An Algorithm to Align a Quadrilateral Grid with Internal Boundaries

James. M. Hyman[1] , Shengtai Li[1,2] , Patrick Knupp[3] and Mikhail Shashkov[1]

[1]Theoretical Division
Mail Stop B284, Los Alamos National Laboratory
Los Alamos, NM
87545

[2]Department of Computer Science
University of California, Santa Barbara
Santa Barbara, CA 93106

[3]Parallel Computing Sciences Department
Mail Stop 0441, Sandia National Laboratory
Alberquerque, NM 87185-0441

September 21, 1999

Abstract

We will present a new numerical algorithm that aligns a quadrilateral grid with internal alignment curves (IACs). These IACs can be used to delineate internal interfaces, discontinuities in material properties, internal boundaries, or major features of a flow field. The IAC grid generation algorithm readjusts a predefined reference grid to create a nearby grid where the mesh cell edges are aligned with the IACs. On an aligned grid, numerical discretizations of partial differential equations can be formulated to satisfy the interfacial relations, such as matching fluxes across the discontinuity, to reduce the numerical errors introduced by the discontinuity. We present examples to demonstrate the effectiveness of the IAC grid generation algorithm for multiple imbedded interfaces in a quadrilateral grid.

1 Introduction

When numerically approximating physical systems with discontinuous coefficients, often the largest numerical errors are introduced in a neighborhood of the discontinuities. These errors are often greatly reduced if the grid is aligned with the discontinuities and special formulas are used to incorporate the jump conditions directly into the numerical model. For example, when solving the equations governing the conservation of mass, momentum and energy in multimaterial or multiphase flows, such as a liquid-gas or liquid-solid interfaces where the normal and tangential stresses must be matched at the interface or the equations of state are drastically different at an interface. When modeling the strain and stress of materials and the coefficients are discontinuous, then the numerical solution is often extremely sensitive to the proper alignment of the control volume with the boundary. In the numerical approximations of wave equations, discontinuities (e.g., the tensorial dielectric constants in Maxwell's equations or the tensorial stiffness tensors when solving general plasticity models) can introduce spurious errors and reflections at an interface boundary unless the boundary is aligned with the discontinuity [7].

The rate and direction of underground flows predicting the extent of contamination and environment danger posed by subsurface flows from hazardous waste sites is governed by discontinuities in the geomorphology of the flow field. Finite difference approximations are more accurate when the underlying grid is aligned with the discontinuities to minimize the heterogeneity within a grid cell.

Although grid generation [11, 15] is the heart of most numerical algorithms for heterogeneous regions, there has been little attention to automated alignment of with the internal boundaries. Most of the proposed methods have been to locally align a grid with interfaces and sharp gradients in the solution [1, 3, 4, 8, 9, 10, 12, 13, 16, 18] or to explicitly treat discontinuities as an immersed interface in a grid and locally adjust the difference methods to accurately account for any discontinuities [5, 17]. The GEGA [3], directional control [1, 4], and Jacobian-weighted [9, 10] methods can generate smooth but only qualitatively aligned grids. The GAG [13] alignment algorithm can generate an aligned grid but it can be extremely rough and irregular. Because the accuracy of finite difference approximations is related to the smoothness of the grid, it is essential to generate an aligned grid which is smooth.

We will describe a new approach where after delineating the discontinuities with internal alignment curves (IACs) and generating a boundary fitted quadrilateral reference grid, this reference grid is rearranged so the grid cell edges are locally aligned with the IACs. The IACs can be defined to delineate internal interfaces, discontinuities in material properties, internal boundaries, or major features of a flow field. The new grid then captures these discontinuities and will allow numerical discretizations of PDEs to directly incorporate the interfacial relations, such as matching fluxes across the discontinuity, at the grid cell edges and reduce the numerical errors introduced by the discontinuity.

The IAC algorithm is motivated by the observation that in many applications with internal discontinuities, the computation domain can be split into several homogeneous smaller domains by the IACs. An alternative to the IAC approach would be to split the physical domain into separate pieces, generate a separate grid for each piece and have the separate grids communicate with each other through the interfacial boundary conditions. There are advantages to having a single quadrilateral grid that imbeds the discontinuities as generated by the IAC versus the flexibility (but added complexity) of having several component grids, each with it's own data structure. For the most complex problems, probably the ideal grid would be a combination of the two approaches. In this paper, we will only consider the case of a single quadrilateral grid with the grid aligned with the internal discontinuities.

After describing the IAC algorithm for the simplest case of aligning the grid with IACs that span between opposite sides of the domain boundaries, we describe the more difficult IAC algorithm needed to align the grid with the imbedded quadrilaterals. We then combine the two algorithms to illustrate the robustness of the approach in a wide range of examples with internal discontinuities.

2 Grid Alignment for Spanned IACs

When describing the algorithm, it is useful to classify the type of IACs according to whether or not it describes an region enclosed within the domain. In this section we will describe the algorithm for when the discontinuities can be delineated by *spanned IAC* (SIAC) that connects opposite sides of the domain boundaries. These IACs are appropriate in layered domains when approximating flows in underground flow fields, modeling the dynamics of layered materials, or the interface dynamics between two fluids. Note that although a discontinuity may not stretch across the entire domain, it can be delineated by a spanned IAC by extending the IAC from the discontinuity to the boundaries. In our algorithm, we approximate a SIAC by a sequence of (possibly curved) line segments.

In this section we will describe the IAC algorithm for SIACs. In the example shown in Fig. 2.1-a, there are two horizontal SIACs (H_0 and H_1) and one vertical SIAC (V_0). These consists of a sequence of linked line segments (e.g. H_1 has line segments connecting the points H_{10}, H_{11}, H_{12}, and H_{13}). Note that the line segments need not be a straight line; the segment between H_{11} and H_{12} is the arc of a circle. We assume that horizontal (vertical) SIACs do not cross other horizontal (vertical) SIACs. If a line segment is shared by both a horizontal SIAC and a vertical SIAC, it can easily be transformed to a case where it is not by adding more line segments. Therefore, for simplicity, we require that none of the line segments are shared by both horizontal and vertical SIACs.

In addition to the IACs, the algorithm requires an initial (not necessarily aligned) boundary fitted grid as in Fig. 2.1-b. The initial grid can be generated by any standard grid generation algorithm based on, say, transfinite interpolation [15], an elliptic grid generator [11] or the divide and conquer [2] algorithm.

The alignment process begins with the simple assignment of a single horizontal grid line to each of the horizontal SIACs and a vertical grid line to each of the vertical SIACs.

To select the appropriate grid line to align with the SIAC, we will describe the process for a horizontal SIAC. First, for each vertex of the SIAC, find the point of the non-aligned grid which is closest to the vertex. Next take the average of the vertical indices of all of the closest non-aligned points and round to define vertical index of the closest horizontal grid line to the SIAC. After doing this for each vertical and horizontal SIAC, the intersection points between two SIACs have unique grid indices (logical coordinates) in each direction. The location of the grid points between any two SIAC vertices are then uniformly distributed (default) or distributed by the virtual function defined by the user. The original distribution of the grid points can be preserved by computing the intersections between the SIAC and grid lines and then redistributing them with inverse interpolation [6]. On a curvilinear grid this should be done in physical space instead of grid index (logical). If a SIAC crosses a sparse region and a dense region, an alternative way to preserve the distribution is to split the SIAC into two SIACs, one in sparse region, one in dense region.

The assignment of the reference grid line to the SIAC may result in grid points crossing on the domain boundaries and these points must be regularized while retaining their relative positions. Linear interpolation is usually sufficiently accurately define new grid locations along smooth boundaries but it can introduce large errors near kink points or internal corners. A kink is a specified point on an IAC or boundary where the slope of the curve is discontinuous (e.g., point H_{11} in Fig. 2.1-a). We handle the kinks in the boundary by not restricting the grid point movement during the redistribution and then moving the boundary points nearest to the kinks to the kinks location. This approach is also used in the divide and conquer grid generation method [2].

We regularized the boundary grid points by first using the original distribution of the original grid points on the boundary as a guide for the density of mesh points along a boundary line. We use inverse interpolation between the intersection points of the SIACs with the boundaries to redefine the grid points on the boundary so that they match the relative distribution of the original reference grid. The inverse interpolation is done in grid index space by assuming the grid spacing between the original grid points is uniform. The distribution and location of the new grid points is defined by calculating the length of each boundary segments and dividing it by the new number of intervals.

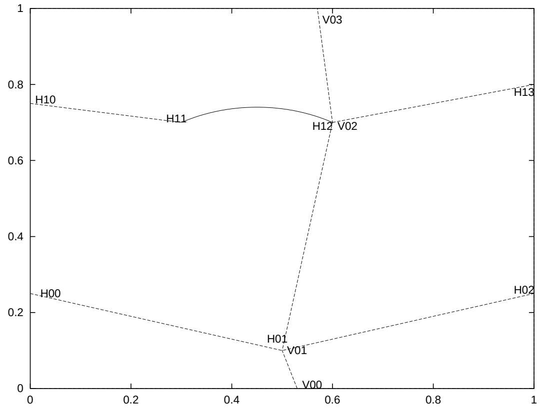

Figure 2.1-a: The goal is to align a grid with the two horizontal and one vertical SIACs in this square domain.

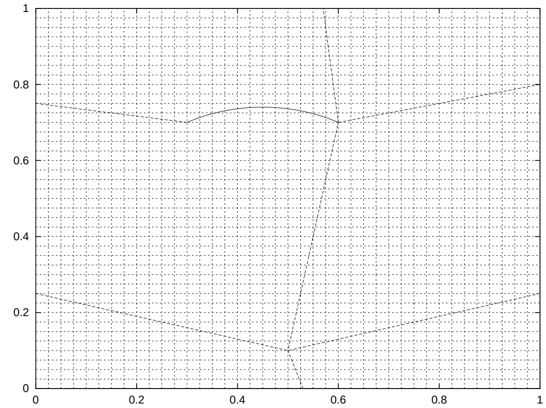

Figure 2.1-b: The initial reference grid for the domain in Fig. 2.1-a is a uniform 41×41 point grid.

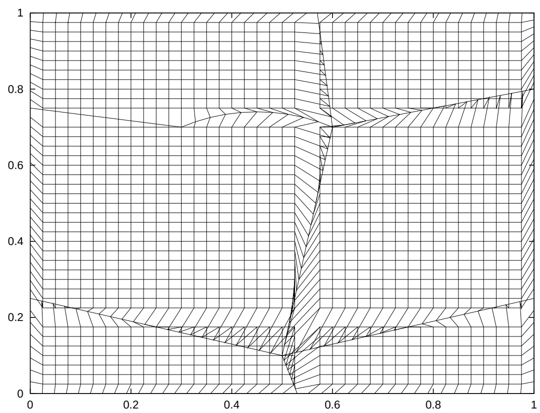

Figure 2.1-c: Moving the reference grid points closest to the SIACs to lie on the SIAC generates a possibly overlapping grid.

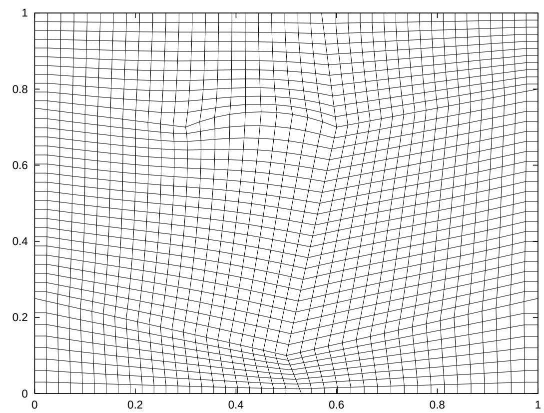

Figure 2.1-d: After smoothing the location of the grid points and redistributing the points along the SIACs and the domain boundary, the aligned grid accurately tracks the IACs.

2.1 Grid Smoothing Algorithm

Redefining the grid points on the SIACs and domain boundaries can result in severe distortions or tangling of the internal grid points (e.g., see Fig. 2.1-c). we regularize the positions of the grid points not on the SIACs or a domain boundary. That is, the mesh points on all SIAC and domain boundaries are frozen and the positions of the remaining points are smoothed by an iterative elliptic smoothing grid generation method. In the examples presented here, we use Gauss-Seidel iterations to solve the Thompson, Thames and Mastin (TTM) smoothing equations [14]. The smoothing regularizes the distribution of the grid points, and eliminates the overlapping (see Fig.2.1-d).

Once a smooth grid has been generated by freezing some of the grid points, the points on the domain boundaries are freed up and the smoothing grid generator is called again. We use the inverse interpolation algorithm to set the relative spacing of the points on the boundary to be the same as the relative spacing of the grid points along the first internal line of grid points. We found this two step process to be more robust than allowing the IAC points to move on the domain boundary in the initial smoothing step. This is especially true when the grid must be aligned with multiple IACs. For most problems, the boundary points need to be smoothed only a few times.

3 Grid Alignment for Quadrilateral IACs

We first consider the case when the internal discontinuities can be delineated by the boundary of a collection of *quadrilateral IACs* (QIACs). In our implementation, the grid is aligned sequentially with the perimeters of the QIACs and we assume that the alignment with one quadrilateral does not affect the alignment with the others. Therefore, although the QIACs can be adjacent to each other or nested within one another, they should not overlap.

To align a grid with internal quadrilaterals it is possible transform the problem into alignment with SIACs by extending the edges of the quadrilaterals as horizontal and vertical line segments to the domain boundaries. This approach succeeds for one or two quadrilaterals, but will usually produce a grid that is not as well structured as the one produced with an algorithm designed specifically for quadrilaterals. This is especially true when there are large numbers of quadrilaterals or the boundaries of the quadrilaterals and the relations between them, such as several quadrilaterals nested within each other, are extremely complicated. For these situations, it is extremely difficult to automate a procedure that will embed the QIACs into a SIAC problem.

3.1 Single QIAC

Consider the initial reference grid and the embedded quadrilateral shown in Fig. 3.2-a. We will use this example to illustrate the basic single QIAC algorithm:

1. Identify the closest reference grid point for each vertex and assign logical (index) coordinates for the vertices and the center of QIAC:

   - Identify the grid point closest to each vertex of the quadrilateral and assign its logical coordinates to the vertex. In Fig. 3.2-a if the grid lines are numbered from 0 to 10, then the grid index closest to the vertex V_0 at $(0.2, 0.6)$ is $(2, 6)$, V_1 is at $(5, 2)$, V_2 is at $(8, 4)$, and V_3 is at $(5, 8)$. Here we have ordered the four vertices in counterclockwise in the logical (index) coordinate system.
   - To determine a starting vertex of QIAC for the algorithm in a consistent manner, we use an algorithm that identifies it as the *lower left* corner of the logical reference

grid. The horizontal logical coordinate of this vertex must be smaller than the next vertex, and average of the absolute values of the logical slopes of the candidate starting edge and the corresponding opposite side of the QIAC must be less than that of the other two edges of the QIAC. For this example, only V_0 and V_1 satisfy the first condition. For the sum of absolute value of the logical slopes between $V_0 - V_1$ and $V_2 - V_3$, is $((6-2)/(5-2)+(8-4)/(8-5) = 8/3)$. The sum between $V_1 - V_2$ and $V_0 - V_3$ is $((4-2)/(8-5)+(8-6)/(5-2) = 4/3)$. Therefore, we define V_1 to be the lower left corner in logical coordinates.

- Define reference points in the logical coordinate system.
  - Define the logical center of the quadrilateral as the average (truncated to an integer) of the logical coordinates of the four vertices. For this example the logical center is at $V_c = (V_1 + V_2 + V_3 + V_0)/4 = (5,5)$.
  - Define average logical length in the x and y directions of the logical coordinate system as $l_x = (V_2 - V_1 + V_3 - V_0)|_x/2 + 1 = (8-5+5-2)/2+1 = 4$ and $l_y = (V_3 - V_2 + V_0 - V_1)|_y/2 + 1 = (6-2+8-4)/2+1 = 5$.
  - Define the logical coordinates for the four vertices based on the logical coordinates of the center and average length in each direction. The logical coordinates for the lower left corner V_1 is $V_1 = (V_c - (l_x, l_y)/2) = (3,3)$. The logical coordinates for other three vertices are V_2=(7,3), V_3=$(7,7)$, V_0=$(3,7)$.

2. Split the domain into nine parts by connecting the four vertices with the domain boundaries by moving the closest reference point to each vertex as shown in Fig. 3.2-b. For this example, the closest reference point of each vertex and vertex itself are at the same location. For general grid, this may not be true and some parts may degenerate into a single grid line if one side of the QIAC is on the domain boundary.

3. Using inverse interpolation, compute the grid locations for the line segments on the quadrilateral, along the lines connecting these segments to the domain boundaries and on the domain boundaries as shown in Fig. 3.2-c.

   - First define the grid points on the lines connecting the vertices with the domain boundaries. Note that to preserve the order of the grid points on a boundary, the locations of the boundary points and the points nearest to the boundaries are not changed by the inverse interpolation.
   - Using the inverse interpolation, define the grid positions on the domain boundary.
   - Define the grid points between the vertices on the QIAC.

4. If the domain boundary has a kink point, move the reference grid point closest to the kink point to the kink point, as we did for aligning the grid with an SIAC. (Note, that there are no kink points in the example shown in Fig. 3.2-a). Because we modify the grid point locations only on the four lines, where the edges of quadrilateral lie, and the domain boundaries, the grid we generate after this step is very irregular (see Fig. 3.2-d).

5. Using the same smoothing operator as for the SIAC, generate the aligned grid while freezing the grid points on the QIAC and domain boundary. Then the points on the domain boundaries are freed up and the smoothing grid generator is called again (see Fig. 3.2-e).

To insure that initially the different vertices have different logical coordinates, the reference grid should be sufficiently fine so the local minimum spacing for the reference grid should be larger than the distance between any two vertices of the QIACs. Otherwise, two vertices may have the same nearest reference point, which could lead to the singularity of the quadrilateral. When this happens for a thin layer where two vertices are close, it can be avoided by using a finer reference grid near the thin layer. To prevent multiple QIAC vertices being assigned to the same reference grid point, we define the reference vertices sequentially. That is, we bind the vertex that is closest to a reference point to that point and eliminating it for consideration as a reference point for any other vertices. This approach is robust, and can create a locally fine grid near the quadrilateral.

3.2 Multiple QIACs

When there are several quadrilaterals inside the domain, we recursively align the grid with the quadrilaterals one at a time using the single QIAC algorithm. Before beginning the alignment process, we analyze the QIACs to identify potential problems, such as overlapping QIACs or QIACs with shared boundaries. We minimized these problems by defining new QIACs to be used during the alignment algorithm.

When a QIAC shares a boundary with its parent QIAC, we say the child shared boundary is contained in the boundary of the parent QIAC. When two QIACs are not nested but share only part of a boundary, it is convenient to generate a new QIAC and to define three QIACs where the shared boundary is now contained in the boundary of another quadrilateral. An example is shown in Fig. 3.3-a. If two QIACs intersect at a vertex, then we add another quadrilateral which shares a boundary with both quadrilaterals (see Fig. 3.3-b). At this time, our algorithm does not support more than three quadrilaterals intersecting at a single point.

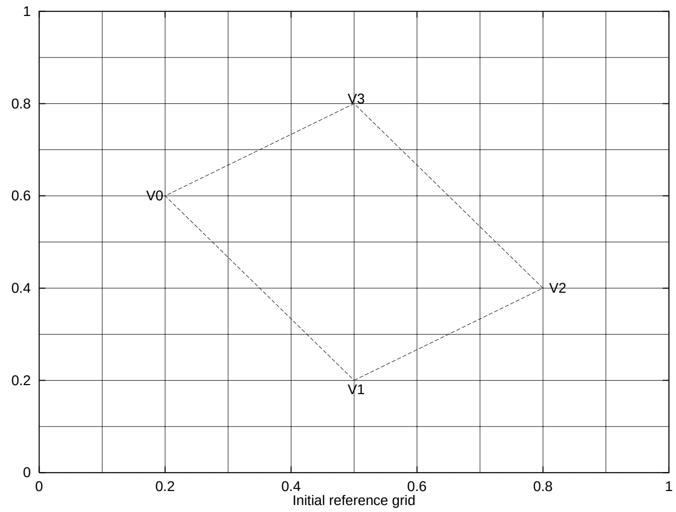

Figure 3.2-a: The initial QIAC is embedded in a uniform 11×11 initial grid.

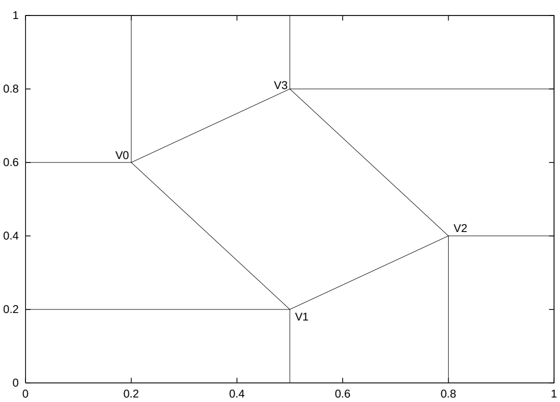

Figure 3.2-b: The domain is split into nine subdomains by extending the vertices to the domain boundaries.

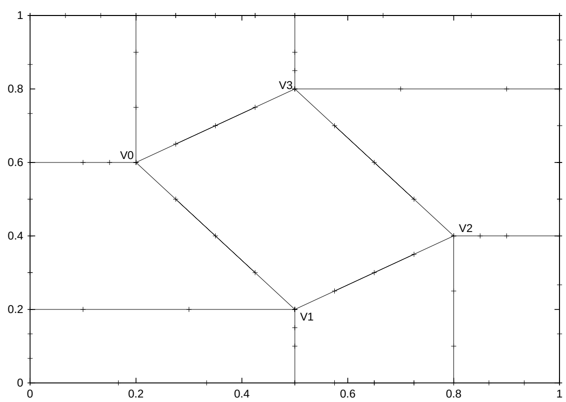

Figure 3.2-c: The reference grid points are defined for the line segments on the QIAC, the lines connecting the vertices to the domain boundaries and along the domain boundaries.

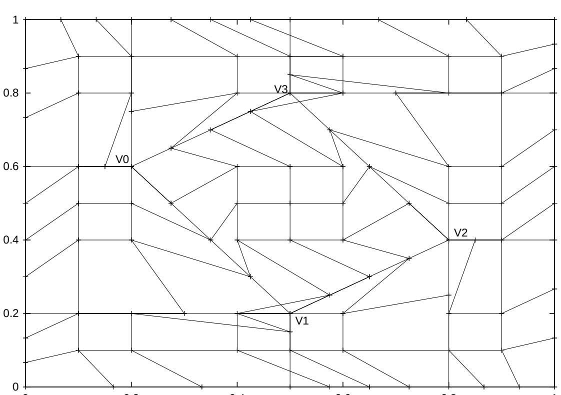

Figure 3.2-d: The initial grid points nearest the IACs are moved to the newly defined reference grid points.

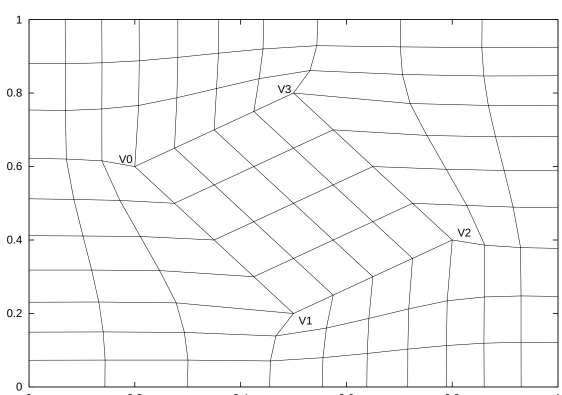

Figure 3.2-e: The grid is regularized by the two-step smoothing algorithm described in Section 2.1.

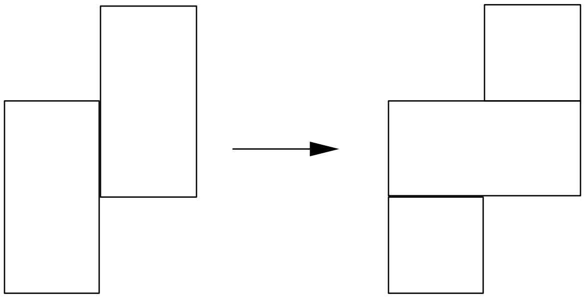

Figure 3.3-a: Two QIACs that initially share part of the boundary are transformed into three QIACs where the shared boundary is now contained in the boundary of another quadrilateral

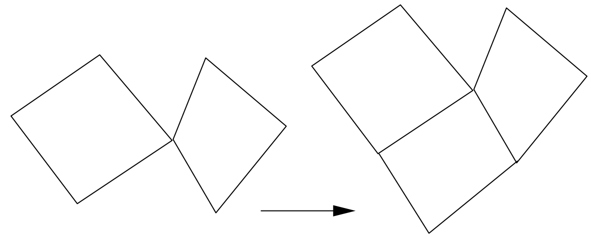

Figure 3.3-b: Two QIACs that share a vertex are transformed into three QIACs where the shared boundary is now contained in the boundary of another quadrilateral

We order the alignment process by defining a tree structure (as shown in Fig. 3.3-c) to catalog then nesting of the QIACs with respect to each other. A QIAC with another QIACs inside is defined as the parent and the included QIACs are its children and are linked below the parent in the alignment tree. Two QIACs are siblings if they have the same immediate parent but do not contain each other. We define the whole domain as the first parent at the top of the alignment tree. The recursive alignment algorithm begins at the top of the tree and works it way down. After the grid has been aligned with a QIAC, then the alignment algorithm aligns the grid for its children. After the grid has been aligned with all the children of a QIAC, then is aligned with the siblings of the QIAC.

The order in which the grid is aligned with the QIACs on the same level does affect the final aligned grid. In practice the resulting grid is insensitive to the order that the grid is aligned with the siblings on the same level. We do find that the algorithm is usually more effective and requires fewer smoothing iterations if the grid is aligned with the sibling that has the most share parts with other QIACs first.

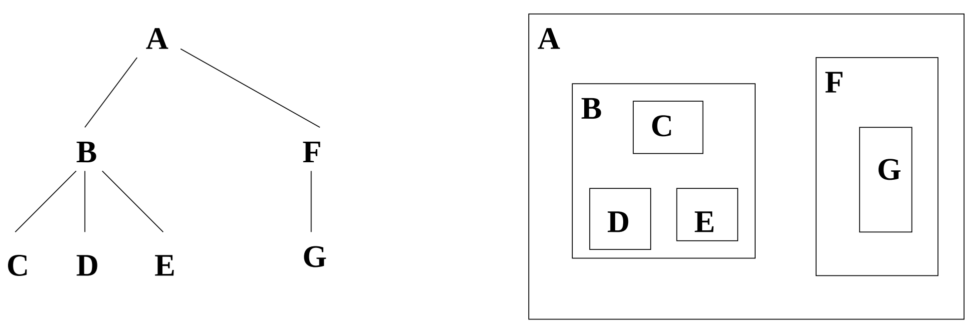

Figure 3.3-c: The multiple QIACs in the figure on the right are analyzed by the tree diagram on the left and the grid the QIACs are considered in the order A-B-C-D-E-F-G.

Once the alignment tree has been defined and the next QIAC to be aligned has been selected, the first step is to find the largest isolated *subdomain* of a previously aligned region that contains the *selected* QIAC (see Fig. 3.3-d). The boundary of the subdomain could be the boundary of the whole domain, the boundary of the previous aligned quadrilaterals, or previously aligned horizontal or vertical SIACs.

To minimize the effect of defining the grid in the subdomain, the subdomain is defined to be as large as possible, but not to contain any previously aligned QIACs. To identify the largest isolated subdomain, we start with the whole domain, and shrink it step by step.

1. First we check if the quadrilateral is inside another previously aligned quadrilateral. If so, then because we are refining from top to bottom of the parent-child alignment tree, if a quadrilateral is nested inside another one, the larger one will have already been aligned. If this is the case then we shrink the subdomain to be the boundaries of the parent aligned quadrilateral.

2. Next if the current quadrilateral is not the child of a previously aligned QIAC, then we check if it is between any SIACs. If so, then shrink the subdomain by using these IACs to define the new boundaries of the subdomain.

3. Finally, we check if there are previously aligned QIACs inside the current subdomain. If so, then the subdomain is shrunk again until there is no other aligned QIACs in the subdomain. The rectangular subdomain is shrunk until it is the largest isolated rectangular region containing the selected QIAC that does not intersect or contain a previously aligned QIAC. This can be done by sweeping each boundary of the subdomain to exclude the previously aligned QIACs (see Fig. 3.3-d). Note that the boundary of the largest isolated subdomain usually overlaps with the boundaries of previously aligned QIACs.

It may seem that the ideal subdomain for the selected QIAC would be one whose boundaries contain the boundaries of the selected QIAC. However because the subdomain boundary does not change during our alignment and smoothing, unless the selected QIAC is on the external boundary of the domain or on a previously aligned IAC, this is usually a poor choice for the alignment.

The grid is aligned for the selected QIAC by applying the single QIAC alignment algorithm within the selected subdomain. Although for the single QIAC alignment algorithm, the distribution of points on boundary of the whole domain is unrestricted, the boundary of the selected subdomain may contain other previously aligned QIACs. If so, the logical indices for the vertices of previously aligned QIACs should not be changed, although the points can move along the subdomain boundary during the subdomain alignment. We allow this movement by identifying the vertices of the previous aligned IACs on the subdomain boundary and redistribute the points based on this information of these IACs and the selected QIAC.

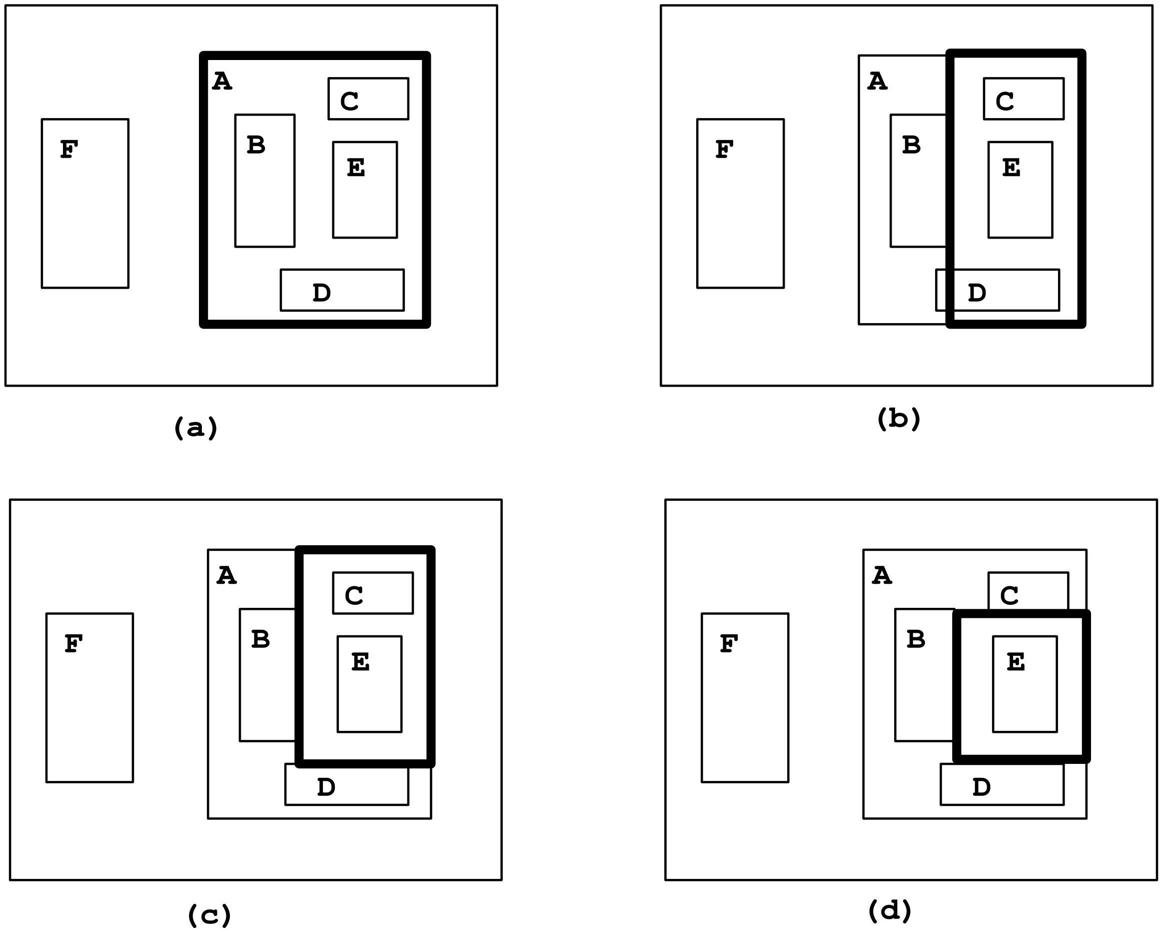

Figure 3.3-d: If the grid has already been aligned with the QIACs **A, B, C** and **D**, then to identify the largest isolated subdomain for the QIAC **E** that is as large as possible but does not contain other previously aligned QIACs. First since QIAC **A** is a parent of **D**, we take it as the initial subdomain for **E** in Fig. (a). Because this still contains the previously aligned QIAC **B**, this subdomain is shrunk to the one shown in Fig. (b). The subdomain is further reduced in Figs. (c) and (d) to exclude the previously aligned QIACs **C** and **D**.

When the boundary of the selected subdomain is a SIAC, then we have more freedom in defining the number of grid points along the IAC. If the boundary of the subdomain contains vertices of an SIAC, these points are treated as kink points.

Finally, when the vertex for the selected QIAC is also a vertex of a previously aligned QIAC, then the logical indices for this vertex have been assigned and cannot be changed.

The algorithmic flow of the alignment algorithm for multiple QIACs:

1. Preprocess the QIACs to eliminate overlapping QIACs and shared boundaries.

2. Define the parent-child alignment tree to catalog how the QIACs are nested within each other

3. Define an order for the alignment process by sweeping the alignment tree from top to bottom and ordering the 'most difficult' siblings first.

4. Recursively align the grids with the QIACs using the single QIAC algorithm applied to the QIAC subdomain.

   a. Define the selected QIAC as the next one in the order queue.

   b. Define the subdomain that contains only the current QIAC.

   c. Align the grid with the selected QIAC within the subdomain.

   d. Smooth the grid for the selected QIAC within the subdomain.

   e. If there are other QIACs to align, go to step a.

5. Smooth the entire grid again.

When the boundary of a QIAC is not straight lines, the position of the grid points on that boundary should be computed according to the shape of the edge.

When the grid is to be aligned with both SIACs and QIACs, we could first transform the line segments into the boundaries of QIACs and then use the multiple QIAC algorithm to align the grid. However, because the SIAC alignment algorithm is usually simpler, more efficient and can produce a better structured grid than the QIAC algorithm, it is advantageous to keep the SIACs and directly combine the two algorithms. We do this by first aligning the grid with the SIACs. Next, if there are QIACs that cross SIACs these quadrilaterals are split so they do not cross an SIAC. We then apply the QIAC algorithm while freezing the mesh points on the SIACs.

If the IACs are not quadrilaterals, they can usually be subdivided into quadrilaterals. We represent triangles as quadrilaterals by defining the center of the longest edge of the triangle to be a degenerate (180^o) vertex of a quadrilateral with the same perimeter. The user can represent the IAC regions with more than four edges by dividing the region into quadrilaterals and triangles. Because those splitting introduce additional internal boundaries which are not IACs, the mesh points are not required to stay on these artificially created internal boundaries during the smoothing iterations. This is accomplished by setting a flag for each vertex on the internal boundaries to indicates if the point is on an IAC or not.

4 IAC Grid Alignment Examples

To demonstrate the strengths and weaknesses of the IAC grid alignment algorithm, we consider examples demonstrating the effectiveness of the algorithm for a problem with multiply imbedded regions, including circles and nested quadrilaterals. Next we illustrate how by ordering and subdividing of QIACs can result in very different aligned grids. The examples all are initialized with a 41×41 uniform grid, unless it is explicitly stated otherwise.

The computer software for these examples was developed by Shengtai Li and Patrick Knupp and is available through the web site "http://engineering.ucsb.edu/˜shengtai". There is also a graphics interface to display the grid during the refinement process and allow for interactive steering of the alignment process. The original IAC alignment code was written in Fortran, but lacked the flexibility to accommodate multiple shaped internal boundaries, without extensive changes for each special case. The current C++ software is fare more flexible than the Fortran software in accommodating different shaped boundaries. The virtual function in the C++ allows different shape of the boundaries to be computed with the same routine as long as the boundaries are access through pointer or reference.

4.1 Multiple imbedded SIAC and QIAC Examples

In the first example we illustrate the flexibility of the IAC algorithm by including imbedded QIACs, QIACs with a shared edge and an ellipse section. Note that the large QIAC-I shown in Fig. 4.5-a contains two smaller ones (II and III) and that QIAC IV shares an edge with the large QIAC. Also, QIAC V is a circle that has been delineated by four circular arcs connecting the vertices. The numbering of the QIACs is also the order in which the IAC algorithm aligned the grid.

In all of the examples, the internal quadrilateral are defined by four vertices ordered in an anti-clockwise. This same order is used to flag if the mesh points on the internal boundaries are IACs or were artificially created by the algorithm when dividing the region into SIACs and QIACs.

The aligned grid in Fig. 4.5-b illustrates the effectiveness of the algorithm on this complex example. Note that the initial uniform reference grid points along the boundary are now nonuniform to better accommodate the IACs. These mesh points were allowed to slide along the boundaries when the grid was aligned with the internal QIAC. That is, the boundaries of outside QIAC define a small domain for the inside QIAC. This holds also when we align the grid with object IV. The inverse interpolation preserves the shape of the previous QIACs, although the points on the QIAC boundary are redistributed. We also observed that during the alignment algorithm, when the grid was aligned with QIAC V and VI, the number of points on the boundary of QIAC I did not change. Although the default line property is a straight line, the boundaries of the objects and line segments can be defined by the user to be any shape. This is done to define the ellipse as QIAC V.

In geophysical applications, it is common to have very thin layers that must be aligned with the grid. Among the mixed SIACs and QIACs in Fig. 4.5-c is a very thin QIAC layer. This layer is below the thickness of our original reference grid, but the algorithm resolves the potential conflict in assigning a single reference grid point to multiple QIAC vertex points. The IAC grid shown in Fig. 4.5-d. also illustrates how the circular arc in the SIAC results in a kink point being created where the straight line segments and circular arc meet. The kink point can be used by the user to indicate special points where a mesh point must be placed. A list of the kink points defining their position must be supplied by the user before the initial grid generation. This was done to define the kink in the upper horizontal SIAC.

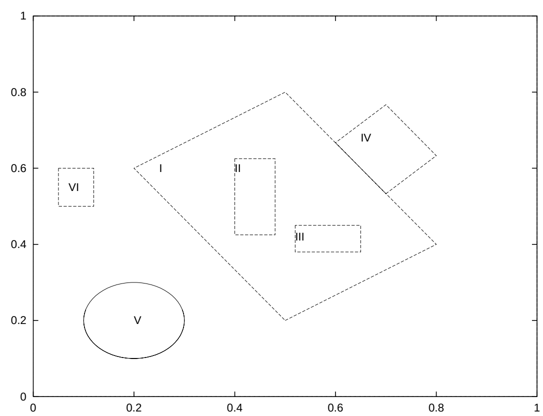

Figure 4.5-a: Six QIACs, including a circle, two QIACs with a shared boundary and two nested QIACS, are defined in a square domain.

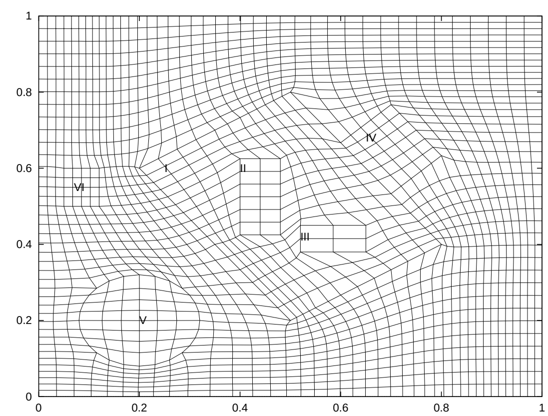

Figure 4.5-b: The IAC alliged grid for the QIACs shown in Fig. 4.5-a.

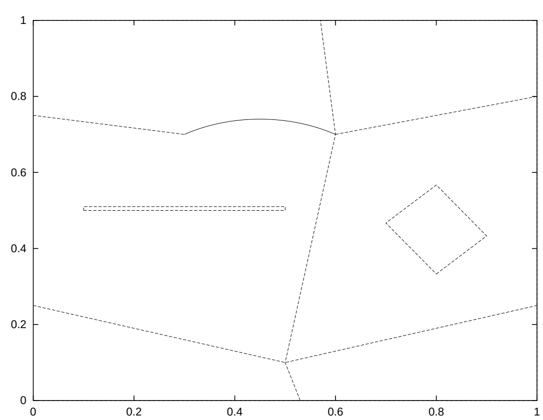

Figure 4.5-c: Three SIACs and two QIACs identify the interfaces to align the grid. The thin QIAC is approximately half the original grid spacing.

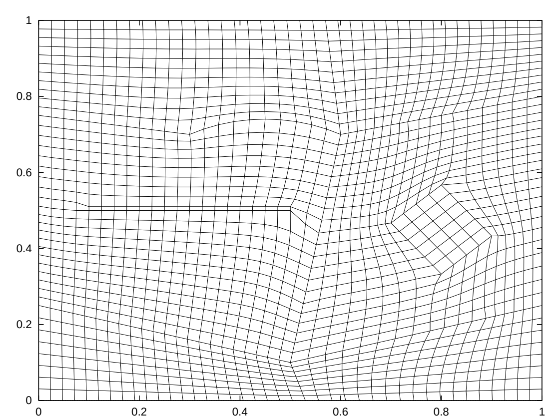

Figure 4.5-d: The IAC aligned grid for Fig. 4.5-a accurately tracks the interfaces and resolves the very thin QIAC.

4.2 Sensitivity of the grid to the IAC procedure

To investigate the sensitivity of the final aligned grid to the way in which the region is subdivided, we consider the QIAC shown in Fig. 4.6-a. If we follow the procedures described in Sec. 3.1, the resulting grid is almost a square grid imbedded in the slanted rectangle shown in Fig. 4.6-b. Note that the density of grid points inside the QIAC is less than elsewhere in the domain.

We next divided the QIAC into two triangles and one slanted rectangle, as shown in Fig. 4.6-c. We described the triangles as degenerate QIACs with a vertex in the middle of the long edge. We set the flag not to freeze the grid points along these long edges in the final grid smoothing step. That is, the grid points will stay aligned with the original QIAC, but not the internal ones. However, because the intermediate steps require more grid points to align the grid with the triangle, more grid points are 'trapped' inside the original QIAC. This results in far more internal grid points in the final aligned grid, Fig. 4.6-d.

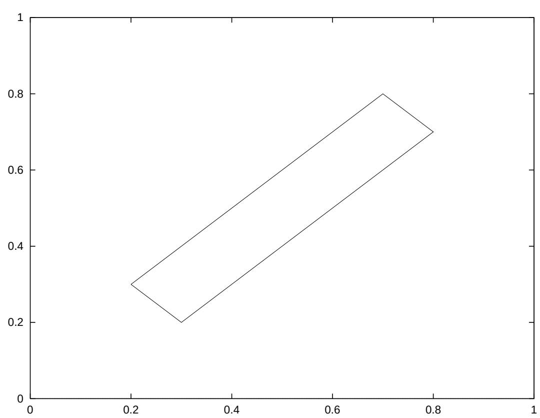

Figure 4.6-a: A diagonal QIAC is imbedded in a square domain.

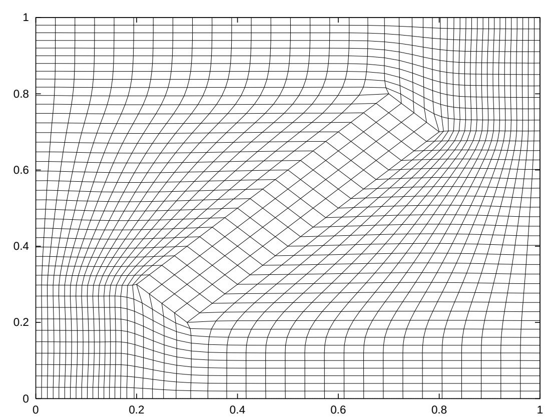

Figure 4.6-b: The IAC aligned grid inside the QIAC for Fig. 4.6-a is coarser than elsewhere in the grid.

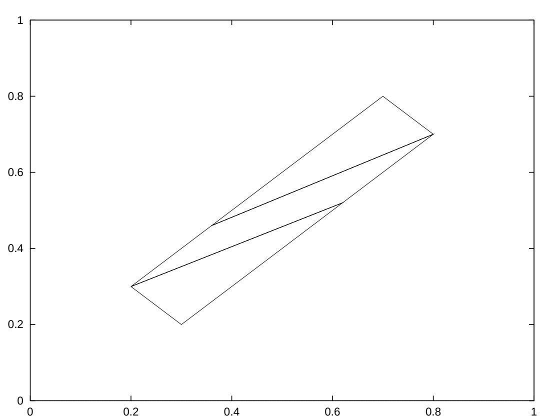

Figure 4.6-c: The original QIAC in Fig. 4.6-a is split into two triangular QIACs and on diagonal QIAC in an attempt to confuse the IAC algorithm.

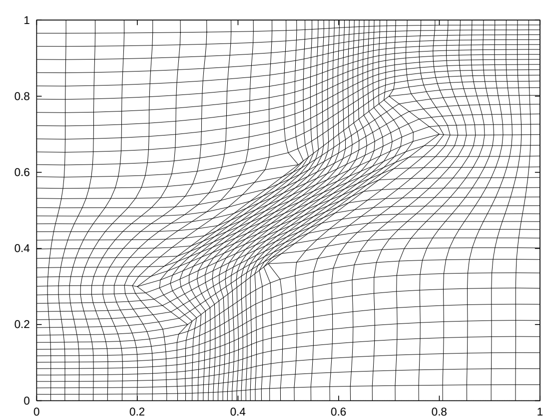

Figure 4.6-d: The internal boundaries for the new QIACs in Fig. 4.6-c are not treated as IACs in the smoothing iteration. Note that the resulting mesh is much finer in the QIAC than in Fig. 4.6-b.

4.3 IACs in nonrectangular domains

Consider the three rectangular QIACs inside a shell domain shown in Fig. 4.7-a. As described in Sec. 3, the current implementation of our algorithm requires that the left or the right QIAC must be aligned first. Note that the sequential nature of the algorithm broke the symmetry of the grid shown in Fig. 4.7-b. Because we align the grid with the left QIAC before the right one, the grid is more regular on the left side of the domain. Other ordering of the QIACs will alter the symmetry. If a symmetric grid is desired for a symmetric problem, then the symmetry should be removed from the IAC process by solving, in this case, for half the domain and then reflecting the grid about the symmetry line to generate the full symmetric aligned grid.

Figures 4.7-c and 4.7-d illustrate the IAC grid for two line segments (a SIAC) and a square QIAC inside a quarter disk domain. As in the first example, the kink point in the SIAC must be explicitly accounted for when aligning the QIAC.

The "C"grid in Figures 4.7-e and 4.7-f is defined to have a kink point at the tip of the triangle on the domain boundary. This example also provided a good test for the effectiveness of the inverse interpolation routine to freeze the fixed point at the kink point in the middle of the boundary.

5 Discussion

We have described a new numerical algorithm that aligns an initial noaligned quadrilateral grid with internal curves delineating internal interfaces in a computational domain. We demonstrated the robustness and versatility of the IAC grid alignment for internal boundaries in a series of numerical examples. The IAC algorithm can be generalized to three dimensions with moderate effort. Just as the problem of generating a two-dimension grid was reduced to that of generating a single grid curve in the plane, so the problem of generating a three-dimensional grid reduced to the problem of generating a surface in space.

Acknowledgement The work was performed under the auspices of the U. S. Department of Energy (DOE) contract W-7405-ENG-36 and the DOE/Bureau of Energy Sciences Program in Applied Mathematical Sciences.

References

[1] J.U. Brackbill, An adaptive-grid with directional control, *J. of Comp. Physics*, **108**, no. 1, pp. 38-50, 1993.

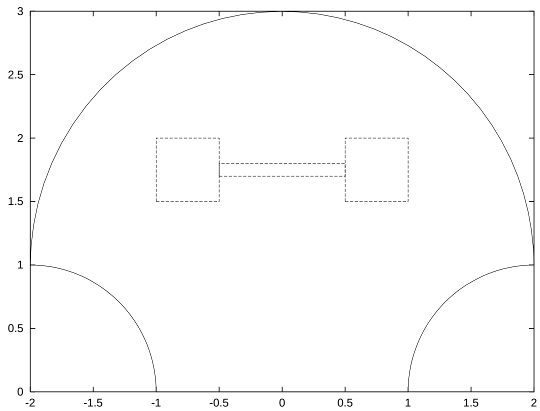

Figure 4.7-a: Two diamond oriented QIACs are input symmetrically in a shell shaped domain.

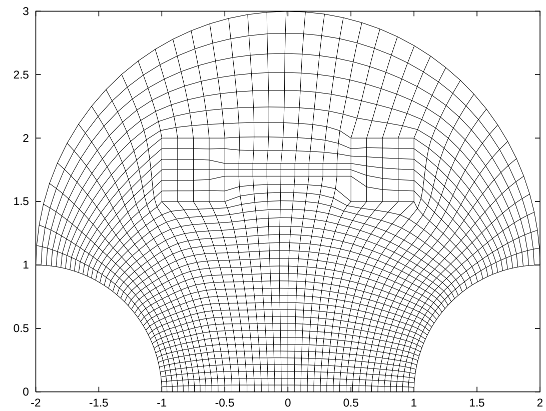

Figure 4.7-b: The sequential nature of the IAC grid algorithm does not preserve the symmetry of the QIACs in Fig. 4.7-a. To preserve the symmetry, the grid should be generated for half the domain and then reflected about the line of symmetry.

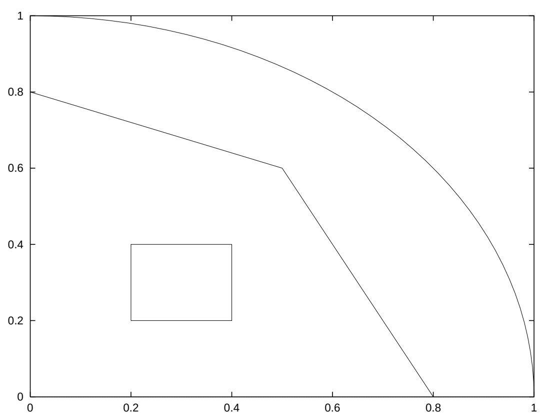

Figure 4.7-c: A QIAC is imbedded in a disk shaped domain.

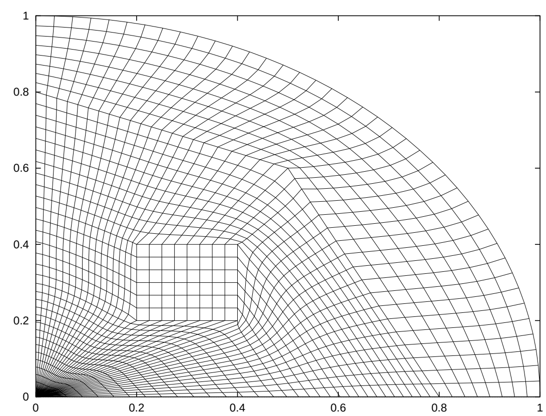

Figure 4.7-d: The IAC grid for the QIACs in Fig. 4.7-c creates an almost perfect square grid for the imbedded rectangle.

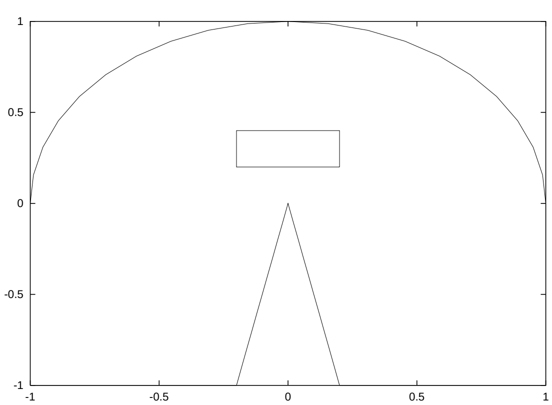

Figure 4.7-e: The initial domain wraps around a triangular point and has an imbedded QIAC.

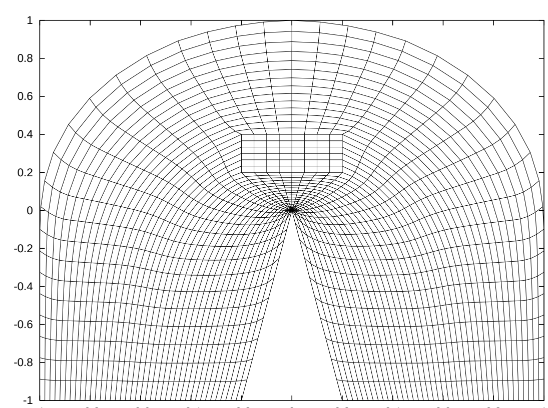

Figure 4.7-f: The IAC aligned "C" for Fig. 4.7-e preserves the symmetry of the domain since there is only a single QIAC.

[2] R.L. Dougherty and J.M. Hyman, A divide-and-conquer algorithm for grid generation, *Applied numerical mathematics*, **14** (1994).

[3] M.H. Garcia, A.G. Journel and K. Aziz, 1990, An automatic grid generation and adjustment method for modeling reservoir heterogeneities, Stanford center or reservoir forecasting Report 3.

[4] A.E. Giannakopoulos, and A.J. Engel, Directional control in grid generation, *J. of Comp. Physics*, **74**, no. 2, pp. 422-439, 1988.

[5] J.M. Hyman , Numerical Methods for Tracking Interfaces,, *Physica D*, **12**, pp. 396–407, 1984.

[6] J.M. Hyman and M.J. Naughton, Static rezone methods for tensor-product grids, in Proceedings of SIAM-AMS Conf. on Large Scale Computations in Fluid Mechanics, SIAM, Philadelphia, 1984.

[7] J.M. Hyman and M. Shashkov, Mimetic discretizations for Maxwell's equations, to appear *J. of Comp. Physics* **151**, no. 2, pp. 881-909 , 1999.

[8] C. Jastram and E. Tessmer, Elastic modeling on a grid with vertically varying spacing, *Geophysical Prospecting*, **42**, no. 4, pp. 357-370, 1994.

[9] P. K. Knupp, Mesh generation using vector Fields, *J. Comp. Phys.*, **119**, pp. 142-148, 1995.

[10] P. K. Knupp, Jacobian-weighted elliptic grid generation, *Siam J. Sci. Comp.*, **17**, no. 6, pp. 1475-1490, 1996.

[11] P.M. Knupp and S. Steinberg, Fundamentals of Grid Generation,1993, CRC press Florida.

[12] V.D. Liseikin, The construction of structured adaptive grids - A review, *Comp. Methods in Math. Phys.*, **36**, no. 1, pp. 1-32, 1996.

[13] T.H. Robey, An adaptive grid technique for minimizing heterogeneity of cells or elements, 1995, *Math. Geology*, **27**, no. 6, pp. 709-729, 1995.

[14] J.F. Thompson, F.C. Thames and C.W. Mastin, Automatic numerical generation of body-fitted curvilinear coordinate system for field containing any number of arbitrary two-dimensional bodies. *J. Comp. Physics* **15**, pp. 299-319, 1974.

[15] J.F. Thompson, Z.U.A. Warsi and C.W. Mastin, Numerical grid generation: Foundations and applications, Elsevier North-Holland, New York, 1985.

[16] J.Y. Trepanier, M. Paraschivoiu, M. Reggio, and R. Camarero A conservative shock fitting method on unstructured grids, *J. of Comp. Physics*, **126**, pp. 421-433, 1996.

[17] C. Zhang and R.J. LeVeque, The immersed interface method for acoustic wave equations with discontinuous coefficients, *Wave Motion*, **25**, pp. 237-263, 1997.

[18] H. Zhang and M.K. Moallemi, MAGG: A multizone adaptive grid-generation thechnique for simulation of moving and free-boundary problems, *Numerical Heat Transfer Part B-Fundamentals*, **27**, no. 3, pp. 255-276, 1995.