shown in Figure 8.7. Nevertheless, it should also be noted that both the magnitude of the oscillation in the wiggles, and the area affected by the wiggles decrease with increasing maximum refinement level. This behavior suggests that ideally, cells near and downstream of the trailing edge should be more refined than elsewhere around the boundary of the cylinder.

The manner in which the entropy increases along a streamline lying close to the boundary of the cylinder is clearly shown in Figure 8.8. In particular, the figure shows how a relatively large localized increase in entropy along such a streamline first occurs around the leading-edge stagnation point. This rapid initial increase is followed by little further increase until after the flow passes the maximum-velocity points and starts to decelerate towards the trailing-edge stagnation point. During this deceleration phase, in which the flow is traveling in an adverse pressure gradient, the accumulating increase in entropy is relatively very rapid. Finally, another abrupt increase in the entropy occurs as the flow passes and turns through the trailing-edge

stagnation point. The entropy increase along the surface of the cylinder is highly "noisy", reflecting the apparently random actions from several potential sources, some of which are described below.

A major contributor to the generation of entropy in the vicinity of the cylinder is the reduction in the local order of accuracy caused by the application of boundary conditions (instead of an interior scheme), as explained in more detail in Chapter III. The effects on the entropy generation arising from the presence of arbitrarily-intersected cells in itself is usually minor, especially (as is the case in this test problem) if the solution is smooth, and if the second-order reconstruction is not modified by gradient limiting.

Another major contributor to the generation of entropy in the vicinity of the cylinder is the cell-merging-and-unmerging procedure, which repeatedly applies spatial averaging to the solution data as that data is cyclically re-distributed between computational groups and computational cells, as explained in detail in Chapter VII.

The two major sources of entropy generation described above both decrease in intensity with decreasing cell size, at least with a first-order dependence, as described in Chapter III, or as described in more detail below.

In order to quantify the effect of merging and unmerging on the local and global accuracy of the solution procedure, a series of simple experiments were carried out. In all the computations presented for this test case, the merging and unmerging cycles were invoked for each time-step. However, because this test case neither involves moving boundaries nor involves solution adaptation, the cell merging can be done only once before the first time-step, and the computational groups formed from this merging operation can be retained for all subsequent time-steps in the computation. Doing this effectively means that the computation would be performed on an

invariant grid (instead of on a grid which changes at least near the surface of the cylinder after every time-step) which has some elongated composite cells in the immediate vicinity of the boundary, as described in detail and shown in several examples in Chapter VII. Doing this also has the consequence that any additional diffusive effects from the merging and unmerging operations is eliminated. The simple experiments mentioned above involved repeating the calculations for the Computational Region with size M with a single merging cycle instead of with cyclic merging and unmerging.

The two main findings from the experiments described in the preceding paragraph were as follows: (i) the entropy generation around the cylinder decreased by about 75-25% for the computations with non-cyclic merging compared to the computations with cyclic merging; and, (ii) the decrease became smaller as the maximum refinement level increased, varying from about 75% for maximum refinement level 9, to about 25% for maximum refinement level 12. The two conclusions drawn from these findings are as follows: (i) the repeated merging and unmerging contributes substantially to the entropy generation, at a level comparable with the contribution from the application of the boundary conditions; and, (ii) this contribution also decreases with increasing spatial resolution, leading at least to a consistent numerical scheme. The latter behavior is as expected since the as the cell size decreases, the spatial regions over which the averaging is applied from merging and unmerging become smaller. Another reason is probably because, as shown in Figure 8.8, as the maximum refinement level increases, the entropy generation away from the cylinder becomes increasingly comparable with the entropy generation near the cylinder, decreasing the relative importance of any effects from merging or unmerging. This effect, however, may be peculiar to the cell-size distribution arbitrarily imposed in

this problem.

An important characteristic exhibited by all four of the contour plots of Figures 8.5, 8.6, 8.7, and 8.8 is related to the behavior of the solution at the surface of the cylinder and in its immediate vicinity: in all cases, the contour lines approach and meet the surface of the cylinder smoothly, without abrupt or rapid changes in direction, implying the following have been achieved in the solution algorithm: (i) correct treatment and application of the boundary conditions for impermeable interfaces; (ii) correct treatment of arbitrarily-intersected cells; and, (iii) correct and consistent treatment of the cell merging and unmerging procedures. The behavior of the contour lines also indicates that cell merging does not discernibly affect the quality of the local solution. The last observation is discussed further below, as part of the discussion of line plots along the surface of the cylinder.

Another important characteristic exhibited by all four of the contour-line plots of Figures 8.5, 8.6, 8.7, and 8.8 is the smoothness of the contour lines across local jumps in the refinement level. This smoothness indicates that at least first-order accuracy is achieved in the solution even across refinement boundaries. However, as can be seen in the outermost contour lines in Figure 8.5, this smoothness across refinement boundaries deteriorates with increasing cell size.

The contour-line plots for the Mach Number and for other derived quantities show similar behavior patterns and similar departures from the ideal isentropic behavior to those exhibited by the density and the total velocity plots of Figures 8.5 and 8.7 respectively (as opposed to those exhibited by the pressure or pressure coefficient plots, which show little apparent effect from the departure from isentropic behavior, as shown, for example, in Figure 8.6).

Figures 8.9, 8.10, 8.11, and 8.12 show respectively the distributions of the density,

pressure, total velocity, and Mach Number along the boundary (or surface) of the circle. The abscissa in these figures represents the angular measure in radians along the boundary (which here also coincides with the arc length along the boundary), using an anti-clockwise convention, starting from the trailing-edge stagnation point (that is, from the point $(0, 1)$). For the reasons explained above, all the data presented in these figures were taken from the time-accurate computations performed with the Computational Region of size M (after an integration period of 5s). Each figure shows the results for the three highest maximum refinement levels used in the computations. The results for the lowest maximum refinement level were omitted to avoid cluttering the plots, especially that the trends and the patterns of variation of the overall results with grid refinement are adequately described by the data for the three highest refinement levels, and especially that much of the attention will be focused on the differences between the solutions obtained with the highest two maximum refinement levels. In addition, the second and third of these figures show the distributions for the exact analytical solution for the corresponding incompressible, inviscid problem.

All four of the Figures 8.9, 8.10, 8.11, and 8.12 uniformly exhibit four important characteristics.

The first of these characteristics is that the overall solution along the surface of the cylinder is smooth, even with the lowest of the three maximum refinement levels represented in the Figures. This implies that the treatment of intersected cells in the flow solver is correct and accurate, and also implies that the cell-merging and unmerging operations have no discernible adverse effects on the smoothness or the quality of the solution in the vicinity of a boundary.

The second of these characteristics is that the overall symmetry about the y-axis of the solution uniformly improves with increasing maximum refinement level,

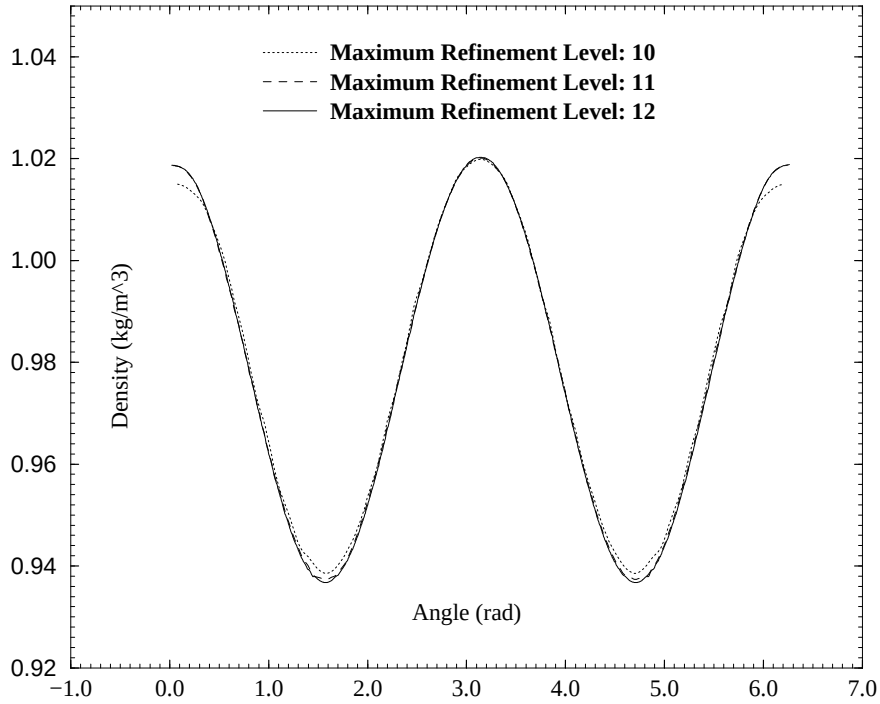

Figure 8.9: Distributions of the density along the surface of a cylinder at $M_\infty = 0.2$ for the highest three values of maximum refinement level used.

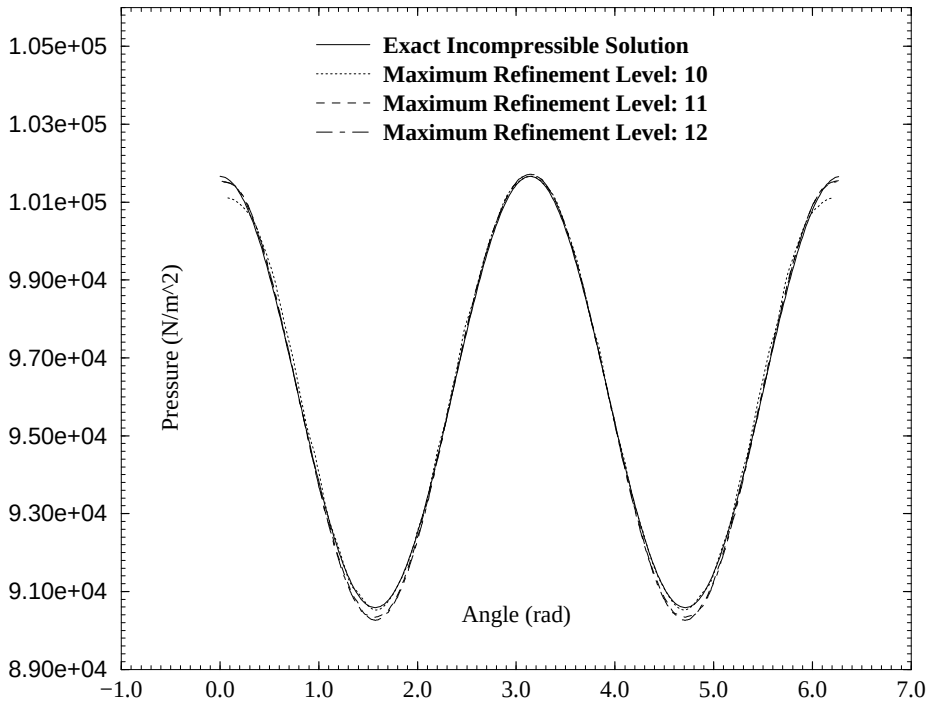

Figure 8.10: Distributions of the static pressure along the surface of a cylinder at $M_\infty = 0.2$ for the highest three values of maximum refinement level used, and for the exact solution for incompressible flow.

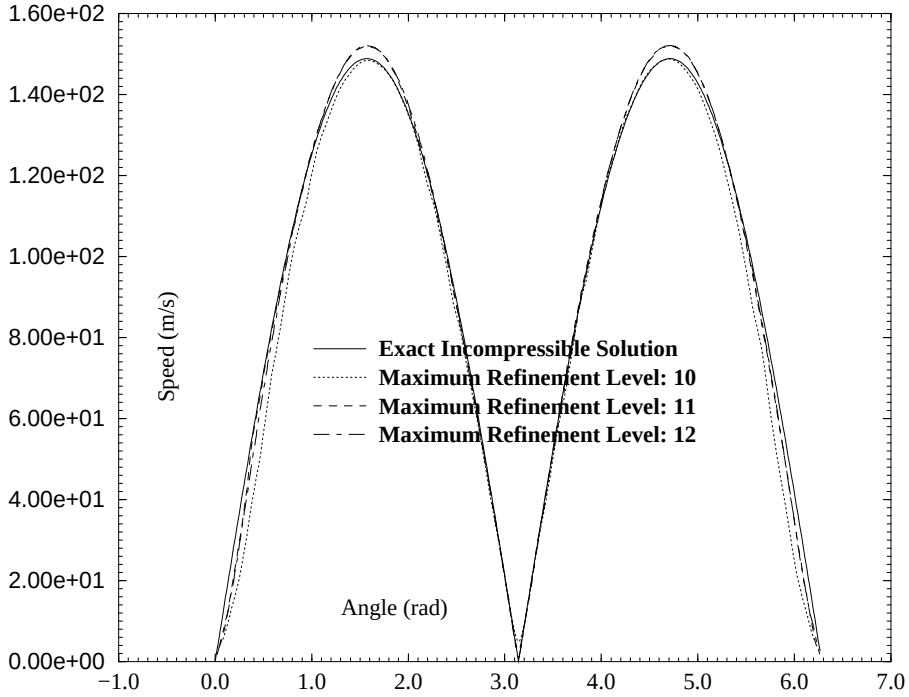

Figure 8.11: Distributions of the total velocity along the surface of the cylinder at $M_\infty = 0.2$ for the highest three values of maximum refinement level used, and for the exact solution for incompressible flow.

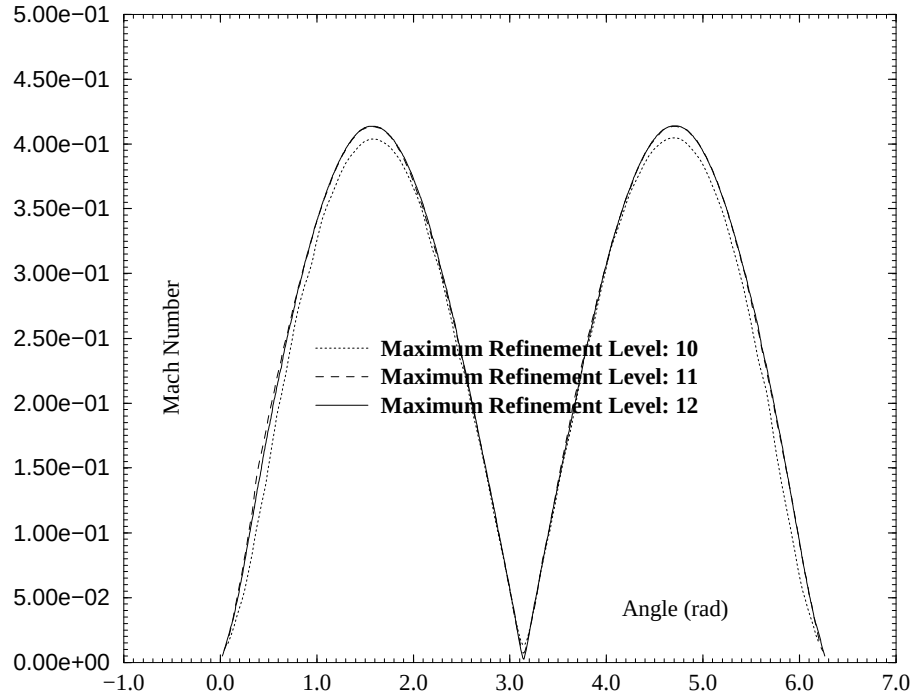

Figure 8.12: Distributions of the Mach Number along the surface of a cylinder at $M_\infty = 0.2$ for the highest three maximum refinement levels used.

as observed in the discussion of the contour plots given above. This implies that the solution algorithm is consistent, and that the solution error (especially from numerical dissipation) decreases with increasing spatial resolution.

The third of these characteristics is that the differences between the solution obtained with maximum refinement levels 11 and 12 are relatively very small. This implies that a so-called grid-converged solution was achieved for this problem, and that further increases in maximum refinement level will produce small and rapidly diminishing improvements in the accuracy of the solution.

The fourth of these characteristics is that the solution becomes progressively closer with increasing maximum refinement level to the correct solution, as measured by point values at critical points. This again implies that the solution algorithm is consistent and accurate, and that the boundary conditions, the numerical flux functions, and reconstruction are all correctly implemented and accurate, at least for subsonic and smooth flows.

The improved matching with increasing maximum refinement level between the computed solutions and the exact values at critical points can be observed in detail in each of the four Figures 8.9, 8.10, 8.11, and 8.12. For example, in Figure 8.9, as the maximum refinement level increases, the maximum density increases towards its exact correct value of about 1.020 (as determined from the stagnation state of the free-stream, as explained below), while the minimum density decreases towards its exact correct value of about 0.937 (as can be deduced from analysis of the isentropic expansion of the free-stream towards the maximum-velocity state, as explained below). In addition, the locations of the maximum and the minimum values becomes increasingly more accurate as the maximum refinement level is increased. Specifically, the points at which the Mach Number and the total velocity should attain their

maxima are the uppermost and lowermost points along the boundary of the cylinder, while the points at which these properties should attain their minima are the leading and trailing edges of the cylinder. The locations of the maxima and minima for the pressure and the density occur at the same four points just mentioned, but in the reverse assignment to that just given. Similar patterns of improvement with increasing maximum refinement level in the accuracy of the values of the maxima and minima, and in the accuracy of the locations of these maxima and minima are also observed in each of Figures 8.10, 8.11, and 8.12, as can be seen by inspection of these figures, and as described in more detail below.

In both Figures 8.10 and 8.11, the computed solution obtained with a maximum refinement level of 10 appears to fall closer to the exact solution for the corresponding incompressible flow than the computed solutions obtained with the other two (higher) maximum refinement levels. However, this is only a coincidence, since the compressible variant of the problem generates lower minimum pressures (which are also called suction pressures), and higher maximum total velocities than than the incompressible variant [431, 325], just as shown in Figures 8.10 and 8.11, and the solutions with refinement levels 11 and 12 are in fact the more accurate ones, as described in more detail below. Both of these two figures also exhibit a characteristic discriminant that readily differentiates the compressible and incompressible variants of this problem [325]: Figure 8.11 shows that the total velocity for the incompressible variant increases initially more rapidly than that for the compressible variant with increasing distance from the stagnation points, but then later starts to increase more slowly and ultimately attains a lower peak value than the total velocity for the compressible variant. Similarly, Figure 8.10 correctly exhibits the converse behavior (for pressure) to that just described for the total velocity, but to a relatively smaller

extent.

Figure 8.10 shows that while the accuracy of the computed value for the pressure at the trailing-edge stagnation point improves appreciably with increasing maximum refinement level, the accuracy of the computed value for the pressure at the leading-edge stagnation point is relatively high with all three maximum refinement levels, and therefore exhibits comparatively little improvement with increasing maximum refinement level. This difference in behavior is attributable to the accumulation of entropy generation along the boundary of the circle, which is clearly shown in Figure 8.8 and discussed above. This accumulating entropy generation in turn causes the corresponding accumulated loss in stagnation pressure to be carried to the trailing-edge stagnation point. In contrast, the loss in stagnation pressure suffered by the streamline(s) terminating in the leading-edge stagnation region is relatively smaller, and does not depend so much on global effects like the integrals of entropy generation across large portions of the Computational Region. The differences in predicted values for the two stagnation points are further discussed below.

Figure 8.12 is the surface-distribution plot which shows most clearly a peculiar feature of the computations with maximum refinement level 11: namely, a relatively higher degree of asymmetry about the x-axis that with the three other maximum refinement levels. It turns out that this is caused by differences in the cell-merging patterns about the cylinder with different maximum refinement levels. For the computation with a maximum refinement level of 11, the merging pattern produces an asymmetric composite-cell pattern (which is effectively equivalent to an asymmetric grid) which affects the symmetry of the solution. It should be noted, however, that as with all other effects of the cell merging and unmerging, any asymmetry produced in this manner, even if it persisted, or if it periodically disappeared and re-appeared

with increasing maximum refinement level, should also progressively diminish with increasing maximum refinement level.

Figures 8.13 and 8.14 show zoomed plots of the Mach Number along the surface of the cylinder, respectively around the upper maximum-velocity point, and around the leading-edge stagnation point. The two plots again display the four important characteristics listed and discussed above for the four Figures 8.9, 8.10, 8.11, and 8.12; namely: (i) the smoothness of the solutions (of a derived quantity, and hence also of the primitive quantities) along the surface of the cylinder despite the presence of intersected cells and the effects of cell merging and unmerging; (ii) the improvement in the symmetry of the solution about the y-axis with increasing maximum refinement level; (iii) the attainment of a "grid converged" solution with increasing maximum refinement level in general, and the relatively negligible difference between the solutions with maximum refinement levels 11 and 12 in particular; and, (iv) the increasing accuracy with increasing maximum refinement level in the prediction of the point values at critical points, as discussed in more detail below.

Compared to the four Figures 8.9, 8.10, 8.11, and 8.12, however, the two Figures 8.13 and 8.14, because of their magnification factors, show more clearly the presence in the solution of the asymmetries discussed above. In particular, even at the highest maximum refinement level used (12), Figure 8.13 shows the persistence of a slight asymmetry about the y-axis, while Figure 8.14 shows the persistence of a (relatively more slight) asymmetry about the x-axis. These asymmetries persist even with the introduction of the small spike at the trailing-edge of the cylinder, as described and discussed above, and for the reasons described and discussed above (which are mainly related to the asymmetry in the cell-merging pattern). Also, the two Figures 8.13 and 8.14 show more clearly than the Figures 8.9, 8.10, 8.11, and 8.12 the improvement in

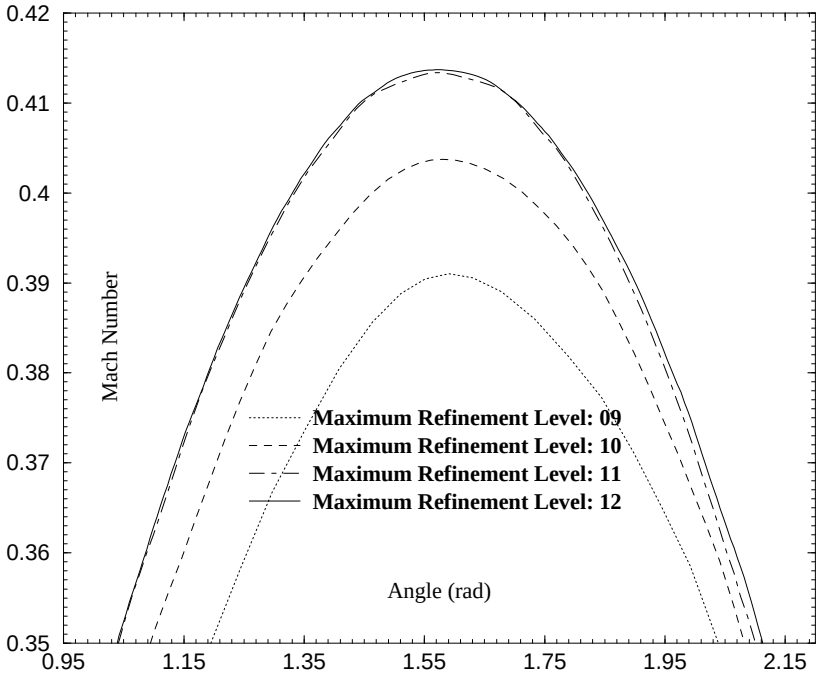

Figure 8.13: Distributions of the Mach Number in the vicinity of the upper maximum-speed region on the surface of a cylinder at $M_\infty = 0.2$ for the four different maximum refinement levels used.

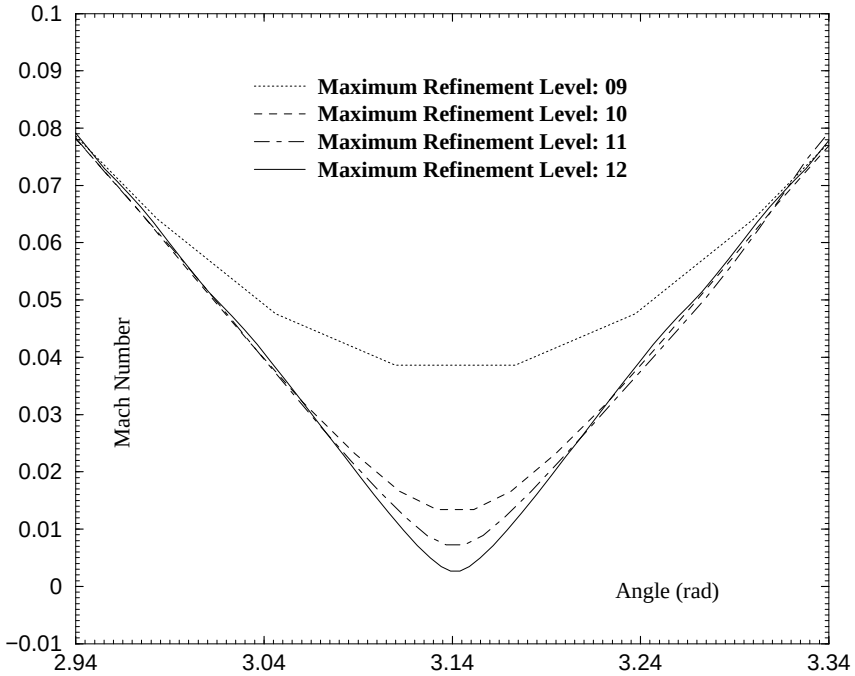

Figure 8.14: Distributions of the Mach Number on the surface of a cylinder in the vicinity of the leading-edge stagnation point at $M_\infty = 0.2$ for the four different maximum refinement levels used.

accuracy at the leading-edge stagnation point, and at the maximum-velocity point. Specifically, Figures 8.13 and 8.14 respectively show that the highest and lowest Mach Numbers with the highest maximum refinement level used (12) are respectively about 0.414 and 0.0027.

The computed point values at several critical points in the flow-field, and the manner in which these computed point values vary with maximum refinement level are shown in Table 8.1. The table also shows the corresponding exact values, computed from isentropic, compressible flow theory. The table also shows one integrated computed quantity: the total drag force, F_D, on the cylinder. Specifically, the table shows the values of the static pressure and static density at the leading-edge and trailing-edge, the values of the maximum and minimum total velocity and the maximum and minimum Mach Number in the Computational Region, the drag force on the cylinder, and the maximum increase in entropy (above its free-stream value). In Table 8.1, the maximum refinement level is expressed in terms of the ratio $h_{min} : R$, where h_{min} is the width of the finest cell in the Computational Region, and R is the radius of the cylinder, which here takes the value 1. The "division" symbol for properties whose value is given at both the leading-edge and trailing-edge stagnation points is used as a device to present the data more compactly, and does not signify arithmetic division. The subscripts LE and TE respectively signify the conditions at the leading-edge and the trailing-edge of the cylinder, which should theoretically correspond to stagnation states. The subscripts max and min have their usual meanings. The entry $\rightarrow 0$ signifies the solution that would theoretically be obtained in the limit $h_{min} \rightarrow 0$, but is here used to identify the exact, analytical values computed from isentropic, compressible flow theory. All other symbols used in Table 8.1 have their usual meanings.

$h_{min} : R$	$\frac{p_{LE}}{p_{TE}}$	$\frac{\rho_{LE}}{\rho_{TE}}$	$\frac{q_{max}}{q_{min}}$	$\frac{M_{max}}{M_{min}}$	F_D	s_{max}
0.125000	$\frac{101623}{100700}$	$\frac{1.01930}{1.0100}$	$\frac{144.10}{2.30}$	$\frac{0.3925}{0.0060}$	313.94	0.00370
0.062500	$\frac{101685}{101100}$	$\frac{1.01994}{1.0150}$	$\frac{148.76}{1.31}$	$\frac{0.4048}{0.0035}$	73.410	0.00150
0.031250	$\frac{101789}{101500}$	$\frac{1.02088}{1.0189}$	$\frac{151.71}{1.05}$	$\frac{0.4137}{0.0030}$	21.12	0.00056
0.015625	$\frac{101717}{101550}$	$\frac{1.02027}{1.0191}$	$\frac{152.02}{0.49}$	$\frac{0.4139}{0.0013}$	7.313	0.00048
$\rightarrow 0$	$\frac{101689.19}{101689.19}$	$\frac{1.0201202}{1.0201202}$	$\frac{152.65}{0.00}$	$\frac{0.41563}{0.0000}$	0.000	0.00000

Table 8.1: The computed and exact point values of various properties at critical points in the flow-field around a cylinder in an $M_\infty = 0.2$ free-stream.

As mentioned above, the exact point values at the critical points of the flow-field for this test problem were obtained from compressible, isentropic flow theory, as explained in, for example, [325]. In particular, for $\gamma = 1.4$, as is the case in all the computations performed here, the maximum speed, q_{max} or u_{max}, is given in terms of the inflow speed, u_∞, by the approximation [325]

$$u_{max}/u_\infty \approx 2.00 + 1.167M_\infty^2 + 2.58M_\infty^4 + 7.53M_\infty^6 + \cdots$$

For $M_\infty = 0.2$, the above expression evaluates to 2.0513, to give in this case a maximum speed of 152.652 m/s. The corresponding (maximum) Mach Number for this speed under isentropic expansion conditions is 0.415626. The exact values for the density or any other property are obtained in a similar manner, that is, employing isentropic compression or expansion relations.

The computational point values obtained with the highest maximum refinement level (12) that correspond to the exact point values described at the end of the

preceding paragraph were respectively 152.0m/s and 4.140, as shown variously in Figures 8.7, 8.11, 8.12, and 8.13. The observed level of agreement between the computed and exact values is quite acceptable considering the coarseness of the grid, even at the highest maximum refinement level of 12. Another indicator of the resolution and the quality of the solution is implied in the value of the Mach Number at the leading-edge stagnation point, which falls below about 3.0e-03 for the highest maximum refinement level, as shown in Figure 8.14.

Table 8.2 shows the error in the computed solutions, relative to the analytically-obtained exact values. The entries is Table 8.2 are wholly derived from those of Table 8.1, and therefore provide no new data. However, Table 8.2 allows both the accuracy and the order of the accuracy of the computational solution and scheme to be more easily or conveniently visualized, as discussed in more detail below.

$h_{min} : R$	$\frac{p_{LE}}{p_{TE}}$	$\frac{\rho_{LE}}{\rho_{TE}}$	$\frac{q_{max}}{q_{min}}$	$\frac{M_{max}}{M_{min}}$	F_D	s_{min}
0.125000	$\frac{66.2}{989}$	$\frac{0.0008202}{0.0101202}$	$\frac{8.55}{2.30}$	$\frac{0.02313}{0.0060}$	313.94	0.00370
0.062500	$\frac{4.19}{589}$	$\frac{0.0001802}{0.0051202}$	$\frac{3.89}{1.31}$	$\frac{0.01083}{0.0035}$	73.410	0.00150
0.031250	$\frac{-100}{189}$	$\frac{-0.0007598}{0.0012202}$	$\frac{0.94}{1.05}$	$\frac{0.00193}{0.0030}$	21.12	0.00056
0.015625	$\frac{-27.8}{139}$	$\frac{-0.0001498}{0.0010202}$	$\frac{0.63}{0.49}$	$\frac{0.00173}{0.0013}$	7.313	0.00048

Table 8.2: The errors in the computed point values of various properties at critical points in the flow-field around a cylinder in an $M_\infty = 0.2$ free-stream.

Several observations can be made about the entries in Tables 8.1 and 8.2, most of them merely confirming or repeating observations already made above for the contour and surface plots.

The first observation is that the accuracy of the computed point values tends to improve with increasing maximum refinement level, just it should for any consistent computational scheme. This behavior can be taken as evidence of the convergence of the computational solution to the correct physical solution. There are a few exceptions to this improvement in accuracy, however. In particular, the absolute errors in the density and the pressure predictions at the leading-edge seem to increase in going from maximum refinement level 10 to maximum refinement level 11 before dropping again on going from maximum refinement level 11 to maximum refinement level 12. Also, the density and pressure predictions at the leading-edge stagnation point unexpectedly overshoot the corresponding values of exact solution. It should be noted, however, that those particular predictions are highly accurate, even with the coarsest grids used. In such situations, achieving indefinite improvements in accuracy by increasing grid refinement is usual not possible, for reasons that are discussed above and below.

The second observation is that there appears to be a clear pattern of diminishing returns and a "slowing down" in the improvement in accuracy for the majority of point values shown in Tables 8.1 and 8.2 on going from maximum refinement level 11 to maximum refinement level 12. The most plausible reason for this is the probable rise to relative significance of errors from sources other than the truncation, such as projection errors, and finite-arithmetic errors. This is somewhat supported by the observations made above regarding the entropy increase in the computational solution away from the surface of the cylinder for the computation with the highest maximum refinement level used.

The third observation regards the order of accuracy of the computational scheme. As implied in the discussions of solution error given in Chapters IV and VI, in the

limit of grid refinement, for a computational scheme which is first-order accurate, whenever the cell width (or length-scale) decreases by a factor of 2, the error should decrease by a factor of 2, while for a scheme which is second-order accurate, whenever the cell width decreases by a factor of 2, the error should decrease by a factor of 4. Based on this, the entries of Table 8.2 imply that on the whole, the predictions for pressure, density, total-velocity, and Mach Number on the boundary fall short of second-order accuracy, but are close to first-order accuracy (especially if the unexplained anomalies described above at the leading stagnation point, in which the order of accuracy appears to fall short of even 1, are ignored). On the other hand, the predictions for the drag force appear to achieve second-order accuracy. It should be noted that the major difference between the drag force and the other point values given in Tables 8.1 and 8.2 is that the former is an integral quantity that is strongly influenced by the global behavior of the solution. The implication on the order of accuracy of the computational scheme of these behavior patterns and their differences is discussed below.

Figure 8.15 graphically displays the behavior of the drag force discussed above, clearly showing how the drag force diminishes with increasing maximum refinement level using a logarithmic scale.

The closeness of the computed drag-force data shown in Figure 8.15 to a linear best fit with slope $m = 2$, graphically shows how closely second-order accuracy is achieved for the drag force. Nevertheless, the slow-down discussed in the preceding few paragraphs for the improvement in accuracy between the solutions with maximum refinement level 11 and 12, can again be observed in the figure, although to far smaller extent than with the point values on the boundary of the cylinder.

The overall behavior pattern reported and discussed above for the errors is usu-

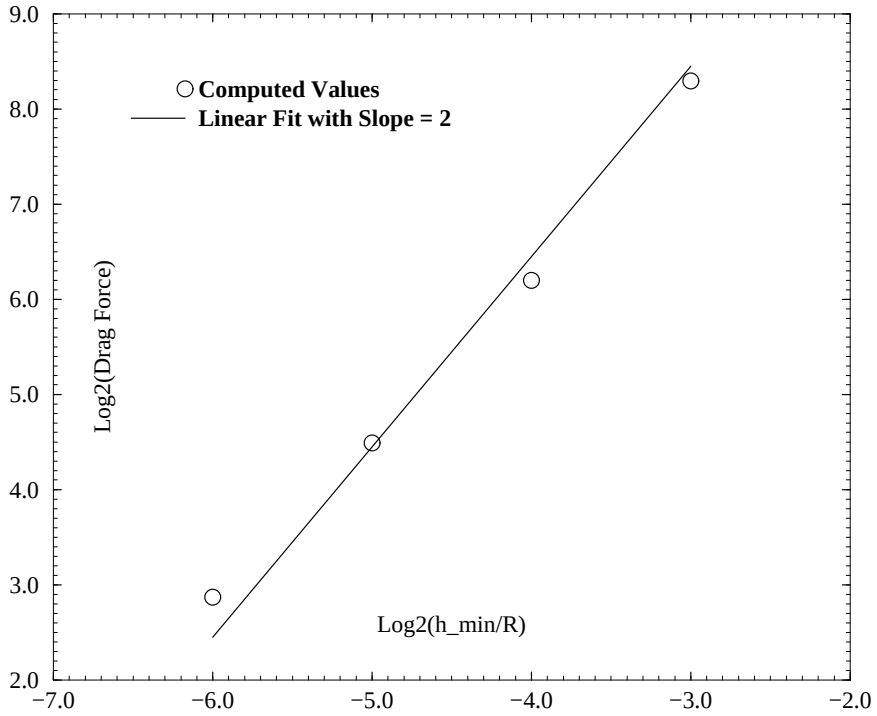

Figure 8.15: The variation (for a cylinder in compressible flow at $M_\infty = 0.2$) of the computed total drag force with the maximum refinement level used in the computation, together with the best linear fit to the computed drag-force data.

ally taken to imply that the computational scheme is locally first-order-accurate on the boundary, and globally second-order-accurate. This is consistent with the expectations for this scheme given the spatial discretization adopted in it, as explained in Chapter III.

8.4 Unsteady, Stationary-Boundary Computations

This section presents and discusses several computations for transient flows about rigid stationary objects. Most of the test cases involved were chosen to serve one or more validation or demonstration purposes, either because the computational results obtained for them may be compared directly with the corresponding analytical or experimental results or with the corresponding computational results from other com-

putational algorithms, or because the test cases involve complex flows with several types of interacting discontinuities that place stringent demands on the adaptation and solution algorithms used.

The computations presented in this section clearly demonstrate the ability of the solution-adaptation algorithm adopted and developed in this work to precisely track traveling discontinuities and maintain them within appropriate refinement zones. They also demonstrate the value of a solution-adaptive computational technique in greatly reducing the number of computational cells and the computational resources required to obtain a resolved computation for a flow with discontinuities or other features involving widely differing length scales.

8.4.1 A One-Dimensional Riemann Initial-Value Problem

A definition of the Riemann Initial-Value Problem (which is also often called the Shock-Tube Problem) is given in Section 3.2. The latter section also describes (as part of the description of the Exact-Riemann numerical flux function implemented in this work) the fundamental solution for the Riemann Initial-Value Problem, especially in terms of a self-similar system of propagating waves. The Riemann Initial-Value Problem and its significance in gasdynamics is also generally discussed in Chapter II.

Variants of the Riemann Initial-Value Problem provide highly appealing test cases for verifying, validating, analyzing, and evaluating the capabilities and characteristics of solution algorithms for compressible, transient flows. This is partly because the exact time-dependent analytical solution for such a problem can often be determined, and partly because such a problem is highly likely to clearly display any errors or weaknesses present in the one-dimensional version of a numerical flux func-

tion, by clearly revealing the correctness and accuracy with which the flux function treats the three fundamental types of discontinuity in the The System of Euler Equations. These discontinuities and their properties are discussed in detail in Chapter II, especially in Section 2.2.

From the infinite number of possible variants of the Riemann Initial-Value Problem, a few have gained popularity as standard or reference benchmark test cases.

The specific variant of the Riemann Problem tested in this sub-section is known most accurately as "Sod's First Problem", and most commonly, as "Sod's Problem". For convenience, the latter name is the one used here. Sod's Problem [337] is fully defined by the following non-dimensionalized initial conditions [337]: $\vec{U}_l = \{\rho_l, u_l, p_l\} = \{1, 0, 1\}$, and $\vec{U}_r = \{\rho_r, u_r, p_r\} = \{0.125, 0, 0.1\}$, where $\vec{U}_l$ applies throughout $x_l < 0$, and $\vec{U}_r$ applies throughout $x_r >= 0$, where the subscripts l and r respectively denote the Left and Right states or regions, and where all other symbols have their usual denotations. In the work presented in this sub-section, Sod's Problem is computed using dimensional (SI) units, with the left state arbitrarily set to $\vec{U}_l = \{\rho_l, u_l, p_l\} = \{1, 0, 100000\}$, leaving the right state to take the value $\vec{U}_r = \{\rho_r, u_r, p_r\} = \{0.125, 0, 10000\}$. For convenience, the time-integration was carried out to a final time of approximately $0.0135s$, corresponding to a traversal distance of the single shock wave which is generated in the solution of approximately $7.49m$.

The two other highly popular variants of the Riemann Initial-Value Problem are the so-called Lax Problem [214], and the so-called Woodward-and-Collela Problem [84, 414]. The Lax Problem is defined by the following non-dimensionalized initial conditions [214]: $U_l = \{0.445, 0.698, 3.538\}$, and $U_r = \{0.5, 0, 0.571\}$, with $x_l < 0$, and $x_r >= 0$, where all symbols denote the same meanings given to them above for

Sod's Problem. The Woodward-and-Collela Problem [84, 414], has the three initial states $U_l = \{1, 0, 1000\}$, $U_m = \{1, 0, 0.01\}$, and $U_r = \{1, 0, 100\}$, where at time $t = 0$, U_l applies in the interval $0 \leq x_l < 0.1$, U_m applies in the interval $0.1 \leq x_m < 0.9$, and U_r applies in the interval $0.9 \leq x_r \leq 1.0$, and where the subscript m denotes the "middle" state. Unlike the situation for Sod's Problem and for the Lax Problem, the Woodward-and-Collela Problem has a finite domain, with boundary conditions of impermeability applied at $x = 0$ and $x = 1$, causing wave reflections to be generated in the solution.

The Lax Problem and the Woodward-and-Collela Problem are more challenging and difficult, and place greater demands on a solution algorithm than Sod's Problem, largely because their solutions involve the creation of new extrema, and the generation or evolution of intermediate states. However, Sod's Problem is chosen here because it serves the purposes of the sought validation more clearly and directly than do the other two.

The criteria that are examined most closely in evaluating the computational solution for a variant of the Riemann Initial-Value Problem include the following: (i) the correctness and accuracy in capturing the physical behavior in the uniform and smooth regions, especially the accuracy in the prediction of the velocity, density, and pressure profiles in expansion fans; (ii) the correctness in predicting the speeds of contact or shock wave discontinuities, and in satisfying the governing jump conditions across these discontinuities (as these jump conditions are described in detail Chapter II); and, (iii) the suitability of the spatial resolution of the computational scheme, typically expressed in terms of the number of cells or points required to resolve a given, large fraction, typically, say 95%, of the jump across a discontinuity.

For multi-dimensional computational schemes, such as the one used in this work,

variants of the Riemann Initial-Value Problem are also used to test and to reveal the effects on the flux functions and the overall numerical scheme of the alignment in space of the nodes or cell faces, and the effects of numerically representing the multi-dimensional wave-propagation problem existing in physical reality through the sum of its projections onto planar waves that are forcibly aligned with cell faces. Thus, variants of the Riemann Initial-Value Problem can be used to quantify the effects of the inherent one-dimensionality of the traditional Godunov-type of computational schemes for multi-dimensional computations.

In addition to its use for all the specific purposes described in the preceding two paragraphs, Sod's Problem is here also used to verify and validate the formulation and implementation of the flow-solution algorithm, including the time-integration scheme, and to perform a study of the effects on the quality of the solution of refinement of the computational cells and the time-step. This is done by comparing the computational results from different computations with each other and with the corresponding exact analytical solution. Finally, Sod's Problem is also used here to compare the accuracy and computational effort requirements of the three main numerical flux functions used in this work: (i) the Roe Flux-Difference Splitting; (ii) the Exact-Riemann Solver; and, (iii) combinations of either (i) or (ii) with the HLLE numerical flux function, with the latter activated only in regions of strong shocks, following the procedure explained in Chapter III, and especially in Section 3.2.

In the series of tests carried out with it, Sod's Problem was computed using eight different configurations of geometry and meshing.

The first four of these, hereafter labelled a(i), a(ii), a(iii), and a(iv), are depicted in Figure 8.16. As the latter figure shows, these configurations involve the entire two-dimensional grid, and no solid boundaries are present. As the figure also shows,

the four configurations a(i) - a(iv) differ only in the initial orientation of an infinite plane passing through the point $(0, 0)$ and separating the left and right states. As indicated in Figure 8.16, these orientations are as follows: in a(i): the normal of the plane is parallel to the x-axis; in a(ii): the normal of the plane lies at $\frac{\pi}{4}$ radians to the x-axis; in a(iii): the normal of the plane lies at $\frac{\pi}{2}$ radians to the x-axis; and, in a(iv): the normal of the plane lies at $\frac{4\pi}{3}$ radians to the x-axis.

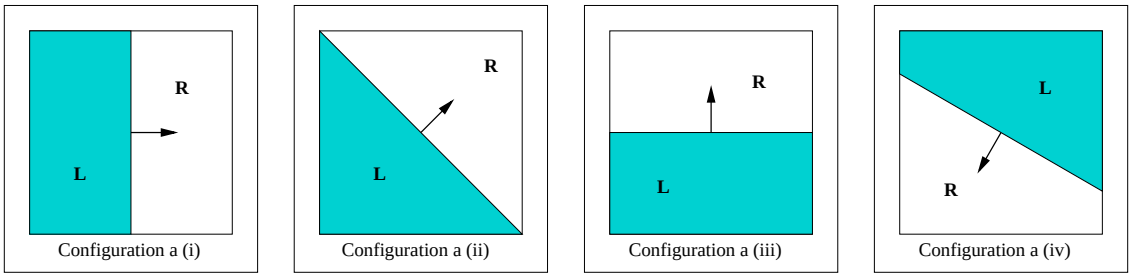

Figure 8.16: The four different configurations of the geometry and initial conditions tested for Sod's Problem using unconfined ("whole-grid") computations.

The second four configurations, hereinafter labelled b(i), b(ii), b(iii), and b(iv), are depicted in Figure 8.17. As the latter figure shows, these configurations confine the "one-dimensional" flow to occur primarily along the axis of an elongated solid object which represents a channel closed at one end and open at the other. For the chosen total evolution time of $t_f = 0.0135s$, none of the waves in the problem reaches the end of the channel, so having one end of the channel closed and the other open has no effect on the computation and is an arbitrary choice. In analogy with the case for configurations a(i) - a(iv), the four configurations b(i) - b(iv) differ only in the initial orientation of the plane separating the left and right states, which also identifies the orientation of the axis of the channel, as shown in Figure 8.17. The orientation angles for configurations b(i) - b(iv) are respectively identical to the

orientation angles for configurations a(i) - a(iv) as given above.

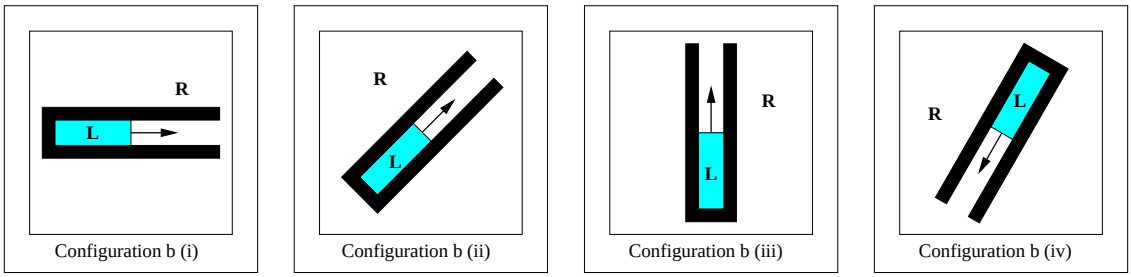

Figure 8.17: The four different configurations of the geometry and initial conditions tested for Sod's Problem using channel-confined computations.

Each of the eight configurations depicted jointly in Figures 8.16 and 8.17 is computed with 6 different levels of maximum refinement; namely: 9, 10, 11, 12, 13, and 14, and with the three different numerical flux functions listed above (the Roe Flux-Difference Splitting, the Exact-Riemann Solver, and the mixed Exact-Riemann and HLLE flux functions, with the use of the latter confined exclusively within and around the shock wave, as explained in more detail in Chapter III). Thus, a total of 144 different computations are performed. For the computations of configurations b(i) - b(iv), the refinement level around the solid boundary, the so-called intersection-refinement-level, is varied with the maximum refinement level, from a value of 7 (when the maximum refinement level is 9), to a value of 9 (when the maximum refinement level is 14). This variation is made to enable the merging algorithm to run at high efficiency, and has no discernible effect on the resolution or the solution quality. This is because the resolution and the quality of the solution, as shown below in this sub-section, are almost wholly determined by the refinement level chosen for the solution-adaptation, because the latter refinement level in all the calculations of this sub-section is higher (by at least two) than the intersection-refinement-level.

In all the computations performed for Sod's Problem, the magnitude of the time-steps was determined automatically using a constant CFL Number of 0.8. Since Sod's Problem is an unsteady one, the second-order, time-accurate, global-time-stepping procedure was chosen for the time-integration. As already mentioned, the final integration time for all the computations, t_f, is, with minor variations from one computation to another, approximately 0.01350s. The minor variations in the total integration times of individual computations are negligible for the purposes of the study in this sub-section, being typically on the order of 1.0e-05s. All computations were performed with second-order spatial accuracy, using the Least-Squares Gradient Reconstruction procedure and the modified form of the Barth Limiter developed in this work, as described in Chapter III.

As explained above, a total of 144 computation is carried out for Sod's Problem. Most of these computations merely serve to confirm or validate a basic requirement of the solution algorithm, and exhibit results that are very similar to those from other computations with similar or identical refinement levels. In order to present the data and findings in the most compact and effective manner, the figures and plots will be confined to data from only 10 different computations. The data and results for the remaining computations will be described verbally by describing how those results agree with or differ from the chosen 10 computations. The reference baseline case is chosen to be the one performed with configuration b(i), with a refinement level of 12, using the Exact-Riemann numerical flux function.

The following observations summarize some of the key findings and identify the computations that are described only verbally (relative to other computations):

- No significant differences in the computational results or the relevant grid adaptation patterns are observed between the computations performed with the

configurations "a" and the computations performed with the corresponding configurations "b". The two main differences between computations with configurations "a" and computations with configuration "b" are not related to the quality or accuracy of the results and are as follows: (i) the computations for configurations a(ii) and a(iv) exhibit small disturbances at the boundary which propagate a small distance into the computational region. These disturbances are caused by the inability to apply the exact boundary conditions needed for discontinuities that are not perpendicular to the outer boundaries of the Computational Region, and are not relevant for the purposes of this sub-section; and, (ii) the computations performed with configurations "a" require far more cells and longer computation times than the ones performed with configurations "b". Because of this situation, only data from configurations "b" will be presented in this sub-section.

Another important motive for choosing to present the "bounded" flows of configurations "b" is that they include and enable the presentation of the effects of cell-cutting (and hence the effects of large variations in cell area), the effects of merging and unmerging of cells, and the effects of the application of the "impermeable solid" boundary conditions at surfaces of rigid boundaries. Investigating such effects is very important and relevant to this work because the cutting and merging and unmerging of cells are characterizing and critical features of the technique developed in this work.

- No significant differences are found between the results obtained with the three numerical flux functions listed above. This not only cross-confirms the validity of all the three flux functions, but also shows that the HLLE flux function when

used exclusively in the region of the shock wave has no observable adverse effect on the solution. This lack of any observable differences is unexpected. Differences between the results obtained with the Roe Flux-Difference Splitting and with the Exact-Riemann numerical flux functions for Sod's Problem become observable only with coarser grids than the ones included in this sub-section, and only in the resolution of the expansion fan. The differences would probably be more significant if there were sonic points outside shock-wave regions, but Sod's Problem does not incorporate such points.

In regard to computational effort requirements, the Exact-Riemann Solver is found to require an average of 10-20% more effort for every flux computation than the Roe Flux-Difference-Splitting numerical flux function. This apparently small difference is attributable to the relatively small number of iterations required to obtain the interfacial primitive state (as described in detail in Section 3.2) for most of the flux computations invoked for Sod's Problem, especially as a result of the choice of configuration. Therefore, this observation should not be extended to the general case, where the differences in the computational effort requirements of the two numerical flux functions could be much higher. No differences are observed as a result of using the HLLE numerical flux function in the shock-wave region, and this is because this flux function is then used in only a relatively very small number of flux evaluations.

- No significant differences are found between the results for configuration a(i) and those for configuration a(iii), or between the results for configuration b(i) and those for configuration b(iii). In other words, the results for the "0-degree" and "90-degree" cases are identical for all practical purposes. Therefore, no

data is shown from any of the "90-degree" computations.

- Only a slight dependence of the results on the orientation angle is observed. Therefore, only a representative sample of such differences is presented and discussed here. This weak dependence is an important property to demonstrate in this work, especially because of the dominant geometric alignment pattern of the Sub-Region Set in any Cartesian grid.

The data and descriptions presented in the remainder of this sub-section together with the verbal summary of observations given in the above list provide a complete picture of the results of the study carried out for Sod's Problem.

Even after reducing the number of specific computations chosen for presentation to only 10, it is not desirable to show all the data for all the 10 cases, especially that some of the graphs would become too cluttered. Therefore, in some instances, only the cases showing the extremes of behavior are shown. Nevertheless, all salient points and all anomalies observed in the computational solutions are mentioned or described.

Figures 8.18 and 8.19 respectively show the density contours and the adapted grid for a computation with configuration b(i). Figures 8.20 and 8.21 respectively show the density contours and the adapted grid for a computation with configuration b(ii). Figures 8.22 and 8.23 respectively show the density contours and the adapted grid for a computation with configuration b(iv). All three of these sets of figures are taken at the end of the chosen time-integration period, at approximately $t_f = 0.0135s$. All three of the corresponding computations are carried out with a maximum refinement level of 12, and are performed using the Exact-Riemann numerical flux function. Also, all three of the density contour plots mentioned above have between 86 and 90

contour levels.

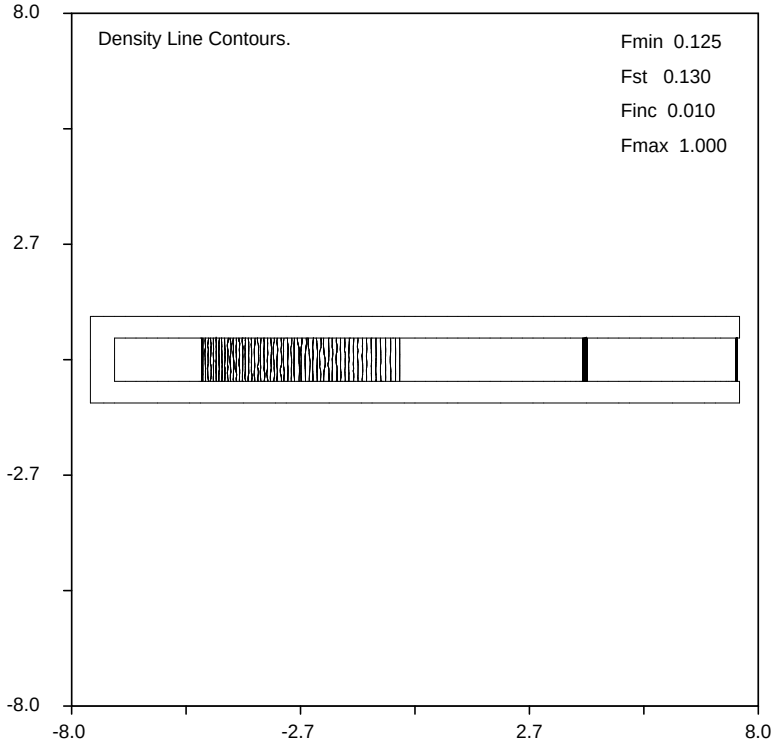

Figure 8.18: The density contours for Sod's Problem with configuration b(i) at $t = t_f$.

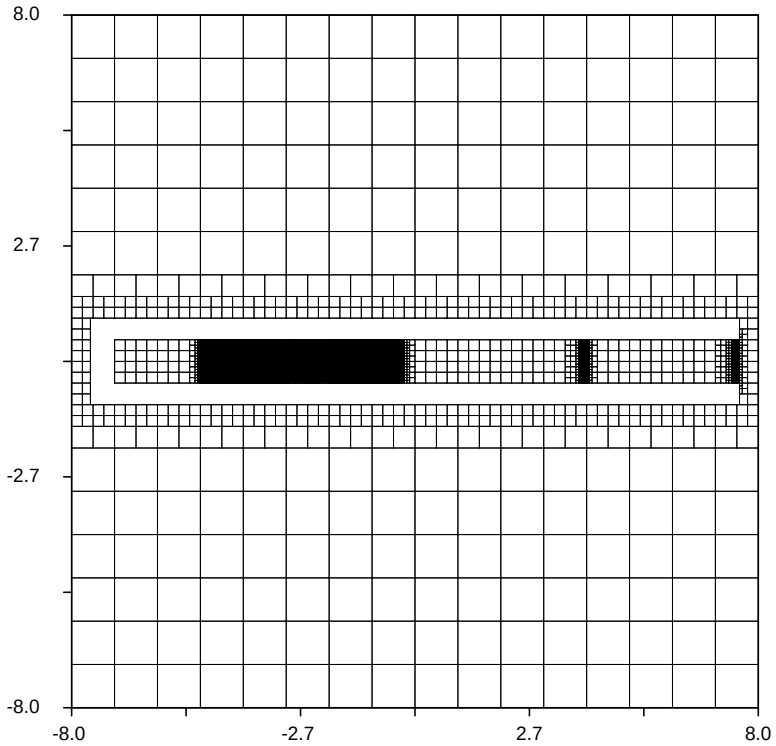

Figure 8.19: The grid plot for Sod's Problem with configuration b(i) at $t = t_f$.

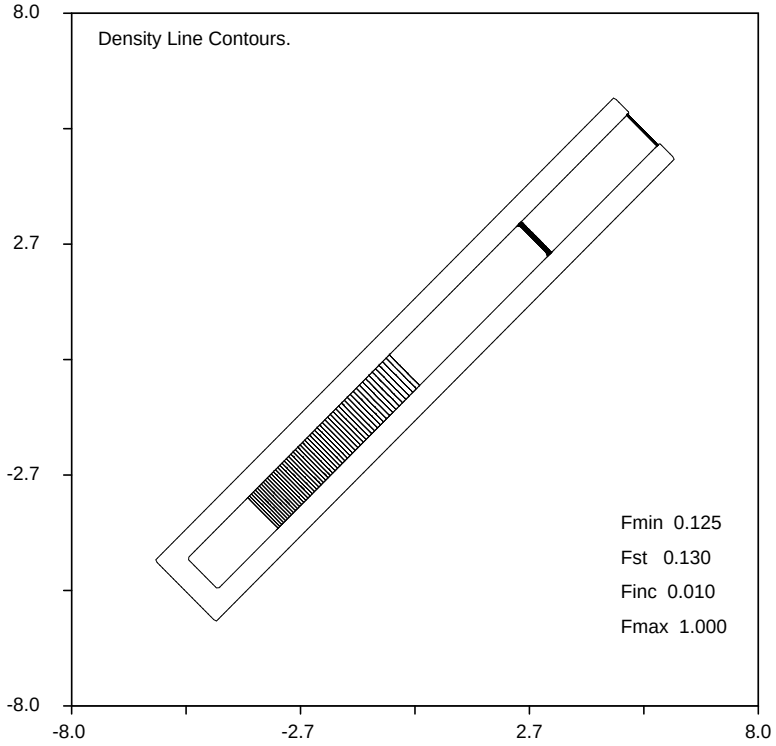

Figure 8.20: The density contours for Sod's Problem with configuration b(ii) at $t = t_f$.

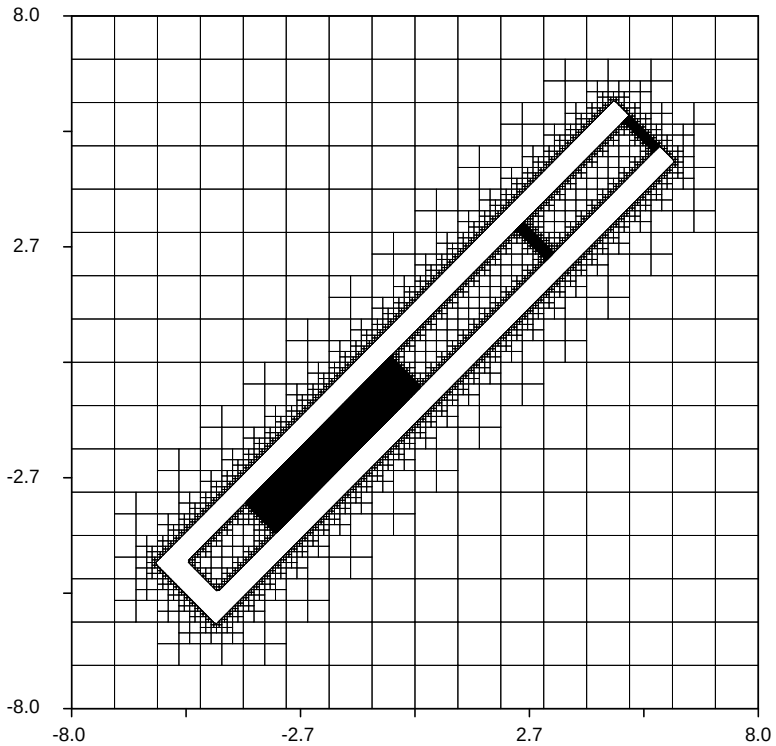

Figure 8.21: The grid plot for Sod's Problem with configuration b(ii) at $t = t_f$.

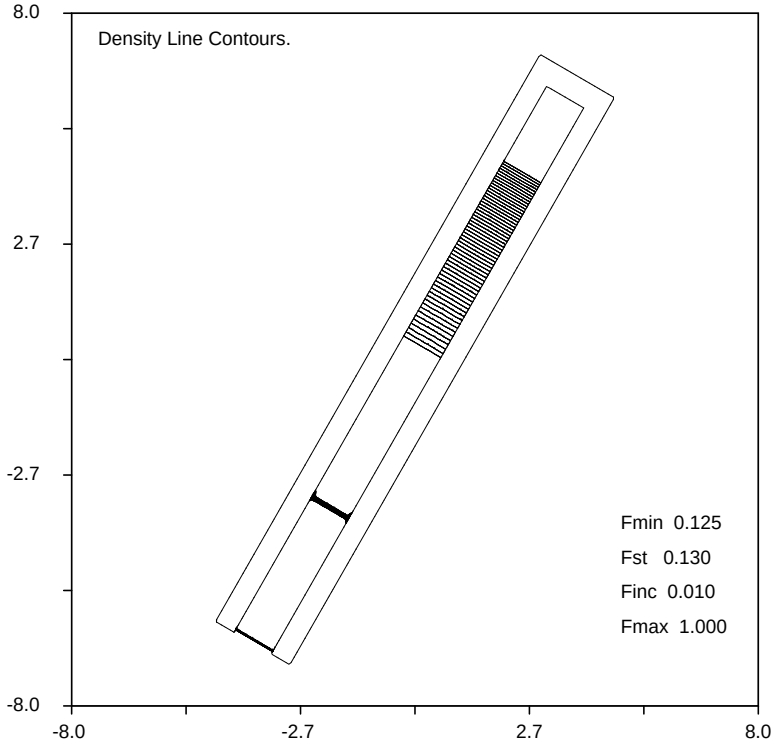

Figure 8.22: The density contours for Sod's Problem with configuration b(iv) at $t = t_f$.

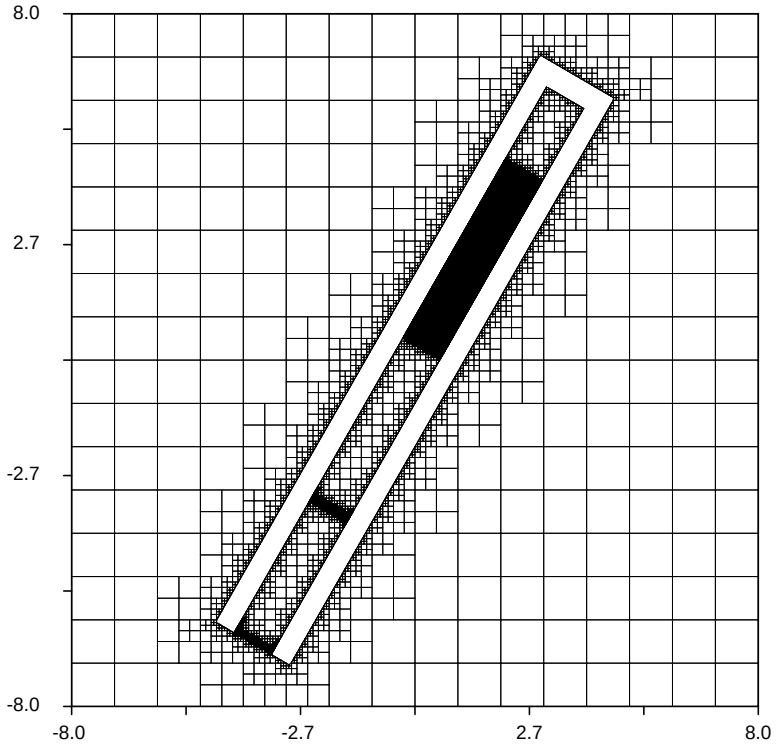

Figure 8.23: The grid plot for Sod's Problem with configuration b(iv) at $t = t_f$.

The Figures 8.18, 8.20, and 8.22 clearly show the resolution and accuracy attained in the computational simulation, especially in the clear separation of the shock and contact waves into distinct, sharp features, and the clear separation and spreading of the expansion fan, in close agreement with the corresponding analytical solution. The total distance covered by each of the main features of the flowfield (which for the expansion fan can be identified as the foot and the head of the fan) is seen to agree very closely with the corresponding distance in the analytical solution [431]. The Figures 8.19, 8.21, and 8.19 also clearly show how the solution-adaptation captures and retains the traveling flow features, and how the grid maintains an optimal refinement distribution. The six figures just mentioned together also show that the results and the refinement patterns are independent of the orientation of the geometry or the flow direction. The results for all other computations show similar solutions and adaptation patterns, with higher refinement levels consistently providing narrower and more resolved features.

An important observation from Figures 8.18, 8.20, and 8.22 concerns the local spreading of the density contours in the regions where the contact wave meets the solid boundary: while this spreading is almost non-observable for the "0-degree" orientation, it increases slightly for the "45-degree" orientation, and increases appreciably for the "240-degree" orientation. As discussed in more detail below, this spreading is caused by the cell merging and unmerging operations, which are performed at every time-step and which tend to smear features by repeated spatial averaging of the gasdynamic properties in the merged cells. The spreading is most severe for the "240-degree" case because the variation in the area of the cut cells is the greatest in that case. As will be explained again below, this spreading extends the contact wave near a solid boundary by about 1-2 cells to each side of the wave.

Therefore, the spatial extent of this spreading diminishes with increasing refinement level, to the point that it is barely noticeable for the computations performed with a refinement level of 14. The spreading is far weaker for shock waves, which tend to steepen themselves, because the signals from both sides of the shock wave propagate into the wave which therefore tends to "eat up" any noise generated ahead of or behind it.

Figures 8.24, 8.25, 8.26, and 8.27 respectively show line plots from the computational solutions for the pressure, speed, density, and Mach Number. Each plot also shows the corresponding exact analytical solution. These figures were taken from the computations performed with configuration b(i), using the Exact-Riemann numerical flux function, as explained above. The data for the line plots was taken from the sequence of computational cells cut by the center-line of the channel that confines the flow. In order to avoid cluttering the figures, only the computations with the coarsest subregion set (having a maximum refinement level of 9) and the finest subregion set (having a maximum refinement level of 14) used for this test case are presented; results from computations with intermediate refinement levels are shown separately below, in expanded (or zoomed) plots.

Figures 8.24, 8.25, 8.26, and 8.27 clearly display the high quality of the computational solutions, with a clearly-detectable second-order character in the smooth region occupied by the expansion fan, and an almost fully monotone, first-order character across discontinuities. No new local or global extrema are introduced in the density or the pressure distributions, but a slight overshoot is observed at the foot of the expansion fan in the speed distribution, and hence also at the corresponding location in the Mach Number distribution. This behavior occurs because the bounds on the limiters were purposely set for this computation to be slightly more toler-

ant to overshoots in the velocity components than to overshoots in the density or the pressure. Because of these properties, the solutions shown in the above figures appear to satisfy the "high-resolution" definition.

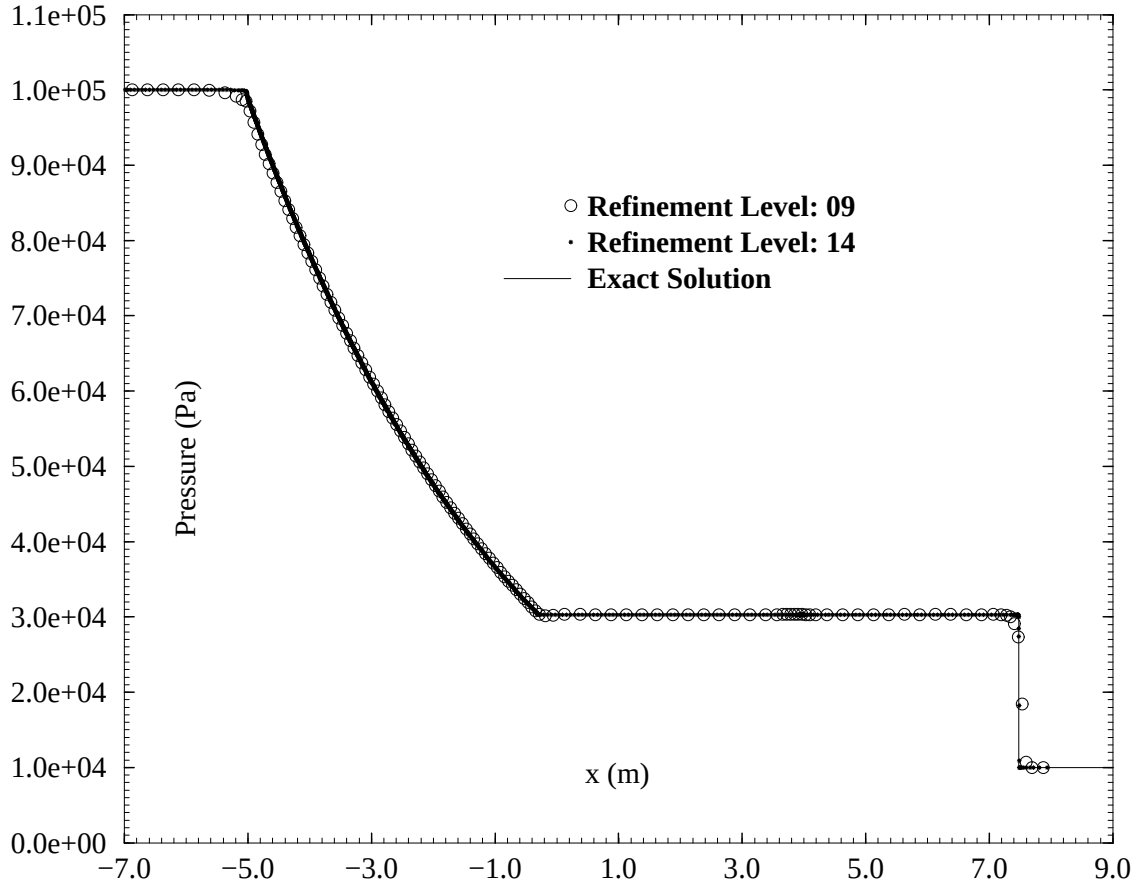

Figure 8.24: The global pressure distribution at t=0.01350s for Sod's Problem, as given by the exact analytical solution, and as predicted by the numerical solutions with the lowest maximum refinement level (9) and the highest maximum refinement level (14) used for this test case.

Close agreement of the computational predictions with the corresponding exact analytical results is clearly evident for all the primitive and derived quantities shown in Figures 8.24, 8.25, 8.26, and 8.27. The line plots for internal energy, kinetic energy and other derived variables exhibit the same behavior as other derived quantity shown in these figures; namely, the Mach Number, and are therefore omitted for

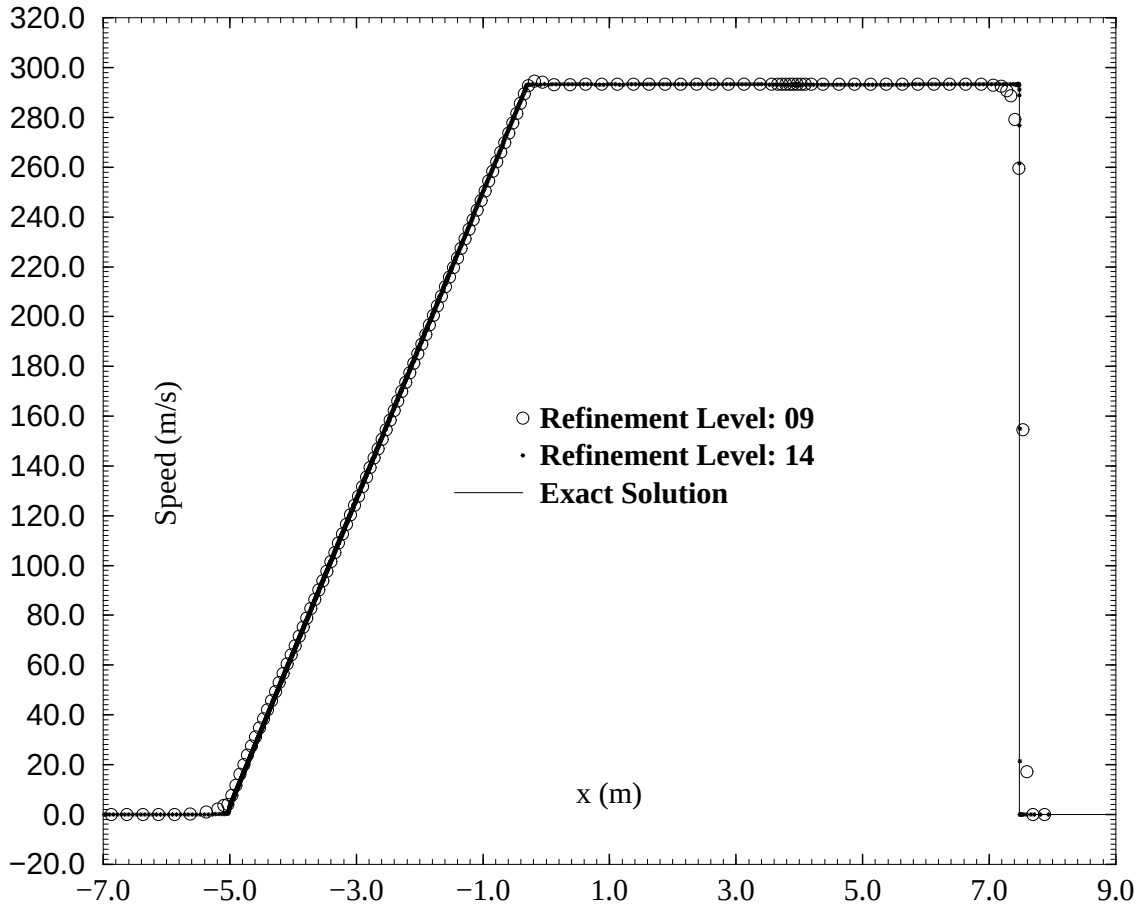

Figure 8.25: The global speed distribution at t=0.01350s for Sod's Problem, as given by the exact analytical solution, and as predicted by the numerical solutions with the lowest maximum refinement level (9) and the highest maximum refinement level (14) used for this test case.

compactness. The only exception to this pattern is the entropy, which has an appreciable overshoot after the shock, and which varies slightly along solid boundaries. The behavior of the entropy results is discussed in more detail towards the end of this sub-section.

Figures 8.28, 8.29, and 8.30 show expanded line plots of the pressure distribution, while figures 8.31, 8.32, 8.33, and 8.34 show expanded line plots of the speed distribution, while Figures 8.35, 8.36, 8.37, and 8.38 show expanded line plots of the density distribution, while Figures 8.39, 8.40, 8.41, and 8.42 show expanded plots

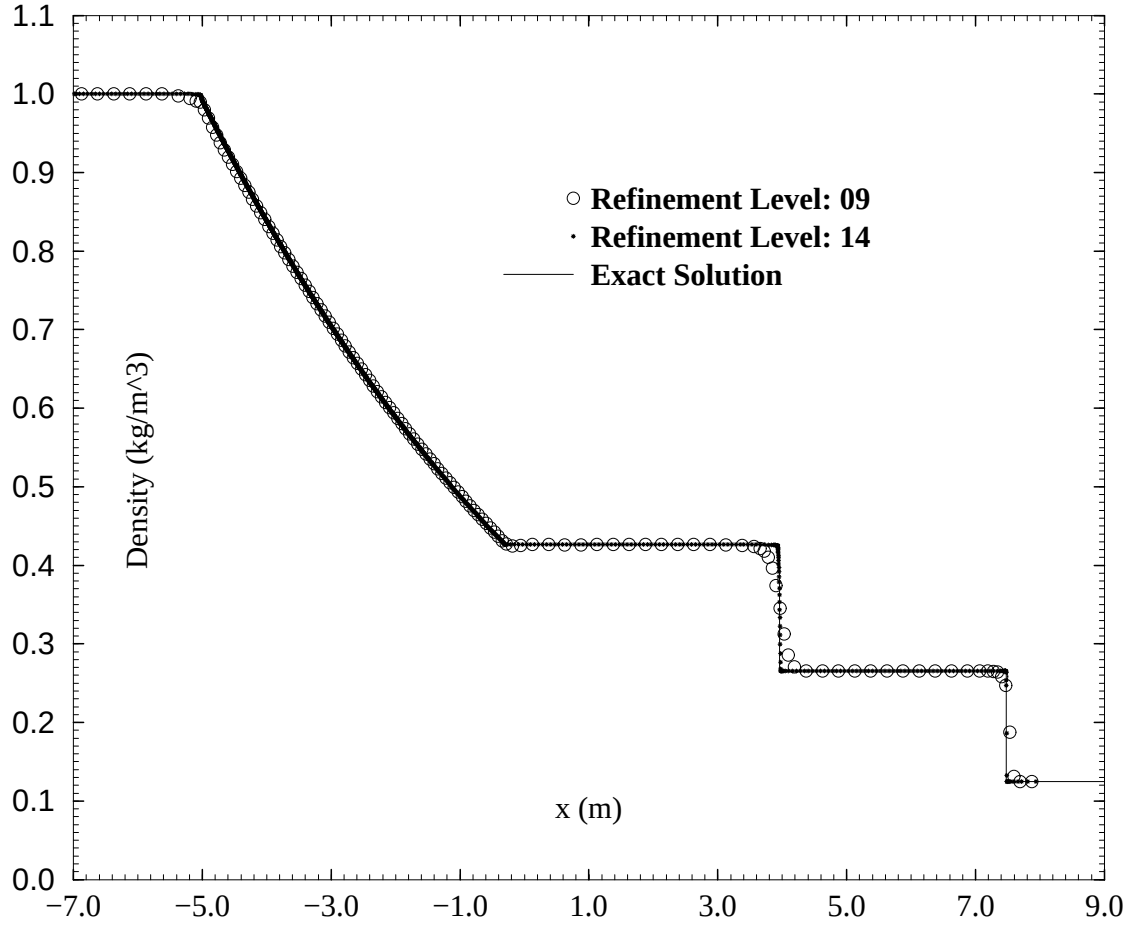

Figure 8.26: The global density distribution at t=0.01350s for Sod's Problem, as given by the exact analytical solution, and as predicted by the numerical solutions with the lowest maximum refinement level (9) and the highest maximum refinement level (14) used for this test case.

of the Mach Number distribution. As indicated in their respective titles, each of the expanded plots is centered about a critical flow feature for the plotted quantity. Unlike the treatment for the global plots, the expanded plots show the results for all six maximum refinement levels used.

The expanded plots listed above show more clearly and conclusively several features that can be detected at least faintly in the global plots of Figures 8.24, 8.25, 8.26, and 8.27 above, as well as some new ones that are not visible in these global plots.

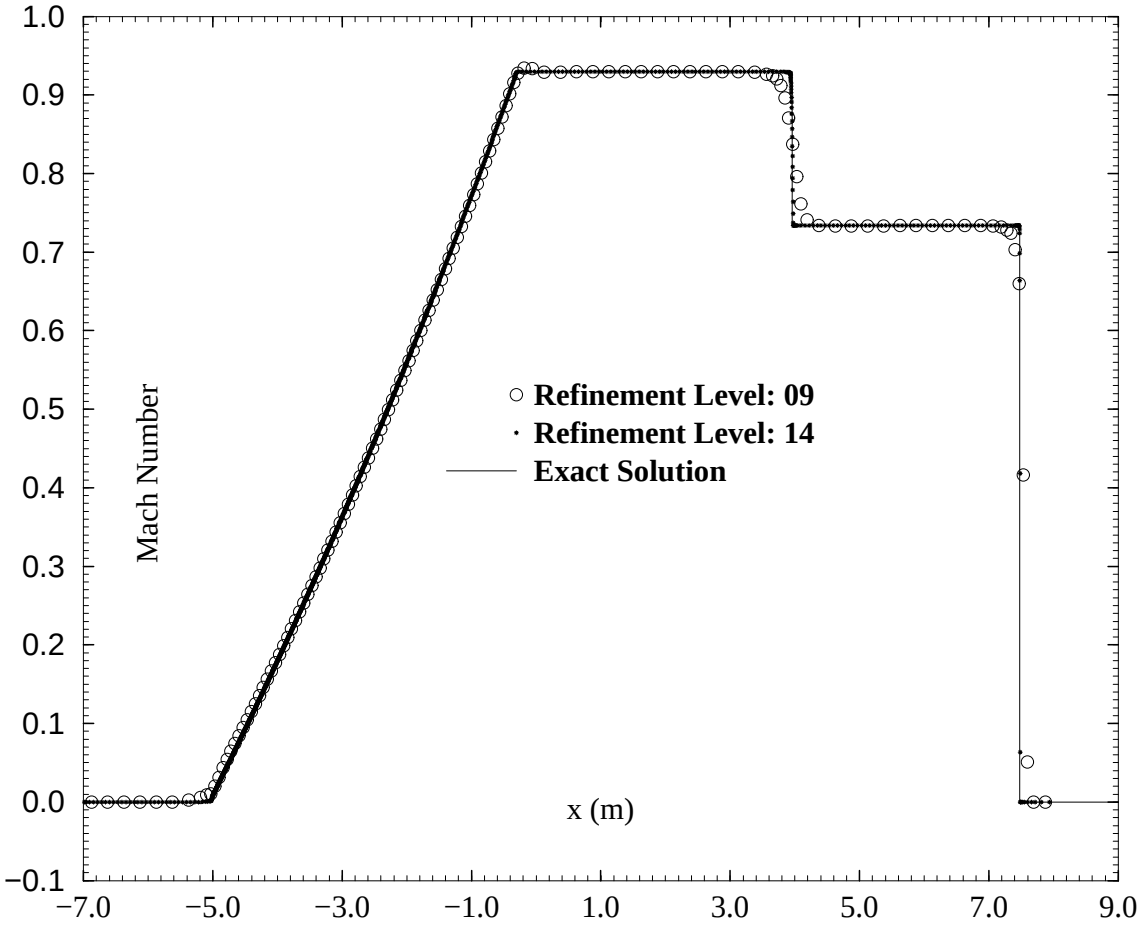

Figure 8.27: The global Mach Number distribution at t=0.01350s for Sod's Problem, as given by the exact analytical solution, and as predicted by the numerical solutions with the lowest maximum refinement level (9) and the highest maximum refinement level (14) used for this test case.

The first of these features is that the jump conditions across all discontinuities, and the average propagation speed of these discontinuities is correctly and accurately predicted. In addition, satisfaction of the jump conditions and correct prediction of the average propagation speeds do not depend on the refinement level.

The second of these features is that the accuracy and the resolution of the computational solutions for all the quantities examined improve with increasing maximum refinement level over the range of maximum refinement level examined, and that this improvement and this approach to the exact solution is almost uniform with

increasing maximum refinement level. Near the head and foot of the expansion fan, increasing the maximum refinement level is seen to "squeeze" the region of greatest error increasingly closer towards the vertices of the curve representing the analytical solution. Although the L_2 or L_∞ error norms of the computational solution (relative to the exact solution) are not computed here, at least the first of these norms visually appears to decrease in a monotone manner with increasing maximum refinement level. More specifically, the expanded plots visually show that most of the L_2 error norm in the computational solution is accumulated across the discontinuities, and since the computational solution there is first-order accurate (because of the gradient limiting), the overall L_2 error norm for this problem would be expected to decrease linearly (rather than quadratically) with the smallest cell dimension, even though the computational scheme used is formally second-order accurate.

The third of these features is that number of computational cells required to capture a discontinuity appears to be independent of the cell size (or the maximum refinement level), suggesting that the computational scheme enables the approach to the exact analytical solution to be indefinitely pursued (to the limits placed by the precision of the arithmetic operations, and the limits placed by the presence of other sources of error) by increasing the spatial and temporal resolutions. In the case of the shock wave, about 95% of the jump across the discontinuity is resolved within 8 cells, and more than 99% of the discontinuity is resolved within 10-12 cells. In the case of the contact wave, about 95% of the jump is resolved within about 12 cells, and more than 99% of the jump is resolved within about 16 cells. As expected, the contact is more smeared than the shock because, as described above, only the latter has an intrinsic self-steepening effect that acts to counter the numerical diffusivity of the computational scheme. The number of cells required to resolve a shock wave

and a contact wave accords closely with what is expected in a first-order-accurate computation (which is the order of accuracy to which the computational scheme degenerates to in the regions of the discontinuities), and hence with what is expected in a scheme satisfying the high-resolution definition. The invariance with maximum refinement level of the number of cells required to resolve a jump again implies that the overall L_2 error norm in the computational solution for this particular test case would be expected to decrease linearly, not quadratically, with the smallest cell dimension.

In all the computations presented and discussed in this sub-section, it should be noted that since a uniform, constant CFL Number (0.8) is used, an increase in the spatial resolution is automatically accompanied by an increase in the temporal resolution. Thus, the improvements that are observed in the solution with decreasing cell size are a result of the increased resolution of both the spatial and temporal discretizations.

An important consideration when using a sub-region set with a predominant geometric alignment, as is the case with a Cartesian grid, is the effect of this alignment on the overall performance and accuracy of the computational scheme. This issue is explored here by examining how the computational solution for Sod's Problem is affected by changing the orientation of the "shock-tube", as described above.

Figures 8.43, and 8.44 respectively show a global and a zoomed line plot of the density distribution for computations with three different orientation angles (namely, 0, 45, and 240 degrees relative to the x-axis), corresponding to the configurations b(i), b(ii), and b(iv), respectively, as described above). All these figures are taken from computations carried out with the baseline case (having a maximum refinement level of 12).

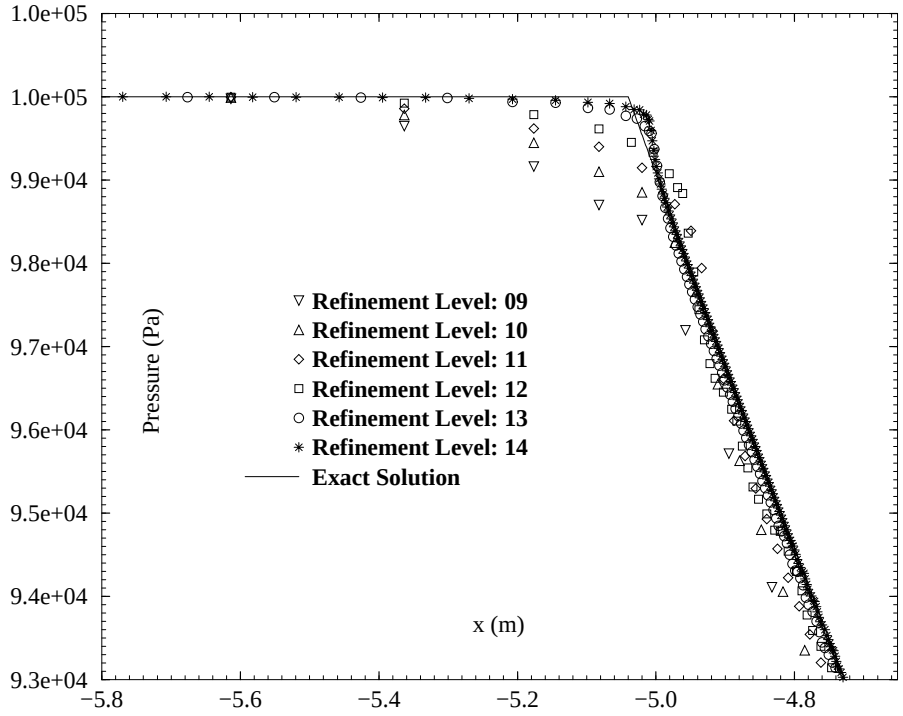

Figure 8.28: Zoomed plot of the pressure distribution in the vicinity of the head of the expansion fan for Sod's Problem.

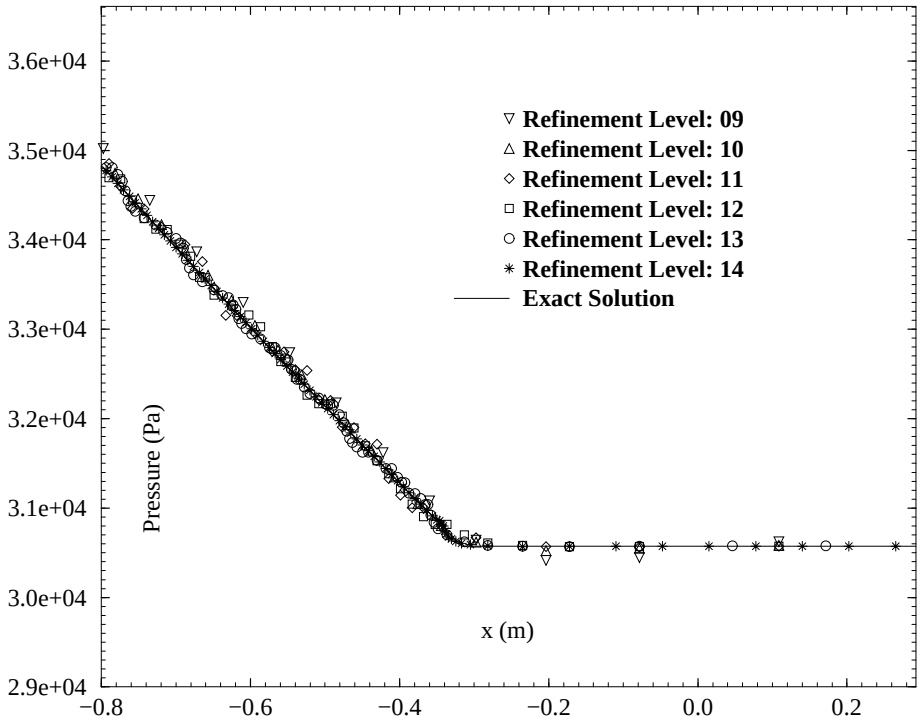

Figure 8.29: Zoomed plot of the pressure distribution in the vicinity of the foot of the expansion fan for Sod's Problem.

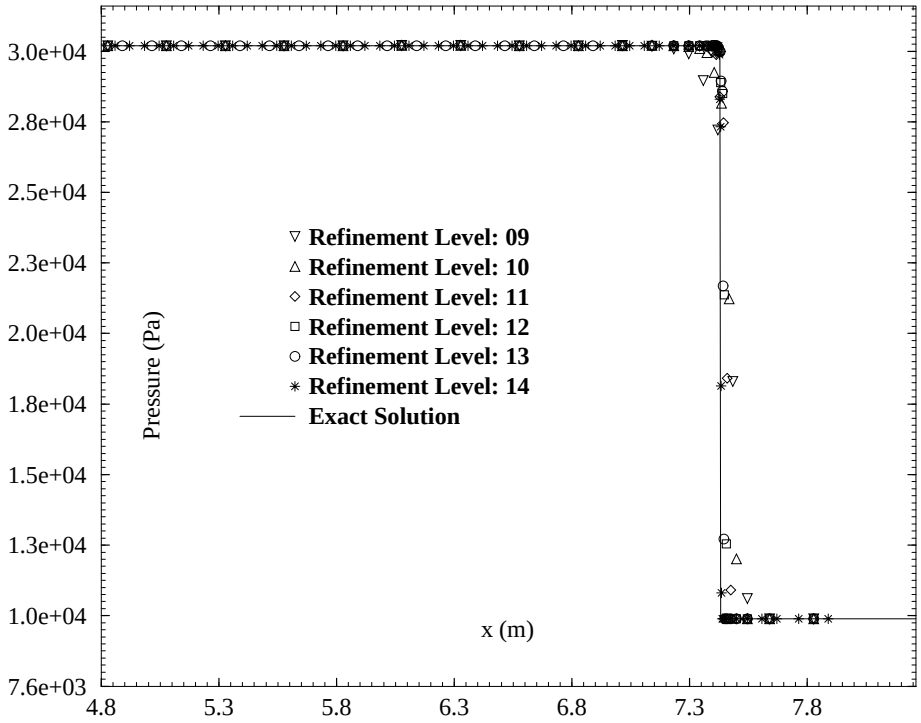

Figure 8.30: Zoomed plot of the pressure distribution across the shock wave for Sod's Problem.

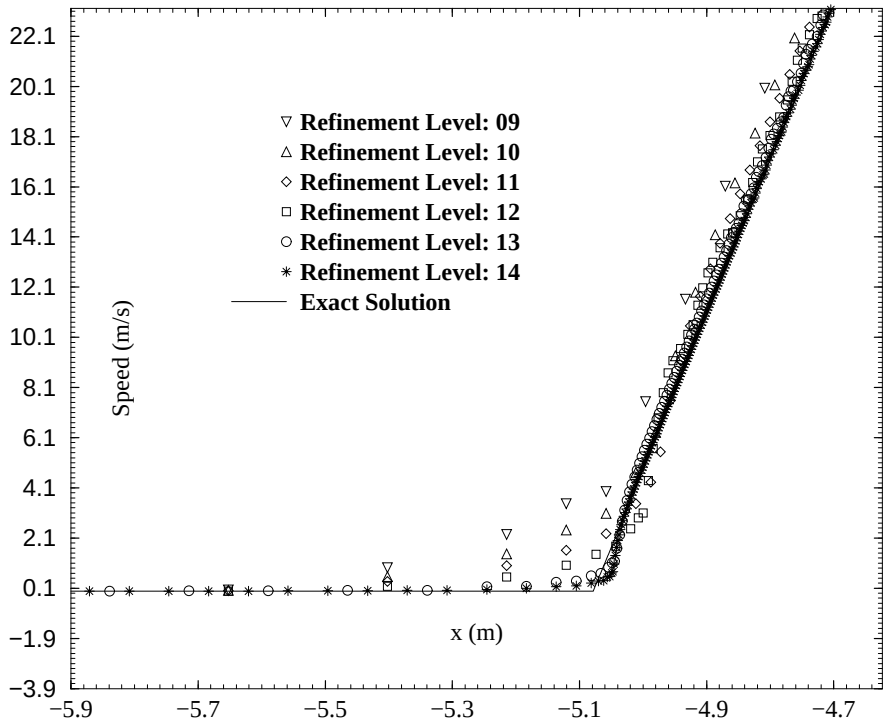

Figure 8.31: Zoomed plot of the speed distribution in the vicinity of the head of the expansion fan for Sod's Problem.

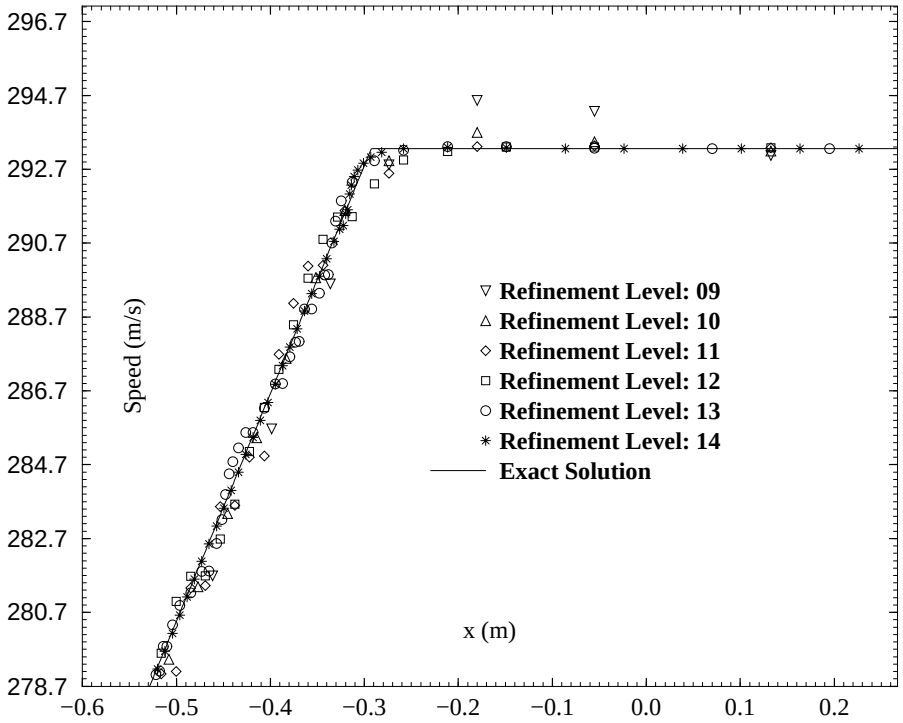

Figure 8.32: Zoomed plot of the speed distribution in the vicinity of the foot of the expansion fan for Sod's Problem.

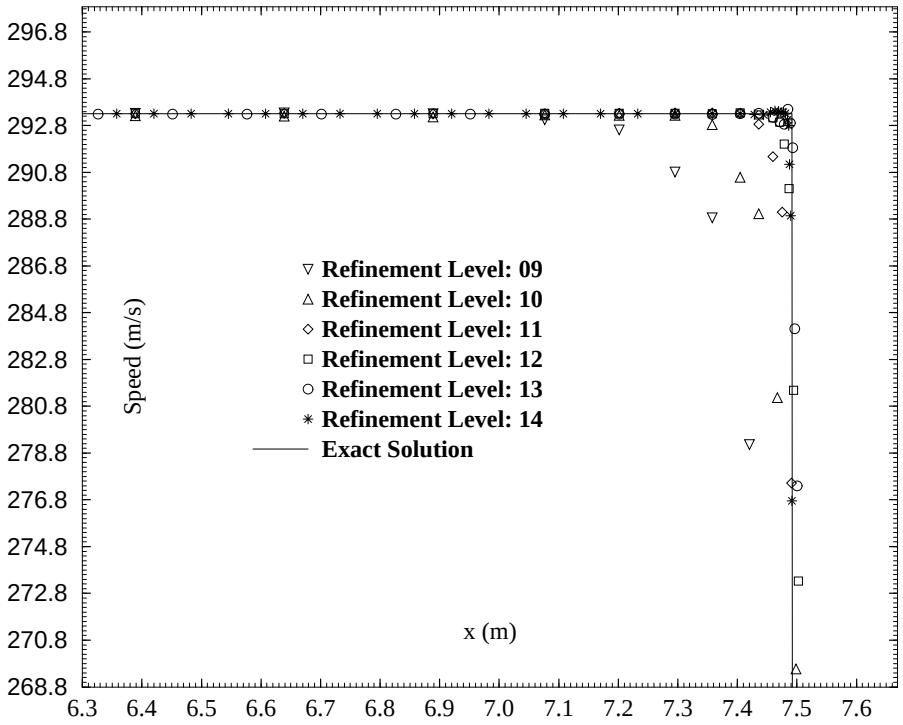

Figure 8.33: Zoomed plot of the speed distribution behind the shock wave for Sod's Problem.

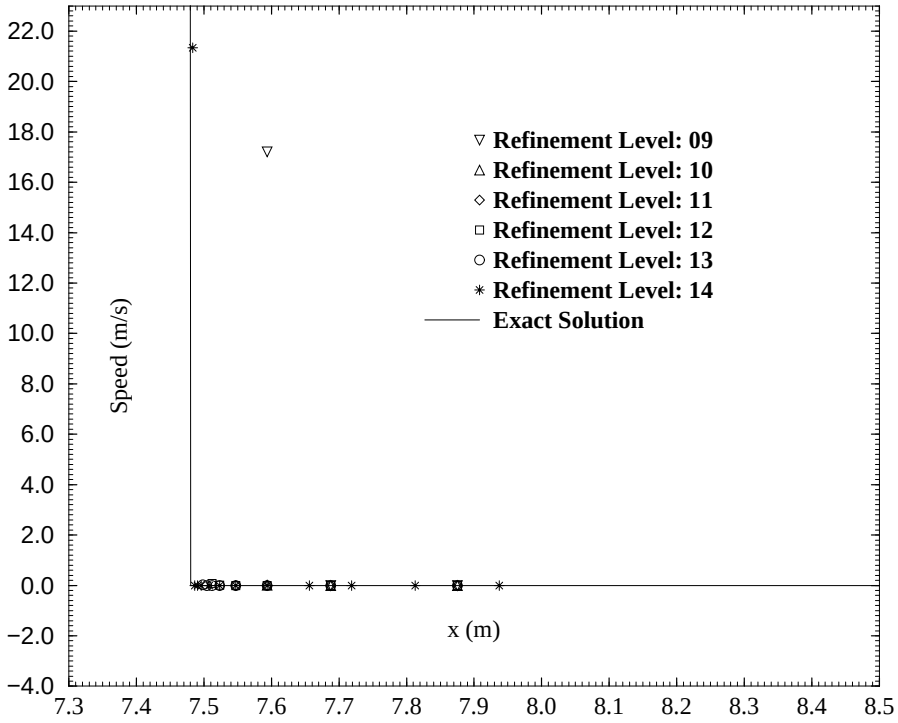

Figure 8.34: Zoomed plot of the speed distribution in-front of the shock wave for Sod's Problem.

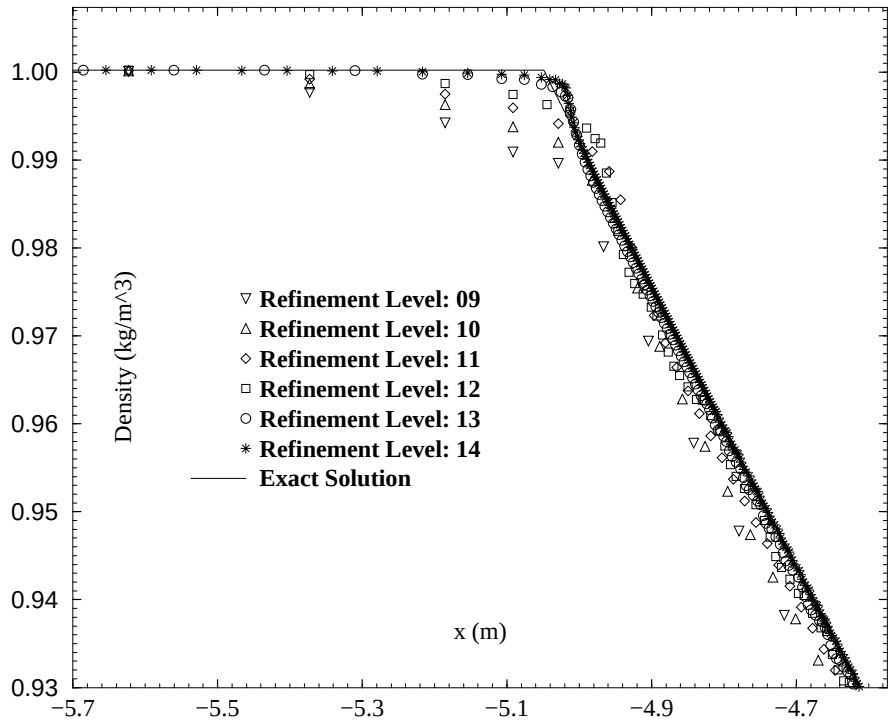

Figure 8.35: Zoomed plot of the density distribution in the vicinity of the head of the expansion fan for Sod's Problem.

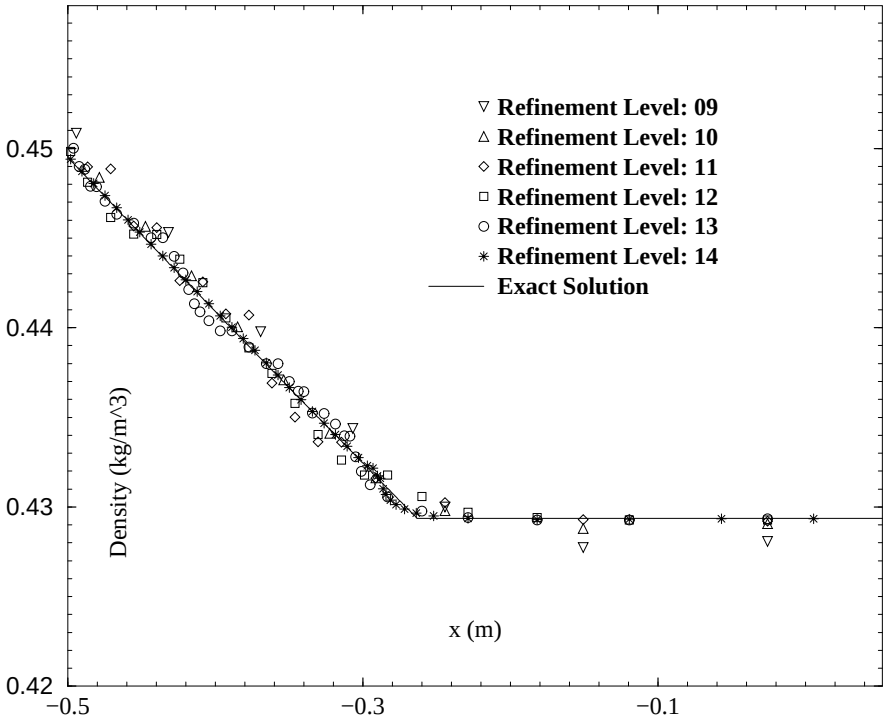

Figure 8.36: Zoomed plot of the density distribution in the vicinity of the foot of the expansion fan for Sod's Problem.

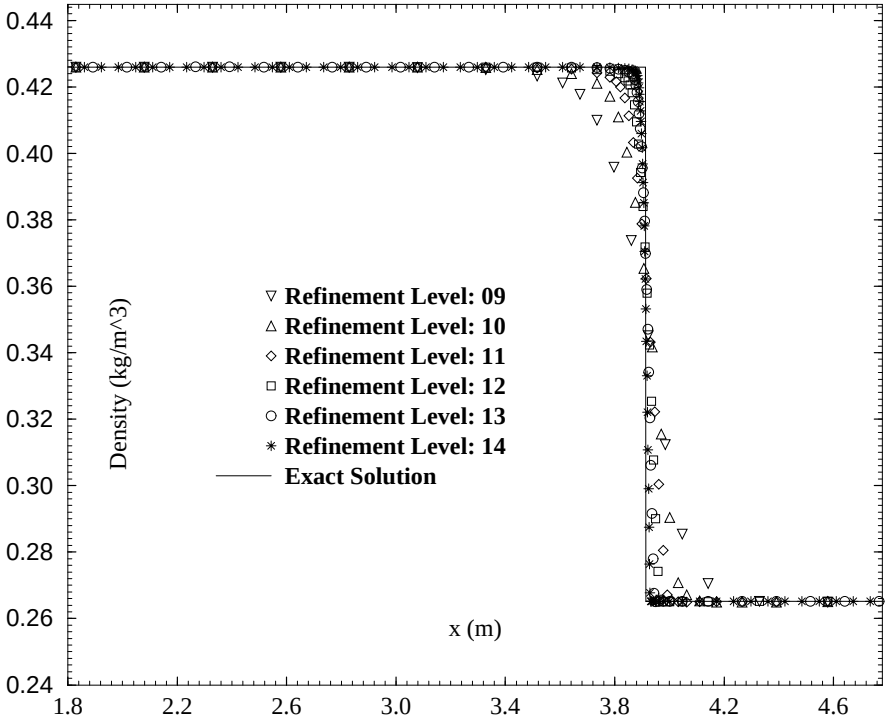

Figure 8.37: Zoomed plot of the density distribution across the contact discontinuity for Sod's Problem.

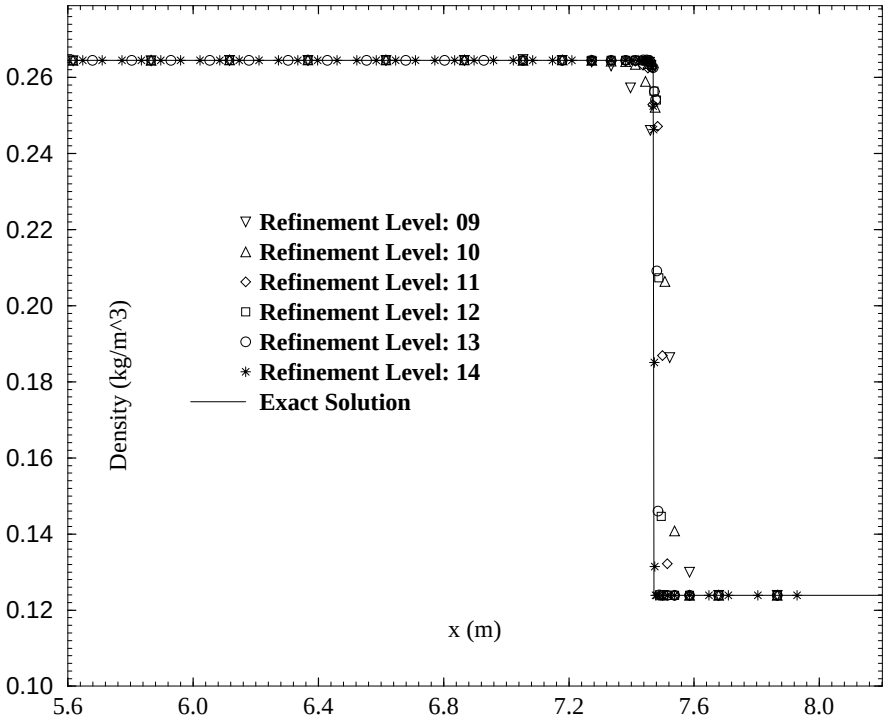

Figure 8.38: Zoomed plot of the density distribution across the shock wave for Sod's Problem.

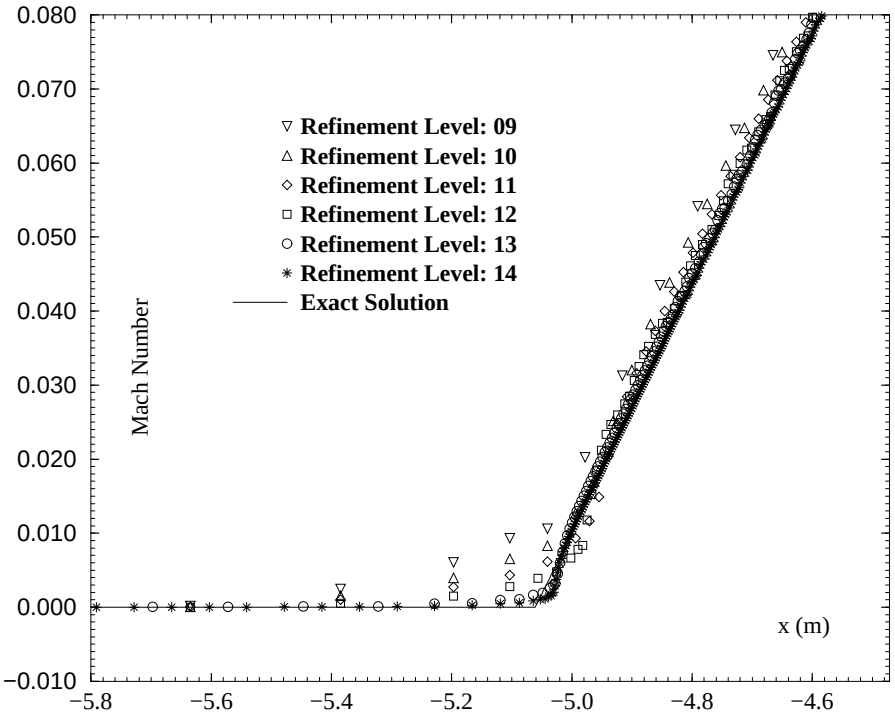

Figure 8.39: Zoomed plot of the Mach Number distribution in the vicinity of the head of the expansion fan for Sod's Problem.

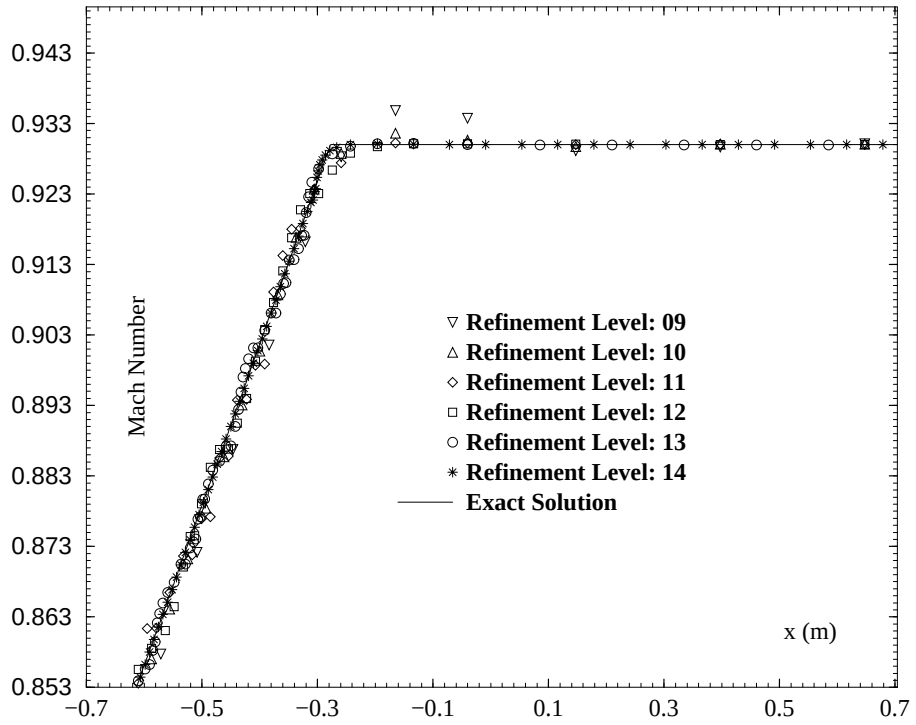

Figure 8.40: Zoomed plot of the Mach Number distribution in the vicinity of the foot of the expansion fan for Sod's Problem.

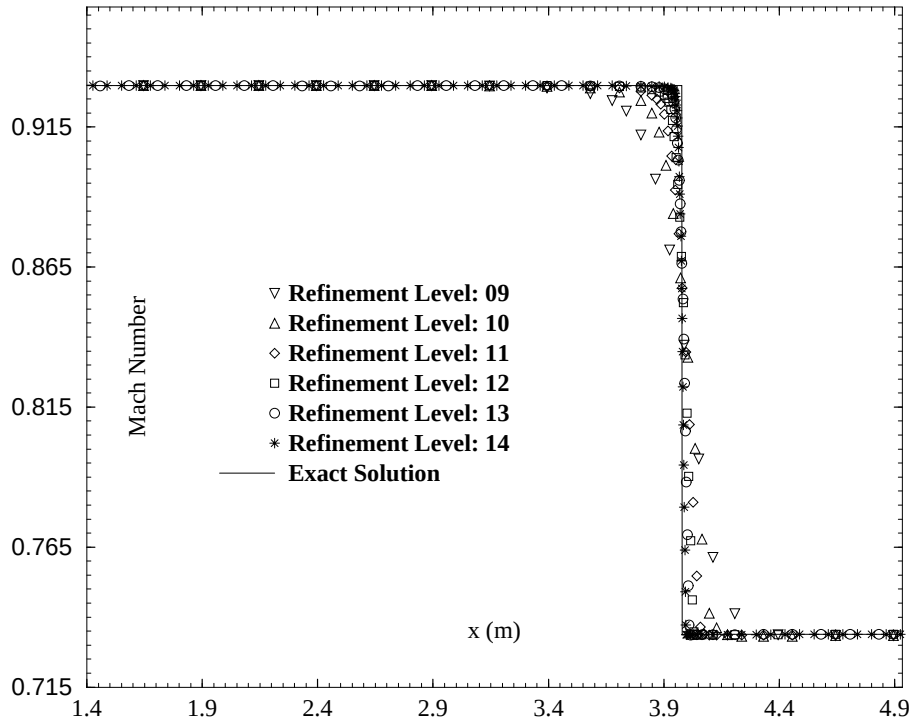

Figure 8.41: Zoomed plot of the Mach Number distribution across the contact discontinuity for Sod's Problem.

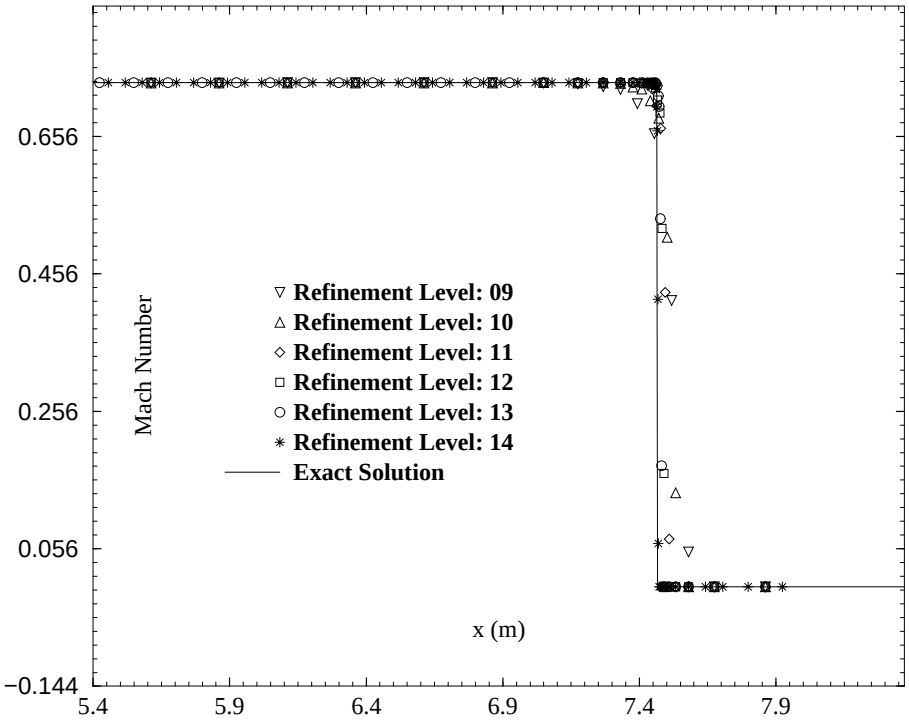

Figure 8.42: Zoomed plot of the Mach Number distribution across the shock wave for Sod's Problem.

Figures 8.43, and 8.44 show that there is a slight effect of the orientation on the solution. However, they also show that all the important features and properties of the solution, such as the mean propagation speed, and the correct resolution of all discontinuities and smooth features, are unaffected. Indeed, although differences can be detected between the results for different orientations, it is difficult to tell which orientation produces the best solution. It should also be noted that compared to the zero angle orientation, the orientations with a non-zero angle include the effects of cell-cutting with large area variations, the effects of non-ideal specification of the initial conditions, and include more adverse effects on the solution from cell merging and cell unmerging. All three of these additional effects taken together appear to have no appreciable effect on the quality or accuracy of the solution. This issue is discussed further below when examining the entropy production in the solution. The

differences between the computational solutions with different orientations are also observed to decrease with increasing maximum refinement level, to the extent that they can barely be detected when the maximum refinement level is 14.

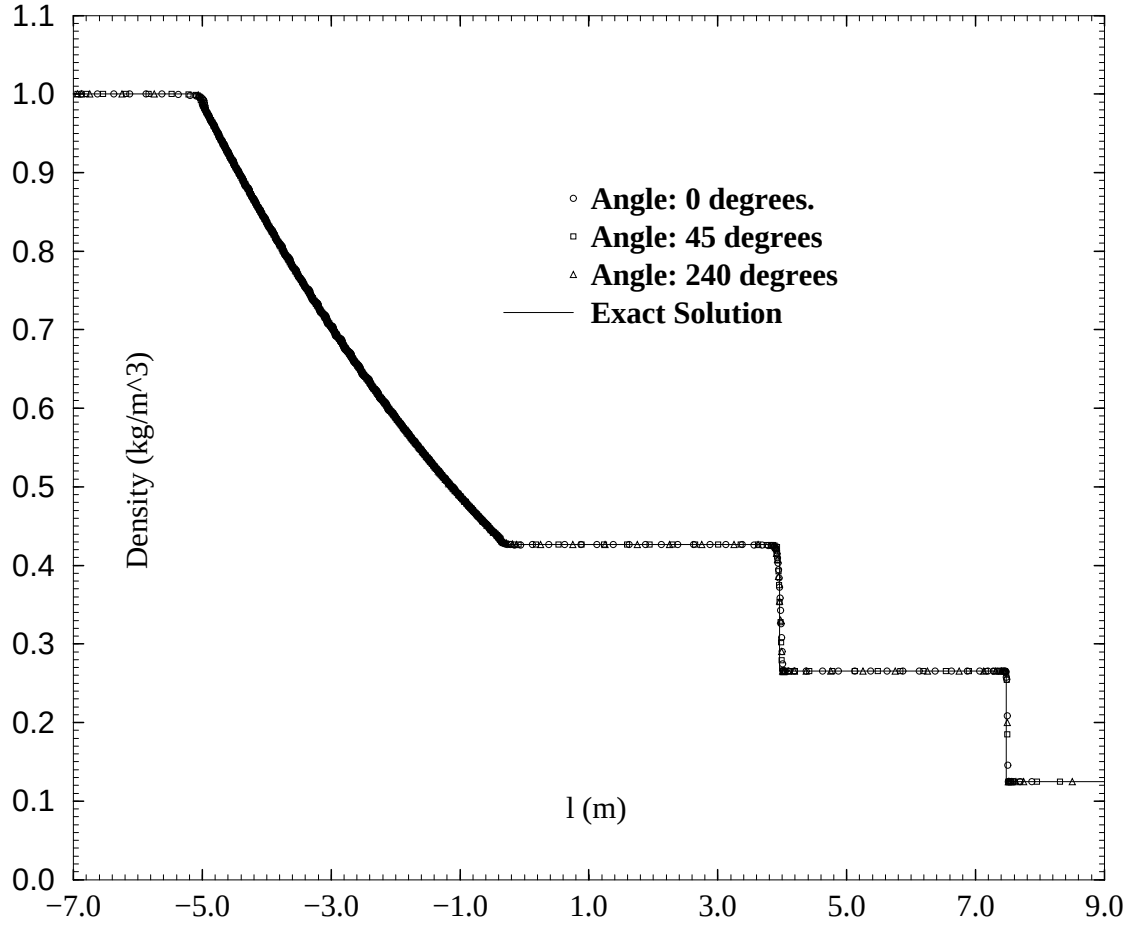

Figure 8.43: The global density distribution at t=0.01350s for Sod's Problem, as given by the exact analytical solution, and as predicted by the numerical solutions with three different alignments (corresponding to those of configurations b(i), b(ii), and b(iv), with a maximum refinement level of 12.

As explained in Chapter II and in Section 6.5, the entropy production in a smooth inviscid flow provides a measure of the numerical diffusion in the solution, and is therefore a useful indicator of the solution quality. Figure 8.45 shows a global line plot of the entropy, clearly showing the expected three distinct levels of entropy in

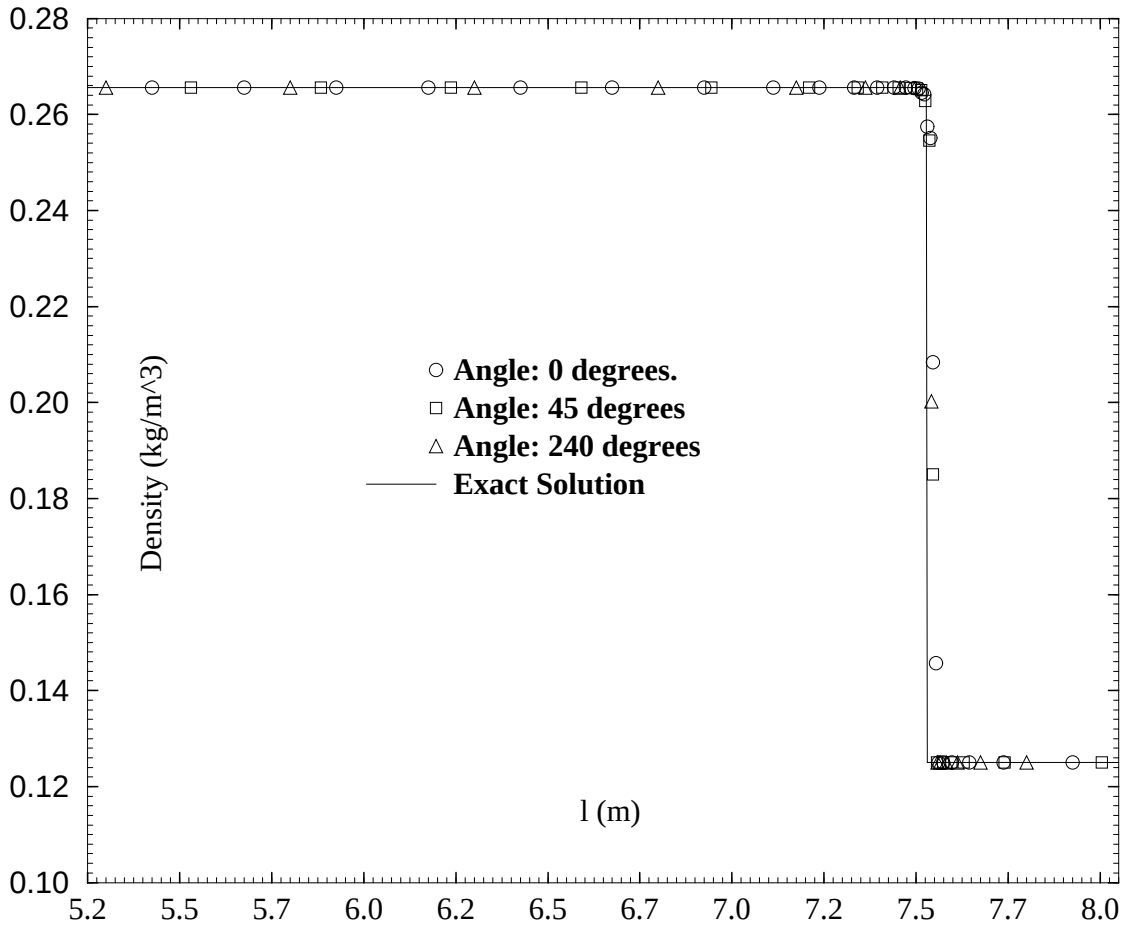

Figure 8.44: A zoomed plot of the density distribution at t=0.01350s around the shock wave for Sod's Problem, as given by the exact analytical solution, and as predicted by the numerical solutions with three different alignments (corresponding to those of configurations b(i), b(ii), and b(iv). The results are for computations with a maximum refinement level of 12.

the computational region: (i) the entropy of the initial Left State; (ii) the entropy of the initial Right State; and, (iii) the entropy of the "shocked" region. No entropy variation should occur in the uniform flow regions, nor in the expansion fan (which is generated through an isentropic process). The entropy jump across the shock wave agrees closely with the exact analytical solution, except for the overshoot immediately behind the shock, which is discussed in more detail below. The figure clearly shows the uniformity of the entropy to the left and right of the shocked region, and clearly

shows the preservation of the entropy in the expansion fan and in the uniform regions, in accordance with the corresponding theoretical requirements.

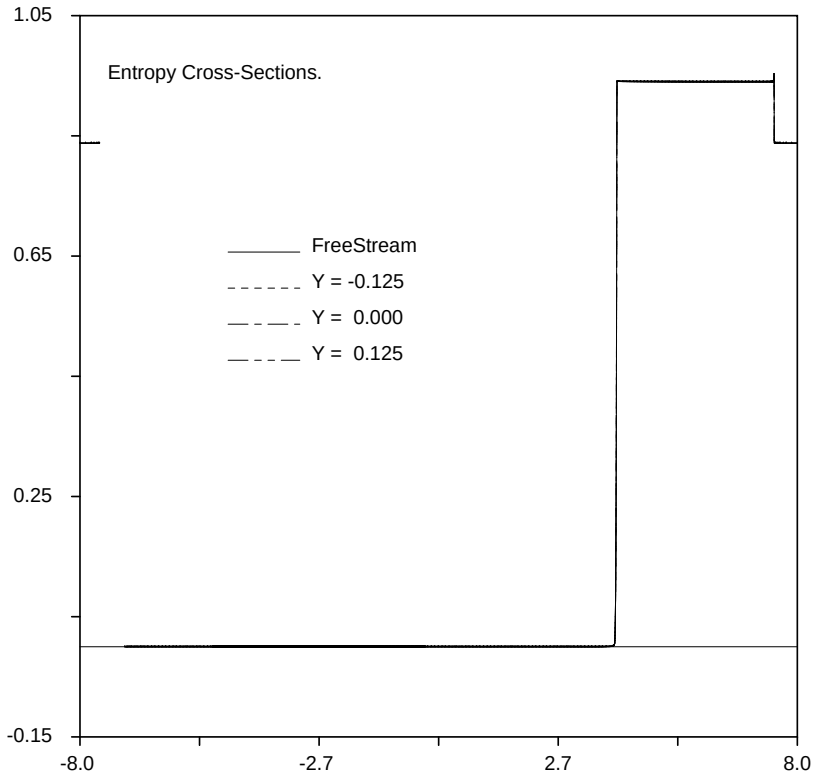

Figure 8.45: The global entropy distribution at t=0.01350s for Sod's Problem, as computed with a maximum refinement level of 14.

Figure 8.46 shows an expanded and scaled view of the entropy variation throughout the expansion fan, with three line plots: one along the center-line of the shock-tube, and one along each of the inner walls of the shock-tube. The line plots in this figure show more clearly than the corresponding line plots in Figure 8.45 that the entropy production in smooth regions is negligible. Even more importantly, comparison of the line plot along the center-line and the two line plots along the walls of the shock-tube demonstrates that the effect of the boundary conditions, the cell-cutting, and the cell merging and unmerging all together cause very little entropy production, and that the overall effect almost appears as random fluctuations.

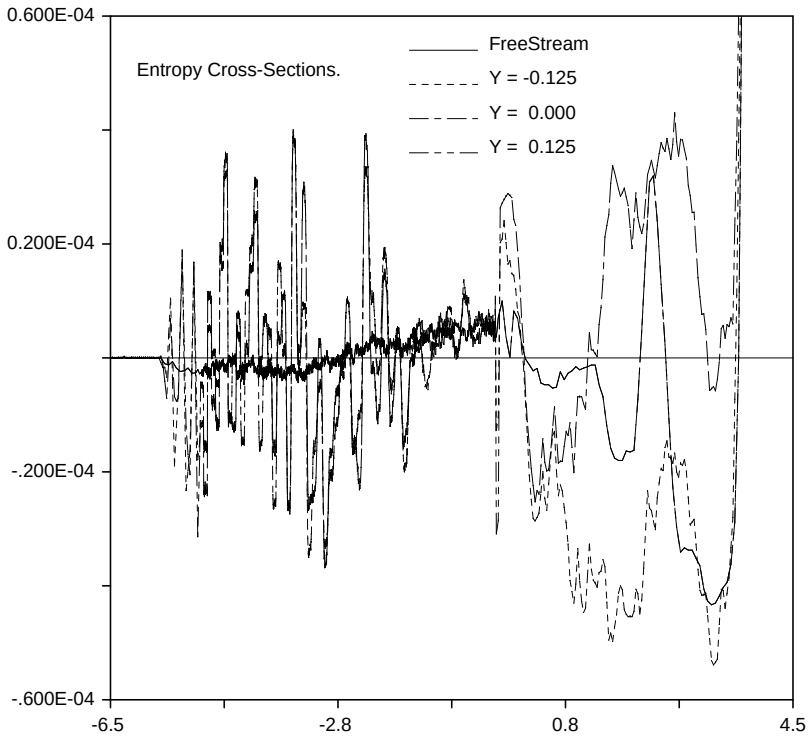

Figure 8.46: Zoomed and scaled plots of the entropy distribution, computed with a maximum refinement level of 14, in the expansion fan region in Sod's Problem at t=0.01350s. The distributions are taken along the centerline and along the upper and lower bounding walls of the bounding channel of configuration b(i).

The entropy distributions presented in Figures 8.45 and 8.46 are taken from computations with configuration b(i) at the highest maximum refinement level (14) used for this test case, and therefore display the best results obtained for entropy generation in isentropic regions. Lower spatial resolutions, and lack of perfect alignment with the grid (as occur with all computations with configurations b(ii) and b(iv)) increase the entropy production from the levels shown in Figures 8.45 and 8.46. In particular, the entropy production in the expansion fan in the worst case (which occurs with configuration b(iv) with a maximum refinement level of 9) is almost 20 times greater than the entropy production levels shown in Figure 8.46. As explained above, among the four configurations presented in this sub-section, the configuration

with the 240-degree angle results in the greatest variation of cell area, and in the greatest level of diffusivity from the cell merging and unmerging, and hence in the most adverse effect on the entropy production.

Figure 8.47 shows an expanded and scaled plot of the entropy production immediately behind the shock. The figure shows clearly the extent of the overshoot, which is relatively much greater than the overshoot in any of the primitive quantities. This behavior is usually attributed to the fact that the limiting is applied to the gradients of the primitive quantities instead of to the gradients of the characteristic variables. If the characteristic variables were limited instead, the overshoot in the entropy will be significantly reduced, but the overshoots in the primitive variables will be increased. The figure demonstrates again that the presence of solid boundaries, the application of the "impermeability" boundary conditions, and the cell merging and unmerging have little consistent effect on the behavior of the entropy distributions. This can be concluded from the fact that the magnitude of the entropy overshoot does not show any consistent difference between its behavior along the center-line and its behavior along the inner walls of the bounding channel.

As discussed towards the beginning of this sub-section, Sod's Problem does not present a severe test for a flow solution algorithm. In order to specifically test the reliability and robustness of the flow-solver more conclusively, Sod's Problem was computed with a pressure ratio of 10^7 instead of 10, and with all other ratios remaining unchanged. The computation was performed without any solver failures, and the results were again in close agreement with the exact analytical solution. With such a high pressure ratio, however, most of the pressure drop occurs across the expansion fan, so that the shock can barely be detected as a discontinuity. Several other such computations were performed with different left and right states, and the

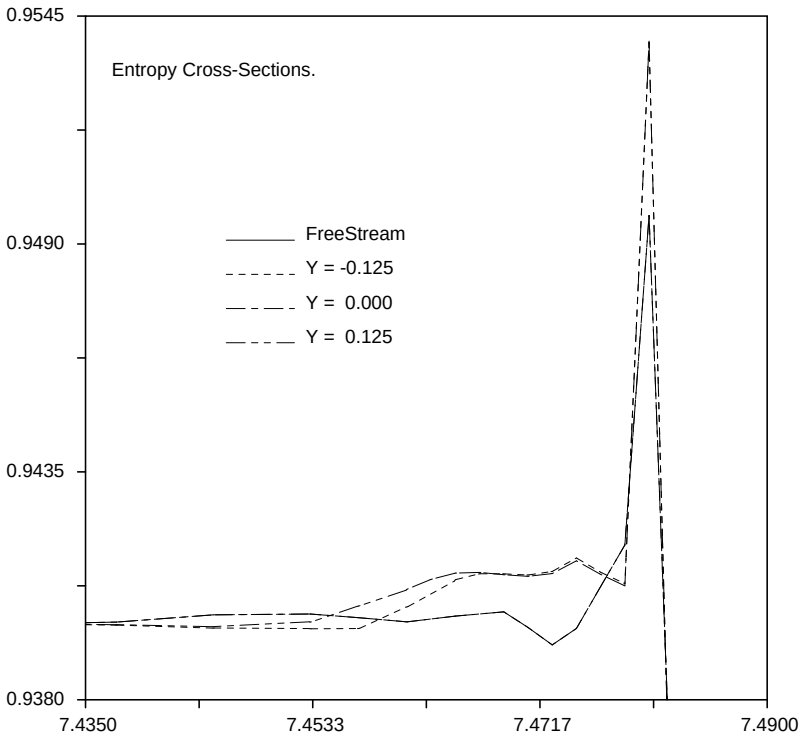

Figure 8.47: Zoomed plots of the entropy distribution, computed with a maximum refinement level of 14, immediately behind the shock wave in Sod's Problem at t=0.01350s. The distributions are taken along the center-line and along the upper and lower bounding walls of the bounding channel of configuration b(i).

results of these computations were used to validate the overall computational scheme and the Exact-Riemann numerical flux function implemented in this work.

8.4.2 Diffraction of Shocks Over Triangular Wedges

This sub-section presents and discusses the main computational results obtained for three different test cases involving the reflection and diffraction of planar shock waves over triangular wedges. Two of the test cases feature a double-wedge geometry, while the third features a single-wedge geometry.

The computational results shown in this sub-section are presented in the form of contour-line plots for the computed density ratios, in order to enable direct comparison with the corresponding experimental data. The contour-line plots for the three

test cases are all drawn with 50 contours.

When a planar shock wave collides with a rigid planar surface which is inclined to it at an angle $0 < \theta < 90$ degrees , the resulting pattern for the flow field and the reflected and diffracted waves falls into one of two different main categories: (i) Regular Reflections; and, (ii) Irregular Reflections. The Irregular Reflection patterns are further classified into so-called von Neumann Reflections, and Mach Reflections. The Mach Reflections are further classified into Direct, Stationary, and Inverse Mach Reflections. The Direct Mach Reflections are further classified into Single Mach Reflections, Transitional Mach Reflections, and Double Mach Reflections, with the latter category having further sub-categories. The overall behavior and the characterizing physical features of the reflection patterns and flow fields named above, and the dependence of the reflection pattern on the angle between the shock wave and the reflecting surface and on the Mach Number of the incident shock wave are discussed in detail in [41]. The rigid planar surface just described is often occasionally termed a "single-wedge" or a "single-ramp" obstacle.

As would be expected, the classification scheme developed for the flowfields and the reflection patterns for single-wedge geometries can be extended for double-wedge geometries. The main complication in doing so is the additional requirement (relative to the requirements for single-wedge geometries) for prediction of the specific combination of reflection patterns that will arise on both of the two ramps for a given specific geometry of a double-wedge and for a given Mach Number. The main characteristics of double-wedge reflections, and all seven possible combinations of reflection patterns from double wedges are discussed in detail in [41] and [42].

In the first test case presented in this sub-section, a planar shock wave which is perpendicular to the x-axis, and which is traveling from left to right (parallel to the

x-axis) with a Mach Number of 2.16, passes over a double-wedge having a horizontal extent of 5 units, a vertical extent of 4 units, and wedge angles of 20 and 55 degrees. The geometries of the shock wave and the double wedge, and their relative locations at the start of the computation are shown in Figures 8.48 and 8.49.

The specific test case described in the preceding paragraph has been studied experimentally [178], and has been used on several occasions for demonstrating, testing, or benchmarking algorithms for computation simulation of unsteady, compressible, inviscid flows, for example, in [292], [178], and [261].

A negligibly thin, horizontal "splitter-plate", approximately one unit long, was also attached at the leading edge of double-wedge, as shown in Figure 8.48. This was done to prevent any interaction (across the subsonic region that eventually forms around the leading edge of the double-wedge) between the stream that flows below the double-wedge and the one that flows above it, in order to simulate the experimental conditions of [178] as accurately as possible.

As expected, no observable change in the shock wave occurs occurs while it propagates downstream along the splitter-plate, towards the first ramp of the double-wedge. As the incident shock collides with the lower ramp of the double wedge, a single Mach Reflection, with its characteristic Mach Stem, is immediately formed. The Mach Stem in this case consists of a curved reflected shock, a straight Mach Shock, and a straight vortex sheet (which also coincides with a contact surface), all meeting at a single point, called the **triple point**, as shown in Figure 8.50, and as explained in more detail in [41]. The incident shock also meets the three discontinuities just described in the triple point. As the incident shock propagates further downstream along the first ramp of the double-wedge, the Mach Stem grows in size, with gradual increases in the areas of the two regions bounded wholly by discontinu-

ities or rigid boundaries, and with a slow gradual increase in the maximum density ratio, which in this case occurs at the leading edge of the first wedge. As described in [41], the triple point in this case follows a straight-line trajectory. The triangle-shaped region between the Mach Shock and the vortex sheet, which is clearly visible in Figure 8.50, has uniform properties, which do not change with time (or propagation distance). The flowfield and the overall wave reflection pattern as the Mach Shock approaches the end of the first ramp are shown in Figure 8.50.

Once the Mach Shock collides with the second ramp, a second Mach Stem, with its own triple point, is again immediately formed, as shown in Figure 8.52. As the latter figure also shows, during the early stages of its formation, the second Mach Shock propagates backward into the triangular "stem zone" of the first Mach Shock, and remains completely enclosed within it, except for the protrusion of the second Mach shock through the first Mach shock.

As the reflected shock from the second Mach Stem expands, it eventually collides with and passes through the vortex sheet from the first Mach Stem, and rolls the first triple point into a vortex, abruptly changing its trajectory so as to cause it to travel roughly parallel to the second ramp, as shown in Figure 8.54. The latter figure also shows how the first reflected shock, which remains attached to the first triple point, even after the latter is transformed into a vortex, is refracted and bent downward by the passage through it of the second reflected shock. Figure 8.54 also shows the second vortex sheet, as well as the compression and bending suffered by the first vortex sheet. The bending of the first vortex sheet is caused by the non-uniformity of the jump condition imposed along it by the passage through it of the second reflected shock. Figure 8.54 also shows the continued evolution of the subsonic flow pockets within the two reflected shocks, including the smooth flow near the leading

edges of the two ramps.

Figure 8.56 shows the computational solution after the second Mach Stem passes over the apex of the second ramp, causing the high-pressure gas enclosed within the Mach Stem to be rapidly expelled into the unconfined region behind the wedge, with a strong shock. The figure also shows the roll-up vortex formed from the interaction of the vortex sheet of the second Mach Stem with the first triple point. It should be noted, however, that the precise behavior of this feature in the computational solution is highly sensitivity to the effects of the diffusivity of the numerical scheme, whereas in reality, its behavior is highly sensitive to the viscous effects that act on the flow. Therefore, accurate prediction of the precise behavior of this feature should not be expected in an inviscid solution.

The Figures 8.50, 8.52, and 8.54 show that the computational simulation accurately reproduces every physical feature, phenomenon, and interaction of the flow field, as well as all of the wave reflection and refraction patterns that are observed in the experiment. The accuracy is observed not only in the locations and dimensions of the various features but also in their overall evolution patterns throughout the computation. At the point in time for which the experimental data in [178] is given, the computed density-ratio values are within 3% of those reported in [178], except in the region at the leading edge of the first ramp, and in the region at the leading edge of the second ramp, where the deviation goes up to about 6%, and is probably partly caused by viscous effects in the experiment that are not reproducible in the computation. The computational results obtained here are also is close agreement with the corresponding ones presented in [292].

Figures 8.49, 8.51, 8.53, and 8.55 and 8.57 present the grid plots for the corresponding density contour-line plots. Several important observations may be noted

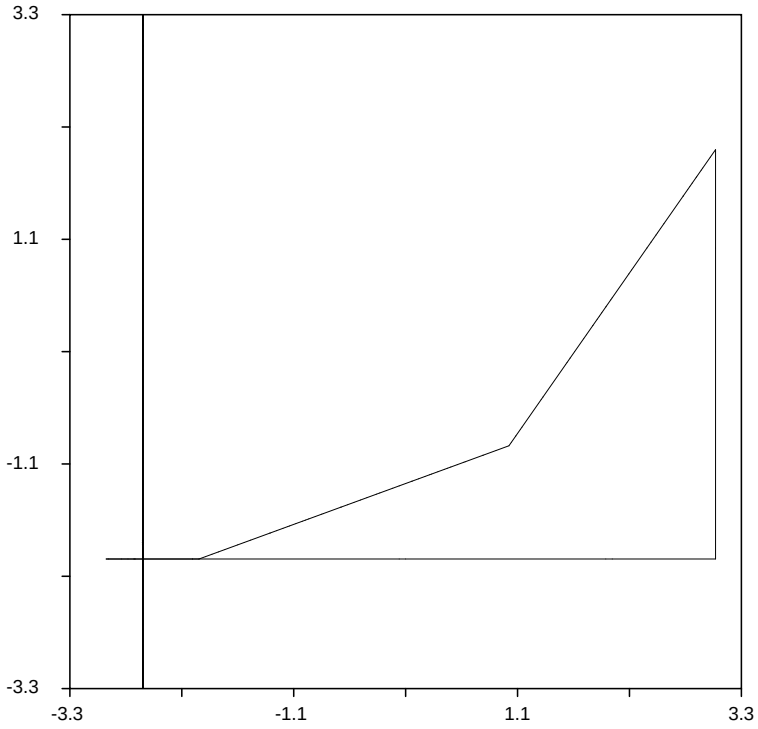

Figure 8.48: Contour-line plot of the normalized density distribution for a double-wedge shock-reflection problem at the start of the computation.

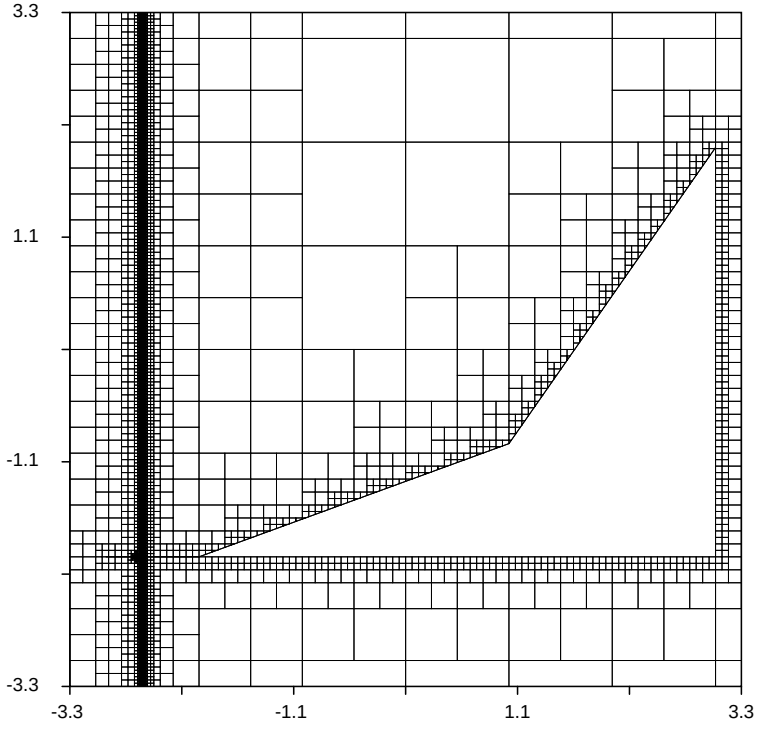

Figure 8.49: The grid plot corresponding to the contour-line plot of Figure 8.48.

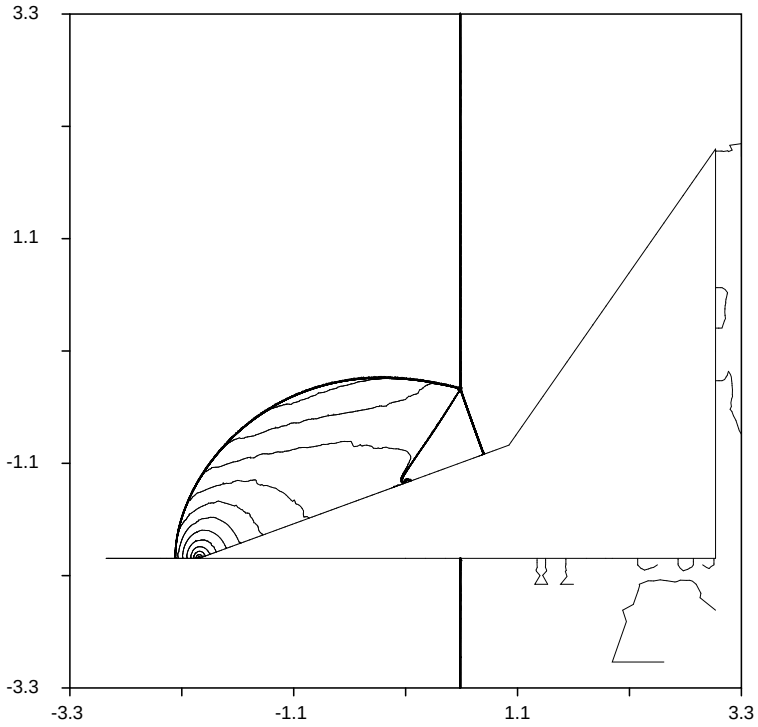

Figure 8.50: Contour-line plot of the normalized density distribution for a double-wedge shock-reflection problem shortly before the Mach Shock formed on the first ramp collides with the second ramp.

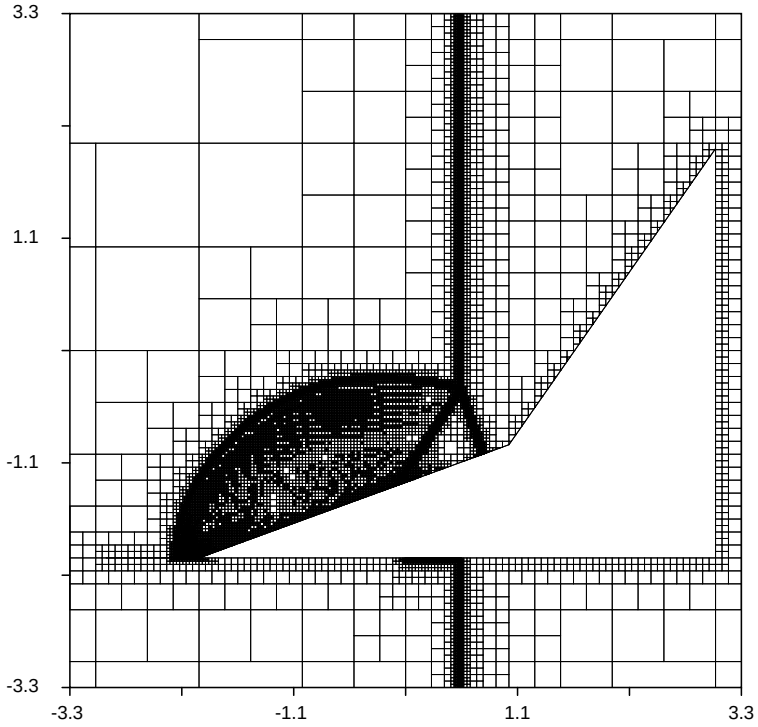

Figure 8.51: The grid plot corresponding to the contour-line plot of Figure 8.50.

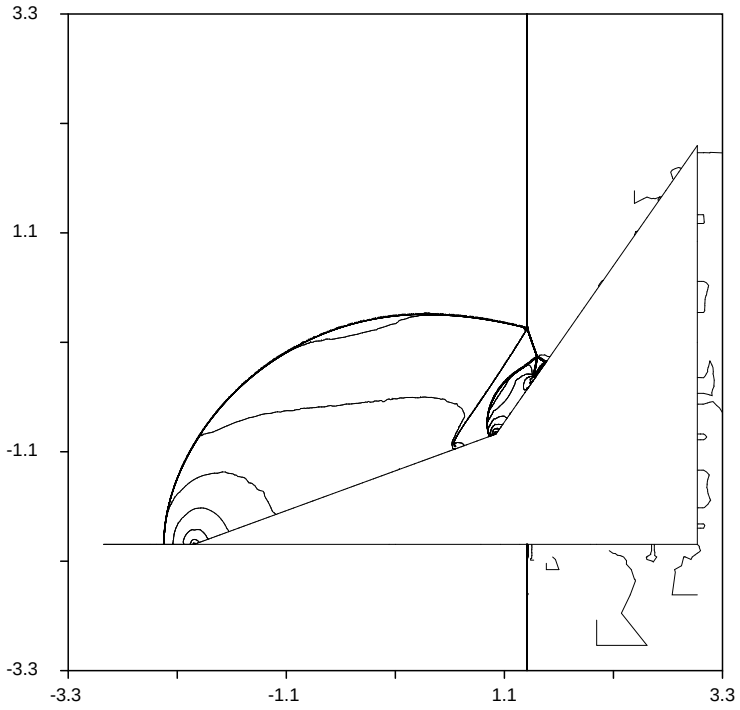

Figure 8.52: Contour-line plot of the normalized density distribution for a double-wedge shock-reflection problem shortly before the second reflected shock passes through the first vortex sheet.

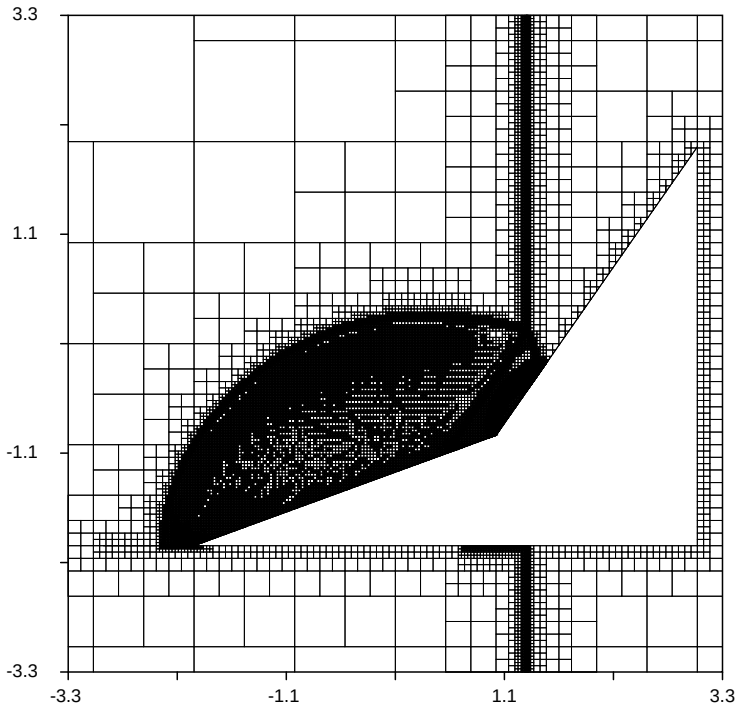

Figure 8.53: The grid plot corresponding to the contour-line plot of Figure 8.52.

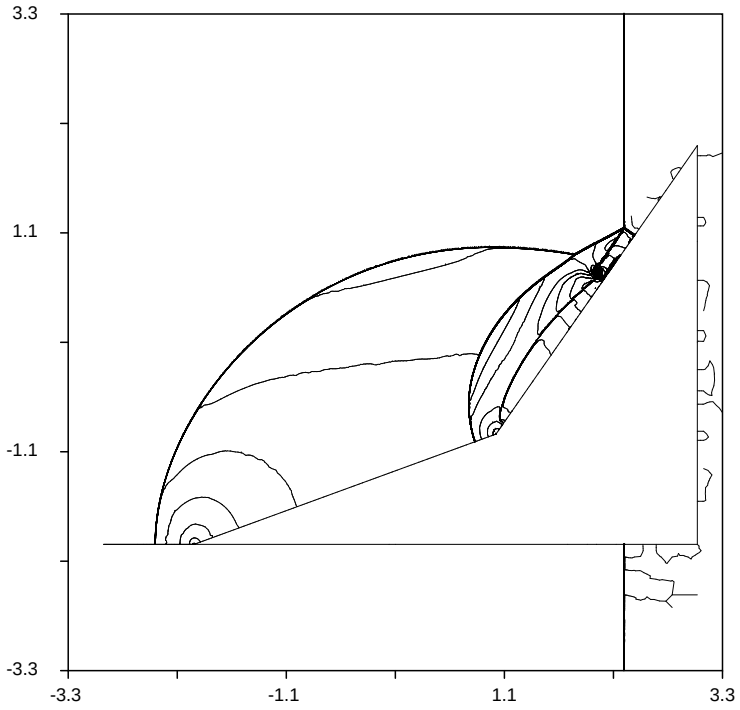

Figure 8.54: Contour-line plot of the normalized density distribution for a double-wedge shock-reflection problem after the second reflected shock passes through the first vortex sheet and the first triple point.

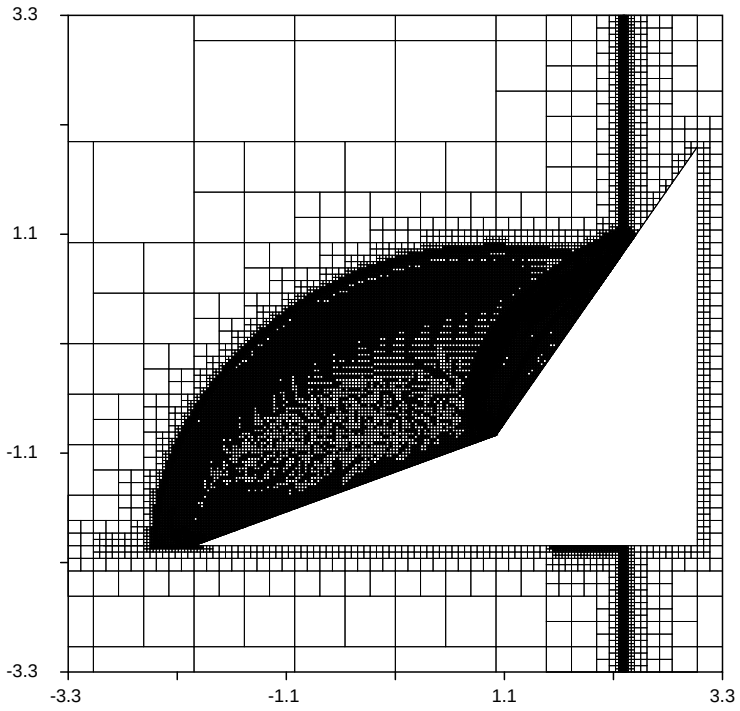

Figure 8.55: The grid plot corresponding to the contour-line plot of Figure 8.54.

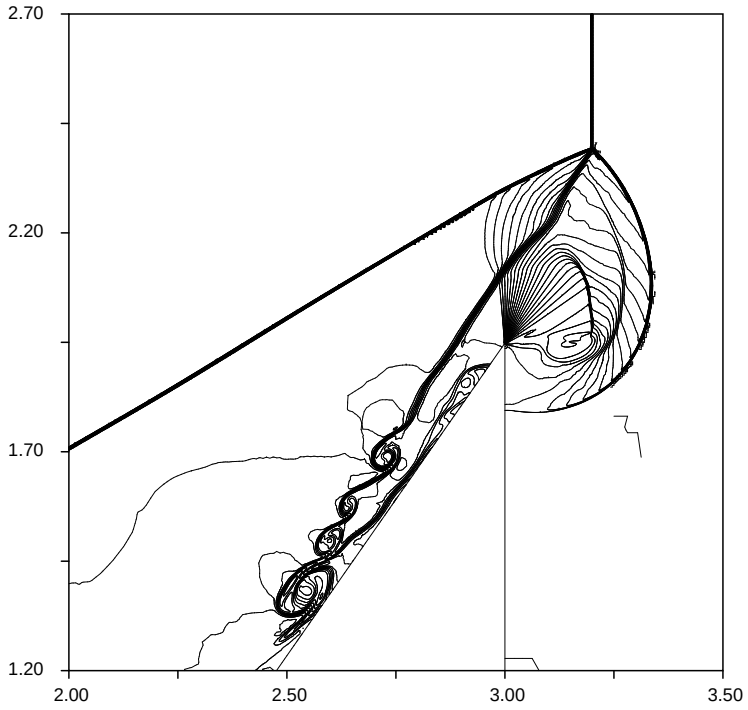

Figure 8.56: Contour-line plot of the normalized density distribution for a double-wedge shock-reflection problem after the second Mach shock passes over the apex of the double-wedge.

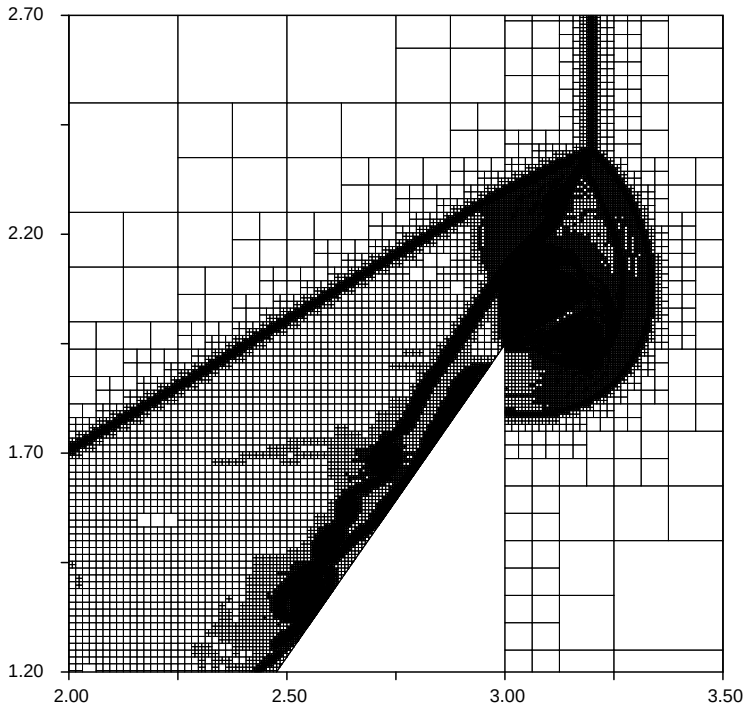

Figure 8.57: The grid plot corresponding to the contour-line plot of Figure 8.56.

for these grid plots.

The first observation that can be made from Figures 8.49, 8.51, 8.53, 8.55 and 8.57 is that the solution-adaptation mechanism adopted and developed in this work changes the grid with time not only to track the moving flow features accurately and effectively, but also to maintain them within appropriately refined zones at all times during their motion and evolution. Because the three different types of discontinuities are identified and tracked separately, as explained in Section 6.5, the solution-adaptation technique adopted and developed in this work is able to maintain different types of discontinuity within their refinement zones even as they interact with and travel through other types of discontinuity. For example, the first vortex sheet remains contained within a zone of cells at the maximum refinement level, even after the second reflected shock passes through that vortex sheet, as shown in Figures 8.52 and 8.53.

The second observation that can be made from Figures 8.49, 8.51, 8.53, 8.55, and 8.57 is related to the extent of the savings in computational resources that can be achieved by using a solution-adaptive meshing technique. In particular, if a uniformly-refined grid had been used to uniformly achieve the highest level of resolution obtained in the solution-adaptive computation presented here, the number of cells in the computation would increase by about 240 times from its time-step-averaged value to about 64 million cells.

The third observation that can be made from Figures 8.49, 8.51, 8.53, 8.55, and 8.57 is related to the absence of a grid smoothing procedure in the solution-adaptation technique adopted and developed in this work. As can be seen in the figures, this leads to the temporary formation of isolated islands of high or low refinement, which in this computation are most clearly evident in the subsonic regions enclosed between

the solid boundary and the reflected shocks. Adding a grid smoothing procedure would be highly desirable for computations of steady-state problems, as shown in [101]. For unsteady problems, the advantages of such smoothing do not clearly outweigh the additional computational cost.

Figures 8.50, 8.52, and 8.54 also show how the part of the planar shock which initially lies below the splitter plate propagates along the bottom of the splitter-plate and double-wedge with no visible change or distortion. This confirms the overall accuracy of the boundary treatment at impermeable surfaces. The corresponding grid plots of Figures 8.51, 8.53, and 8.55, however, show that a small "streak" of grid refinement follows the shock wave along the bottom surface of the wedge. This streak appears in response to small fluctuations in the solution (measuring about 0.5% in the value of the density, for example) behind the shock wave along the bottom edge of the double-wedge. These fluctuations appear to be caused by the incomplete vanishing of the y-component of the reconstructed gradients in the immediate vicinity of the bottom of the double-wedge, probably because the limiter used in all three test cases presented in this sub-section is set to be slightly "compressive", and is therefore prevented from completely suppressing these components.

The non-closed contour lines visible in Figures 8.50, 8.52, 8.54, and 8.56 appear to be caused by an idiosyncrasy of the plotting package used to draw the contour-line plots. However, a possible contributor to their appearance is that the solid boundary in this problem executes a small oscillation during every time-step, emitting density perturbations (having an approximate relative magnitude of 10^{-7}) into the flow-field. These oscillations occur because the boundary of the double-wedge is identified as a moving boundary (having zero velocity). This in turn causes the locations of the control points to be perturbed by very small distances once every time step, in order

to maintain the coefficients of the cubic term in the almost straight sections of the spline interpolation polynomial within acceptable values, as explained in more detail in Chapter V. Identifying a stationary boundary as a moving-boundary with zero velocity was commonly done throughout this work in order to test the robustness of the algorithm just described for perturbing the control points of the Composite Parametric Cubic Spline interpolant.

The results shown and discussed above are from the computation with the highest maximum refinement level attempted for this problem; namely, 13. The computations performed with lower maximum refinement levels; namely, 12, 11, and 10, also give correct and accurate solutions, but do not fully resolve all the flow features in the problem, especially the ones that are sensitive to the level of numerical dissipation, such as the roll-up vortex shown in Figure 8.56.

The computation presented above is one in a series of computations performed in this work with shock diffractions over single-wedge and double-wedge obstacles with differing angles and shapes. The purpose of performing all the computations in the series was to reproduce much of the range of shock and shock reflection types and configurations in the classification given at the beginning of this sub-section, and described in detail in [41]. The results from two other computations in this series (involving shock-wedge reflections), with the grids omitted for brevity, are shown below.

Figure 8.58 shows a contour-line plot of the computed density for the problem of the diffraction of a shock wave of Mach Number 2.16 over a convex double-wedge with ramp angles of 60 and 30 degrees. Because the corner of the double-wedge is concave, the Mach Stem formed from the collision of the incident shock with the first ramp does not collide with the second ramp to generate a second Mach Stem.

Instead, the Mach Stem changes its orientation as it travels over the corner and emits a rarefaction fan which is approximately centered about the corner of the double-wedge in the process, as shown in Figure 8.58. The rarefaction fan bends the reflected shock downwards, toward the wedge, as shown in the figure.

The density-ratio contour-line plot of Figure 8.58 can be compared directly with the corresponding experimental results presented in [178]. The point values in the computational results are everywhere within 5% of the corresponding experimental values given in [178]. The computational simulation can again be seen to accurately reproduce all the flow features observed in the experiment, and their evolutions and interactions with each other, including, for example, the size and uniformity of the properties within the triangular region enclosed by the Mach Stem and the surface of the wedge, and the bending of the reflected shock towards the wedge by the expansion fan. The maximum refinement level, 13, was the same as that for the double-wedge case discussed above in this sub-section. The effectiveness of the solution adaptation in reducing the number of cells is analogous to that explained above for the first double-wedge test case presented in this sub-section.

Figure 8.59 shows a contour-line plot of the computed density for the problem of the diffraction of a shock wave of Mach Number 5.5 on a 30 degree single-ramp wedge, leading to the formation of a Double Mach Reflection. As shown in Figure 8.59, a splitter plate was not used in this test case. The maximum refinement level used in the computation was 11, which is two levels below that used for the other two cases presented in this sub-section.

Figure 8.59 shows that all the characterizing flow features present in the physical problem are again adequately resolved, except for the vortex sheet originating in the second triple point. This vortex sheet is too weak to be visible in the density

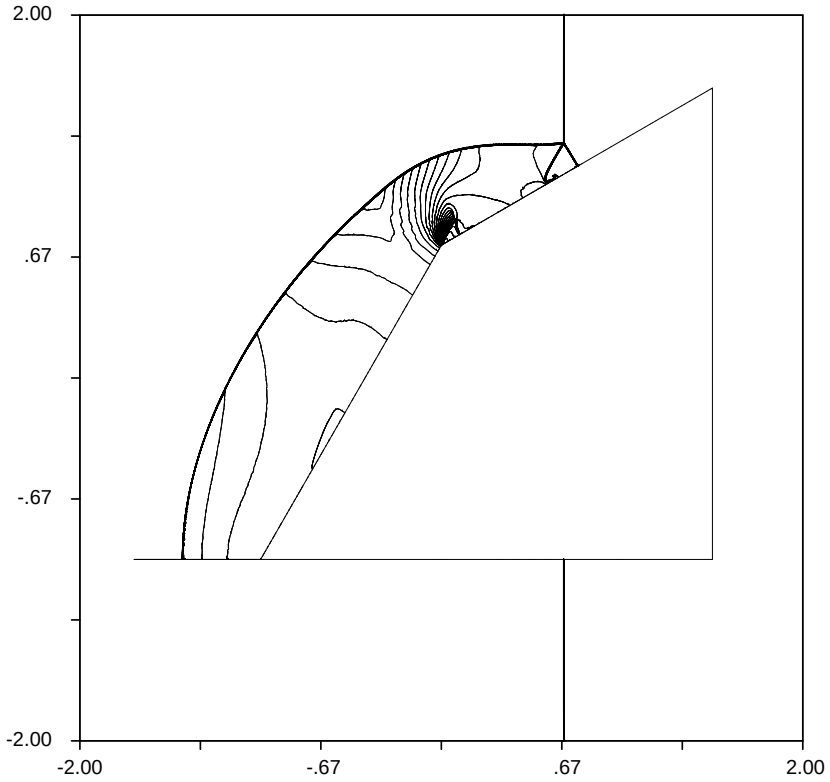

Figure 8.58: Contour-line plot of the normalized density distribution resulting from the collision of a shock wave of Mach Number 2.16 with a double-ramp wedge of ramp angles 60 and 30 degrees, after the Mach Shock has passed the convex inflection and re-generated itself on the second ramp.

contour-line plot, but can be seen to varying degrees of clarity in the contour-line plots of, for example, s, M, u, v, or $q = \sqrt{u^2 + v^2}$, even at the relatively coarse maximum refinement level used in the computation. It is also detected and captured by the solution-adaptation algorithm, and is correctly confined within a zone of cells refined to the maximum refinement level used in the computation. This vortex sheet is apparently so weak that it also cannot be visually observed in the corresponding experimental holographic interferogram. The remaining features which are all resolved include the incident and reflected shocks, the first and second Mach shocks, the vortex sheet originating in the first triple point, and the jet of high-pressure gas originating near the foot of the latter vortex sheet. This jet is formed as a result

of the expulsion of the high-pressure gas behind the second Mach shock, along the surface of the wedge, into the lower-pressure region behind the first Mach shock. The contact wave bounding the jet is also adequately resolved. The figure also shows how the second Mach shock becomes weaker as it extends towards the vortex sheet of the first Mach shock, just as observed in the corresponding experimental results.

The computational results obtained for this test case may be compared directly with those presented in [291] or [293], with the related computational results presented and discussed in [414], or with the related experimental results presented and discussed in [41].

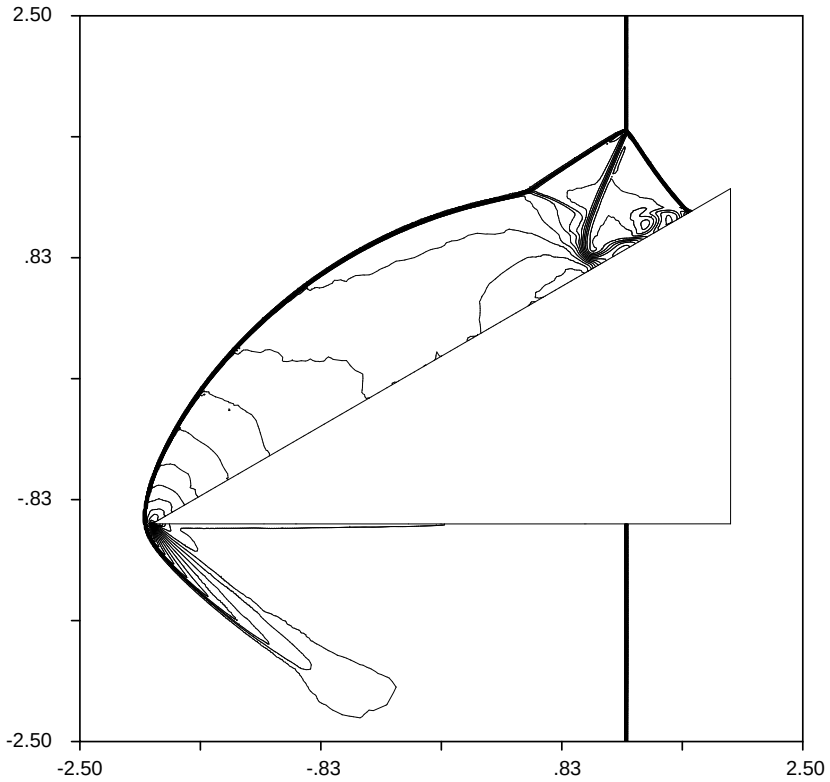

Figure 8.59: Contour-line plot of the normalized density distribution resulting from the collision of a shock wave of Mach Number 5.5 with a 30-degree, single-ramp wedge, showing the formation of a Double Mach Reflection.

The computations for all three test cases presented in this sub-section were per-

formed with the Roe and HLLE numerical flux functions, with the latter being strictly confined to regions occupied by a strong shock, as described in Chapter III. Also for all three test cases, the gradient reconstruction was performed using the Least-Squares procedure, and the limiting was performed using the Venkatakrishnan Limiter described in Chapter III. The time-integration was performed using the second-order Predictor-Corrector Method described in Section 3.3.

The use of the HLLE numerical flux function was found to be necessary for all three of the test cases presented in this sub-section to prevent the formation of a "kink" in Mach shocks, and to prevent odd-even decoupling behind the incident shocks. In fact, a major motive for studying the specific single-wedge shock-reflection problem depicted in Figure 8.59 was to test and evaluate the effectiveness of using a combination of HLLE and Roe numerical flux functions, following the suggestion in [293], to eliminate the "kink" in the Mach shock that is observed with Mach Reflections, while still maintaining a low overall numerical diffusivity. The specific test case of Figure 8.59 exhibits the bending of the Mach shock to a strong extent, even on relatively coarse grids, and therefore presents a useful test case for the given purpose. It was found that in order to fully prevent the bending of the Mach shock in this test case, it was necessary to widen the region around the incident and Mach shocks within which the HLLE flux function is used to at least 4 or 5 cells, so that it envelops almost the whole of the thickness across which the shock wave jump occurs. The work done with this test case largely corroborated the usefulness and effectiveness of the approach of combined numerical flux functions suggested in [293].

8.4.3 Diffraction of a Shock Over a Half-Diamond Wedge

This sub-section presents and discusses the computational-simulation results obtained for the diffraction of a shock wave with Mach Number 2.44 over a rigid, half-diamond-shaped wedge. The wedge is symmetric about its vertical center-line, and its inclined faces subtend angles of 45 degrees to the horizontal, as shown in the relevant figures below. The wedge is attached to a horizontal splitter-plate of infinitesimal thickness, in order to completely separate the flows below and above the wedge, as was done for two of the test cases presented in the preceding sub-section.

The primary purpose of this computation is to validate the computational solution algorithm, and to examine its overall accuracy. This is done by qualitatively comparing the evolution of the flow features and the critical flow structures with the corresponding experimental observations [428], by quantitatively comparing the density ratios with the corresponding experimental results [428], and by comparing the computational results with those obtained with different algorithms, for example, as in [372], and [233]. The computational results are presented in the form of density contour-line plots, for the same reasons that were given in the preceding sub-section, and all contour-line plots in this sub-section are drawn with 50 contour lines.

Because of the convexity and the abruptness of the change in angle at the apex of the wedge, as well as the relatively high Mach number of the incident shock, the flow in this problem creates states close to a vacuum, and exhibits all the main types of discontinuities supported by The System of Euler Equations, as described in Chapter II, and with several of them interacting with each other simultaneously.

The computation is started with the incident shock moving from the left toward the wedge. A Mach Stem is immediately formed upon contact of the incident shock with the ascending (left) ramp of the wedge, and rapidly develops into a Double Mach

Reflection. The structure and properties of Double Mach Reflections are described in detail in [41]. In this particular case, the reflection configuration results in four different regions within the reflected structure, each of which is bounded either by discontinuities or by solid surfaces: (i) a triangle-shaped region of uniform properties bounded by the first Mach shock, the wedge ramp, and the first vortex sheet (which extends from the first triple point to the wedge ramp); (ii) a second three-sided region of smoothly varying properties bounded by the first vortex sheet, the first reflected shock, and the second Mach shock; (iii) a second region of smoothly-varying properties bounded by the second Mach shock, the wedge ramp, and a curved vortex sheet (which extends from the second triple point to the wedge ramp); and, (iv) a region of smoothly varying properties bounded by the second vortex sheet, the second reflected shock, and the solid walls of the splitter-plate and the wedge ramp. This reflected structure can clearly be seen in Figure 8.60, except for the second vortex sheet, which is too weak to be visible in a density contour-line plot, but can be seen in a contour-line plot of either of the two velocity components or the velocity magnitude, and is detected by the solution-adaptation algorithm, as can be seen in the corresponding grid plot of Figure 8.61.

As the Double Mach Reflection system ascends the ramp of the wedge, the four regions described above enlarge and change their shapes slightly in the process, except for the triangular region attached to the first triple point, which maintains its triangular shape and the uniformity of its properties throughout its enlargement. This behavior is in agreement with analytical and experimental studies discussed in [41]. Figure 8.60 shows point values at several points for the density ratio, and these values may be compared directly with the experimental results given in [428]. In most cases, the agreement is within 2% percent of the corresponding value given for

the experimental results.

As the first Mach Stem passes over the apex of the wedge, the flow behind the Mach Shock bursts almost horizontally into the region immediately above the downward ramp, causing separation from the downward ramp, and creating a strong slip line (or vortex sheet) radiating almost horizontally from the apex to the right. Above the apex, an expansion fan is also created that fans a region of approximately 90 degrees, ranging from the slip line just described to the upper parts of the Mach Stem. Close to the apex of the wedge, the Mach number in the expansion fan reaches about 4.7 during the later stages of the computation. As would be expected, the expansion fan is a persistent structure, and it gradually bends the Mach shock, and eventually, the second reflected shock as well. The first vortex sheet developed and elongated during the ascent of the first Mach Stem separates from the apex after it reaches it, and starts forming a roll-up vortex which travels roughly along a particle path for the remainder of the computation. The second Mach shock and vortex sheet continue to expand in size, but do not undergo any other major transformations apart from their interaction with the expansion fan. The flow downstream along the slip line formed by the separation of the flow from the apex forms a shock, as well as a strong vortex with several internal and surrounding structures, including vortex sheets and shock waves.

All the structures and patterns described in the preceding paragraph are clearly visible in Figure 8.62, which shows close agreement with the corresponding experimental results of [428]. In particular, very close agreement can be observed for the geometry, size, and strength of the expansion fan emanating from the apex and the geometric effects of its interaction with the major reflected shocks. Similarly, the slip lines, the roll-up vortices, and the entrainment regions that are formed after the

separation of the flow from the apex of the wedge appear to be reproduced accurately in the computation. This is not surprising because although this is an inviscid-flow calculation, the vortical phenomena here are pressure-induced rather than viscosity-induced. The density values at the later stages of the computation shown in Figure 8.62 (and beyond) are again in agreement with the experimental results to within 1-4% at all points for which experimental data is reported.

The flow around the main separation vortex rotates clockwise throughout, and gradually replenishes the low pressure region immediately below the separation slip line. In fact, the pressure in that region is close to zero (just as it is inside the main vortex), while the Mach number there is about 0.01, and a strong shock (with Mach number varying roughly between 2 to 3 for most of the computation after occurrence of the major separation) is formed by the flow moving upward along the downward ramp into that region. The strongest shock in the entire flowfield, however, is the one formed immediately ahead of and above the main vortex, inclined at an angle of about 45 degrees. The strength of the shock around its root is about 3.7 at the time corresponding to the solution shown in Figure 8.64.

In order to show the shear and contact structures in the vicinity of the main vortex, a contour-line plot of the total-velocity is presented in Figure 8.64. The figure shows the strong shock described in the preceding paragraph, and also clearly shows the existence of several vortex sheets formed during the creation of the separation slip line and its interaction with the strong shock described above. Agreement of the behavior of the complex pattern of secondary shocks, vortex sheets, and the roll-up vortices shown in the Figure 8.64 with the corresponding experimental results appears to be very good, although some of the features described above are barely identifiable in the images of the experimental results.

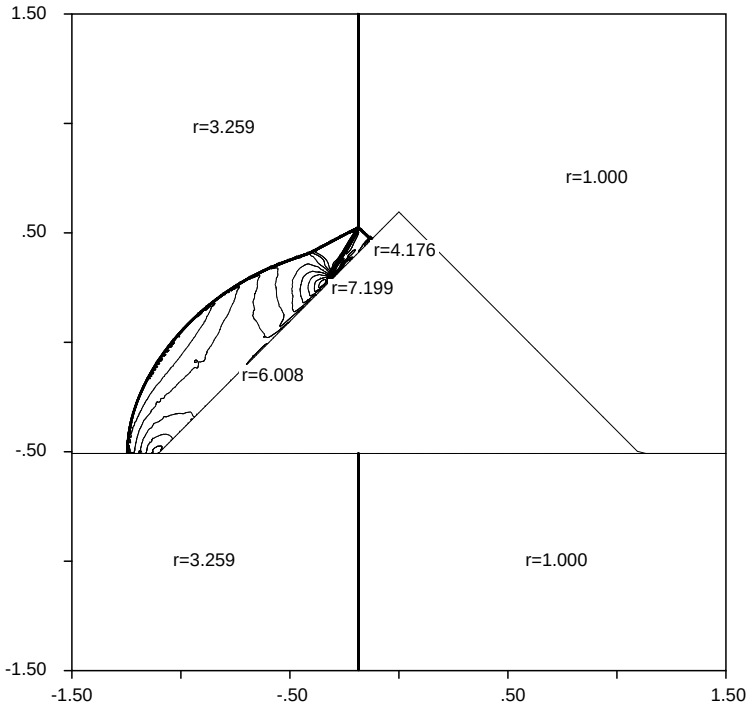

Figure 8.60: Contour-line plot of the normalized density distribution for a half-diamond shock-reflection problem, before the first Mach Stem reaches the apex of the half-diamond, showing key point values in the flowfield.

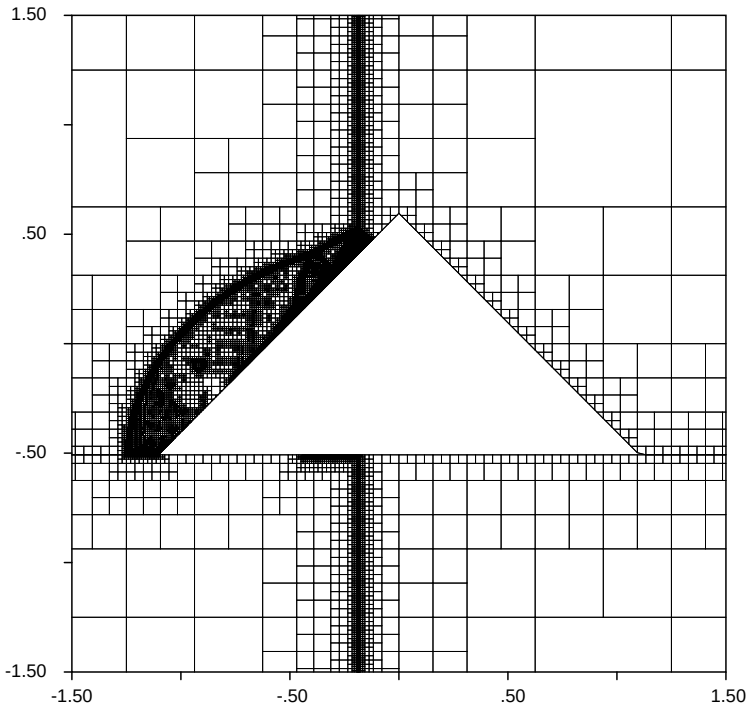

Figure 8.61: Grid plot corresponding to the contour-line plot of Figure 8.60.

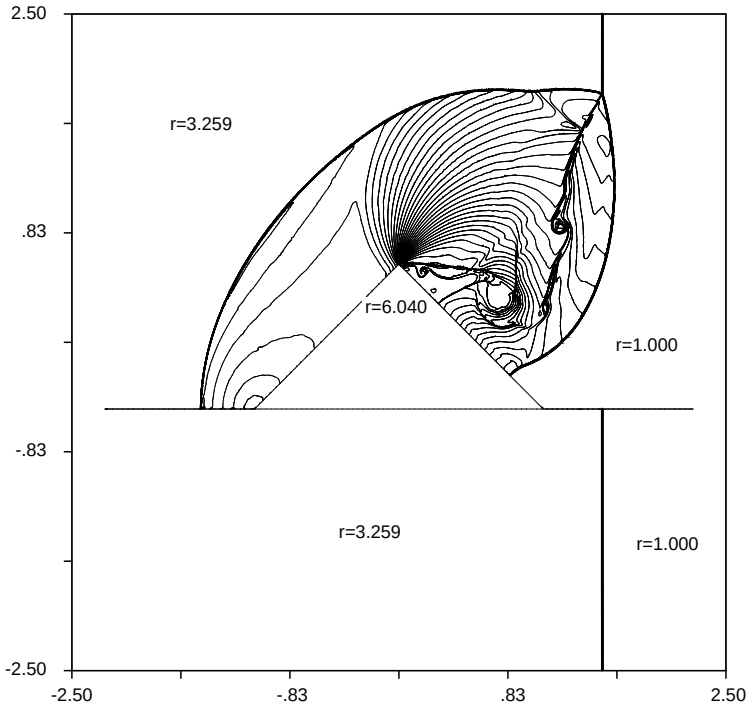

Figure 8.62: Contour-line plot of the normalized density distribution for a half-diamond shock-reflection problem, after the Mach and incident shocks have passed over the apex of the half-diamond, showing key point values.

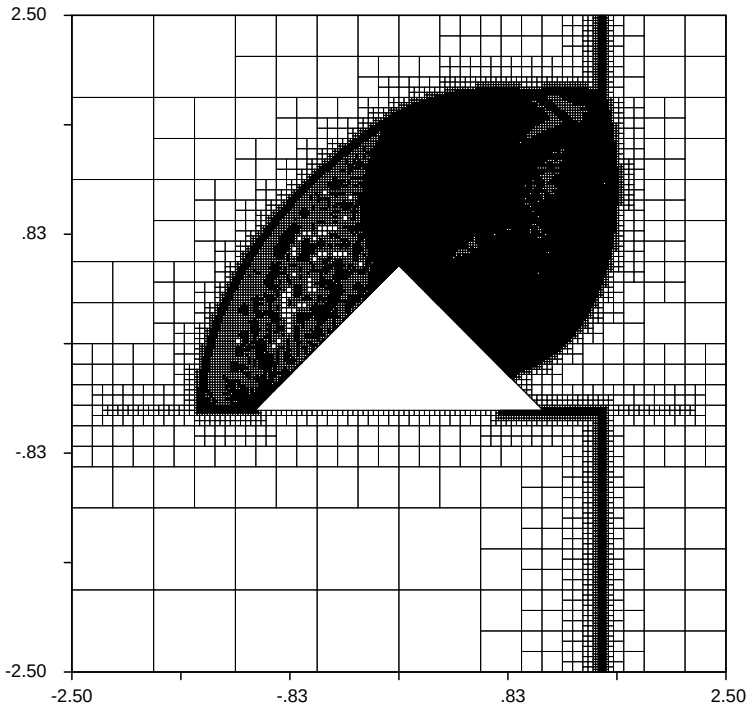

Figure 8.63: Grid plot corresponding to the contour-line plot of Figure 8.62.

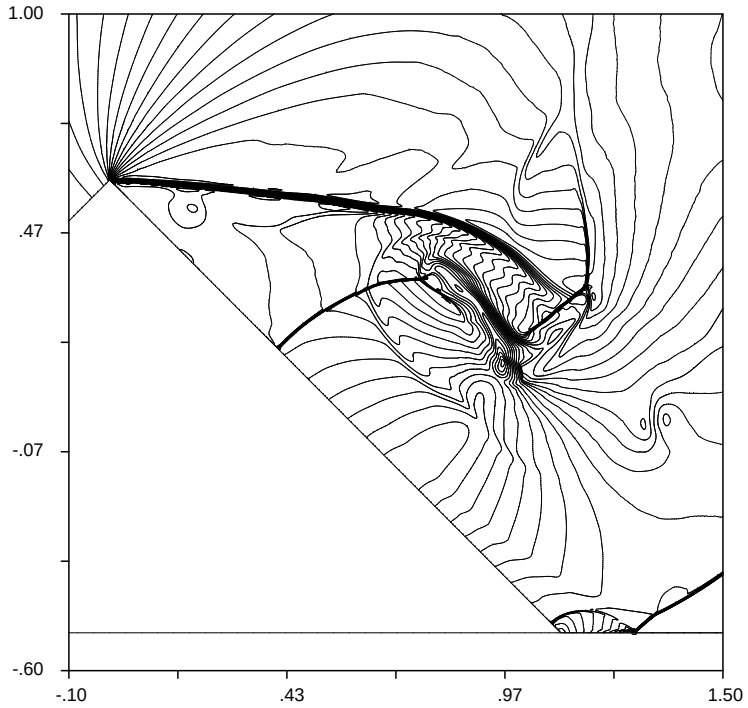

Figure 8.64: Contour-line plot of the total velocity for a half-diamond shock-reflection problem, showing the expansion fan and main vortex, after the incident and Mach Shocks have overtaken the apex of the half-diamond.

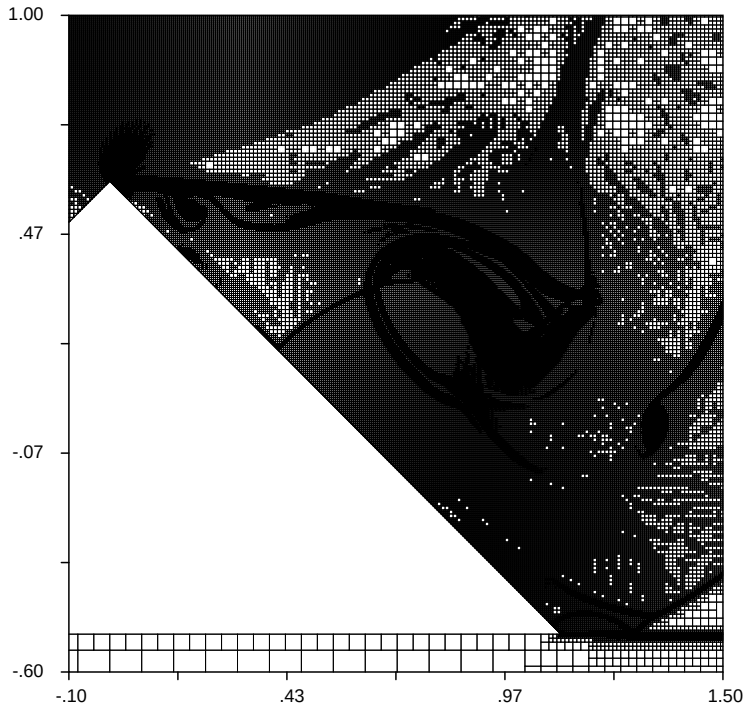

Figure 8.65: Grid plot corresponding to the contour-line plot of Figure 8.64.

Figures 8.61, 8.63, and 8.65 show the grid plots corresponding to the contour-line plots discussed above. Again, these grid plots clearly reveal how the adaptation algorithm is able to accurately track individual features as they travel, evolve, and interact with other features throughout the Computational Region, and how the algorithm is able to maintain these features within refinement zones of appropriate resolution and width.

The computational results presented and discussed above also agree to within about 7% or so with those presented in [372], and agree with the related results presented in [233], although the ones shown here are far more resolved than those of [372], and slightly more resolved than those of [233], especially in regions of shear waves, such as the vortex sheet of the first triple point. The comparison in resolutions was arrived at as follows: at each stage of the computation, the number of vertices reportedly used in [233] is about 1/3 as many as the number of cells used in the present computation, and since the ratio of cells to vertices for triangular meshes is roughly 2, basing the comparison on the number of cells, it can be seen that the present computation has about 50% more cells than the one reported in [233] during most of the computation.

In performing the computations for this problem, four different maximum refinement levels were attempted. It was found that the minimum cell size seems to be more important than total number of cells used. If the number of cells is halved while the maximum refinement level is maintained, there is little apparent deterioration in the quality or the resolution of the solution, while if the maximum refinement level is reduced by one, the quality and resolution suffer dramatically, even if the total number of cells simultaneously increases. This was also observed in other problems which contain several interacting, and spatially close features that must be resolved

fully to retain an accurate evolution history of the overall flowfield. A difficult task in this simulation was to set the various adaptation parameters so that they are sufficiently sensitive to all the types of features present in the computation, while maintaining the total number of cells below 250,000.

Because of the cavitation effect described above, and because of the need for very low numerical dissipation (in order to resolve the shear and contact layers as accurately as possible), and because of the need to prevent oscillations and odd-even decoupling behind strong shocks, and in order to prevent bending of Mach shocks, the fluxes for this test case were computed using the Exact Riemann Solver in combination with the HLLE numerical flux function, as described in detail in Chapter III. The gradient reconstruction was performed using the Least-Squares procedure, and the limiting was performed using the modified Barth Limiter described in Chapter III. The time-integration was performed using the second-order Predictor-Corrector Method described in Section 3.3.

8.4.4 Diffraction of a Shock Over a Cylinder

The test case presented in this sub-section studies the diffraction of a planar shock wave with Mach number 2.82 against a stationary cylinder of diameter 1 unit. The primary purpose of this test case is similar to that of the test case of the preceding sub-section; namely, validation against the corresponding experimental results, here given in [61], and investigation of the overall accuracy of the computational methodology. An important difference between the test case of this sub-section and the one of the preceding sub-section is related to the "types" of flow separation that occur: the test case of this sub-section involves separation of flow from a smooth, curved surface in an adverse pressure gradient, while the test case of the preceding

sub-section involves separation of flow from a sharp edge. Special attention in this test case is therefore focused on the accuracy of the computational predictions for the locations and times at which the various flow separation events are observed.

The planar shock wave propagates from left to right, first making contact with the cylinder at its right leading-edge. A Regular Reflection is formed immediately upon contact, with the reflected shock system propagating along the surface of the cylinder and to the left against the local flow direction. As the incident shock sweeps forward along the x-axis, and as the reflected shock system propagates further along the surface of the cylinder, the ramp angle presented to the incident shock continuously decreases, causing the Regular Reflection to transition to a Double Mach Reflection, then to a Transitional Mach Reflection, and then to a Single Mach Reflection, as explained in [61] and [41]. The first transition occurs in this case when the incident shock reaches roughly the 45 degree point along the surface of the cylinder from the leading-edge. By about 50 degrees, the Mach Reflection and the Mach Stem of the Double Mach Reflection are clearly identifiable. The final transition to a Single Mach Reflection occurs around the "70-degree" location.

As the Mach Stem moves further along the surface of the cylinder after the formation of the Single Mach Reflection, the Mach shock remains attached to the surface, but the vortex sheet from the Mach Stem separates from the cylinder, to form a free roll-up vortex. This occurs shortly after the incident shock reaches the 90 degree point. The roll-up vortex approximately follows a particle-path trajectory thereafter, which here turns out to be closely approximated by a straight line, with a slight downward slope, roughly tangential to the surface of the cylinder at the point of separation of the vortex. Further on in time, a strong expansion is created as the flow behind the Mach shock follows the diverging path on the surface of the

cylinder beyond the 90-degree point, and a strong shock (with Mach number 3.87 in the position shown in the second of the sequence of three sets of figures presented for this test case) is formed within the flow behind the Mach shock.

The two Mach shocks which form on either side of the cylinder, as described above, eventually collide with each other at the 180-degree position, creating a Regular Reflection which begins to expand downstream with the flow. Later on, this Regular Reflection transforms into two Mach Reflections, with their accompanying Mach Stems and triple points. The Mach Reflections start to form from the Regular Reflection at a distance of about 0.8 units behind the trailing-edge of the cylinder. The two newly-formed triple points are connected by a single, strong, vertically-aligned, reflected shock. The structure containing the two connected Mach Stems continues to move to the right, away from the cylinder, while the large-scale separation zone formed after the collision of the first two Mach shocks continues to develop and expand behind the cylinder. The reflected shock system from the collision of the two Mach Stems at the 180 degree position eventually expands far enough to engulf the roll-up vortices created from the separation of the first set of triple points from the surface of the cylinder, and to interact with the vortex sheets of the first pair of Mach shocks.

The computational solutions at three different times are shown in the density contour-line plots of Figures 8.66, 8.68, and 8.70, with all three plots being drawn with 50 contour lines.

Inspection of the results shown in Figures 8.66, 8.68, and 8.70, and their comparison with the corresponding experimental results given in [61], and the related experimental results given in [41], shows that all the key features of the physical problem are reproduced accurately in the computation, and that the computational