INTELLIGENT AUTOMATED SURFACE GRID GENERATION

Ke-Thia Yao and Andrew Gelsey
Computer Science Department
Rutgers University
New Brunswick, New Jersey

Abstract

The goal of our research is to produce a flexible, general grid generator for automated use by other programs, such as numerical optimizers. The current trend in the gridding field is toward interactive gridding. Interactive gridding more readily taps into the spatial reasoning abilities of the human user through the use of a graphical interface with a mouse. However, a sometimes fruitful approach to generating new designs is to apply an optimizer with shape modification operators to improve an initial design. In order for this approach to be useful, the optimizer must be able to *automatically* grid and evaluate the candidate designs. This paper describes an intelligent gridder that is capable of analyzing the topology of the spatial domain and predicting approximate physical behaviors based on the geometry of the spatial domain to automatically generate grids for computational fluid dynamics simulators. Typically gridding programs are given a *partitioning* of the spatial domain to assist the gridder. Our gridder is capable of performing this partitioning. This enables the gridder to automatically grid spatial domains of wide range of configurations.

1 Introduction

Partial differential equation solvers require a grid, a discretization of the spatial regions of interest. In computational fluid dynamics, one type of the spatial regions of interest is the surface area the fluid contacts. The quality of the grid strongly affects the accuracy and the convergence properties of the resulting simulation. Generating a proper grid involves reasoning about the geometry of the regions of interest, the physics of the situation and the peculiarities of the numerical analysis code. To deal with the complexities of gridding, the current trend in the gridding field is toward interactive gridding [Remotique, Hart, & Stokes 1992, Kao & Su 1992]. Interactive gridding more readily taps into the
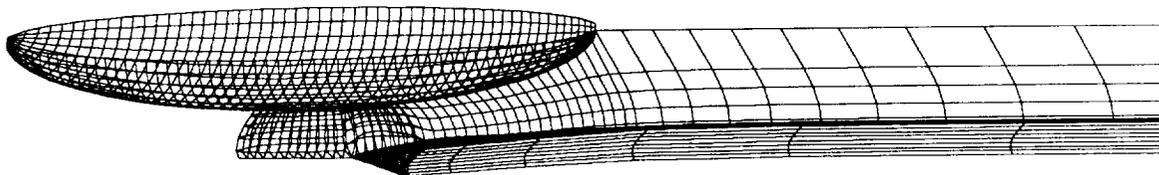
581

Figure 1: Yacht (consisting of three components hull, keel, and winglet) with wake sheets.

spatial reasoning abilities of the human user through the use of a graphical interface with a mouse. However, this approach is not acceptable for automated design systems, such as the Design Associate (DA) [Ellman, Keane, & Schwabacher 1992] for racing yachts. In the process of designing a yacht, the DA must repeatedly evaluate candidate yacht designs. A large number of these evaluations are required, so the capability to automatically evaluate the performance of a candidate yacht design without human intervention is crucial for the success of the DA.

We are working in the physical domain of fluid dynamics, in particular potential flows modeled by Laplace's partial differential equation. The potential flow solver we use is PMARC, a product of NASA Ames Research Center. The input PMARC requires is a panelization — a discretization of an object's wetted surface as a grid of *surface patches*, where each surface patch is an array of approximately planar quadrilateral panels. This array of panels is represented in PMARC as a matrix of corner points. See Figure 1 for a grid of a yacht automatically generated for PMARC by our gridding program.

The yacht in Figure 1 consists of three input components: an ellipsoid hull, the *Star & Stripes* keel, and the *Star & Stripes* winglet.[1] The wake sheets attached to the rear of the yacht are the vortices shed by the yacht. Discussions on how to attach wakes and how to determine the shape of the wakes are beyond the scope of this paper. The *Star & Stripes* winglet attached to the bottom of keel is considered a major innovation in the field of racing yachts, and the success of the *Star & Stripes* was in part due to its winglet. Current automated gridding programs should be but are not able to handle this kind of innovative *topological* change in design without human assistance.

The input to the gridder is expressed in a language we have developed called Boundary Surface Representation (BSR). Figure 2 graphically depicts the BSR input for this yacht example. We shall use this yacht example throughout this paper. Both BSR and the input will be discussed in much more detail later. For now we'll point out that BSR input consists of two major parts: *geometrical* and *topological*. The geometrical part represents the detailed features of the yacht, which are the three input surface mappings (*shape*) in the figure. The *topological* part contains information on the adjacency of the input surfaces.

---

[1]The *Star & Stripes* is the yacht that won the 1987 America's Cup Competition.

The adjacency information is represented by dotted lines in the figure.

2 Why is automated gridding hard?

2.1 Steps to gridding

We divide gridding into three steps. The first step is to *partition* the input surface into griddable *surface patches*. That is, this step finds the appropriate boundary lines (or *partitioning lines*) for the surface patches. As we'll see this step is often the most difficult, because it involves significant physical and geometrical reasoning.

Step two, for each surface patch, *reparametrize* it by defining two families of approximately orthogonal grid lines. A formal definition will be given later when BSR is defined. But, intuitively suppose a surface patch is laying on the xy-plane, then $\{x = constant, y = constant\}$ is one possible parametrization, and $\{x + y = constant, x - y = constant\}$ is another.

The last step is to determine how many grid lines to lay down on each of the surface patches, and in particular where to lay them down. This step corresponds to picking the constants to instantiate the equations in step two. The intersections of these grid lines form corner points of the array of panels, which is the input to PMARC. This step we shall call the grid line *distribution* step. The distribution of grid lines can make grids with the same reparametrization look different and may make the numerical simulator behave differently. For example, using the *equal-distance* distribution scheme, $x = i/10$, where $i = 0, \ldots, 10$, may make the numerical simulator converge slower than a *cosine* distribution scheme, $x = (1 - \cos \pi i/10)/2$, where $i = 0, \ldots, 10$.

2.2 Evaluation criteria

Gridding as defined by the three steps above is unconstrained. The ultimate test for a grid is to check how sound the resulting simulation is, and how well it resolves the physical features of the domain. Short of feeding the grid to a simulator, there are ways of checking the goodness of a grid.

Through our discussion with hydrodynamicists we have formulated a list of grid evaluation criteria and constraints. On the basis of the geometric properties of the grid, these evaluation criteria attempt to predict the soundness of PMARC's output. We divide this list into four levels, ranging from constraints that absolutely must be satisfied to heuristic advice based on experiences of our experts.

1. Simple connectedness constraint: surface patches must be simply connected, i.e., no holes.
2. Coverage constraint: patches must not overlap or leave gaps.
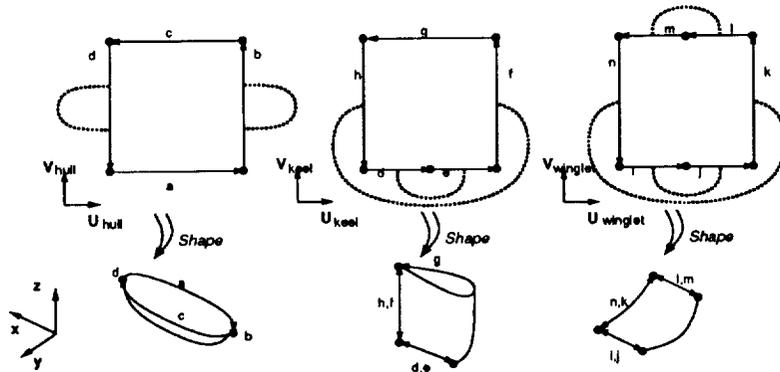3. Planarity criterion: panels must be approximately planar.

Figure 2: BSR input

4. Heuristic criteria:

- following streamlines: grid lines should follow the streamlines of the fluid flowing over the body.
- orthogonality: grid lines should intersect at right angles.
- expansion ratio: the area of the adjacent panels should not increase by more than a fixed ratio.

2.3 Difficulties of partitioning

Much work has been done on the problem of automated gridding [Thompson, Warsi, & Mastin 1985], and many gridding programs have been developed. However, most of these efforts concentrate on developing new methods of reparametrization and new distribution schemes. The choices of which reparametrization method and which distribution scheme to use are usually left to the human expert.

Most of the programs rely exclusively on the human expert to do the partitioning. He is expected to do the partitioning by either writing batch commands, or more recently by using an interactive graphical interface. In either case, the partitions created only apply to the one particular problem at hand. More recently, [Schuster 1992] has been trying to revive batch mode gridding by writing more general batch commands. However, his program is only able to grid a small, fixed set of airplane topologies.

One of the fundamental problems with the current gridding programs is that they do not make use of topology. All the topological information has been distilled away by either having the user provide the partitions or by fixing the possible topologies. The programs can only work on individual surface patches. Another problem is that programs have neither the knowledge of physics nor the knowledge of numerical analysis needed to generate grids that will lead to good simulations.

One manifestation of the lack of physical knowledge is as follows. A closer examination of the surface area near where the hull and keel meet reveals that the keel actually protrudes into the hull, and the hull has an extra surface area where the keel is. Surfaces given to the gridding program often contain *fictional surface areas*, areas that should not be gridded. Fictional surface areas are useful because they allow the hull and keel to be modified independently while still remaining in contact. However, an automated gridding program must be able to distinguish between the real and fictional areas in order to satisfy the *coverage* constraint.

Recall that PMARC represents each patch by a matrix of corner points. This type of representation does not allow for holes in patches, i.e., the patches must be *simply-connected*. If the gridding program has knowledge of the underlying numerical analysis program, it would realize that once it removes the fictional surface area from the hull, it must break the hull in half to "cut" out the hole. This cut can be performed in limitless ways, but how it is done affects how easily the reparametrization and distribution steps can be performed to satisfy the evaluation criteria.

In the following sections we present a geometric language, Boundary Surface Representation (BSR), which is capable of representing geometrical information, topological information as well as associating attributes of the physical domain to the geometry. Also, we present a principled method of solving the partitioning, reparametrization, and distribution problems based on reasoning about physics of the flow domain. We call this method streamline-based gridding.

3 Boundary Surface Representation(BSR)

Surfaces are basically two dimensional objects that reside in three dimensional space. So they are naturally represented parametrically as a mapping from parametric space, $(u, v) = ([0, \ldots, 1], [0, \ldots, 1])$, to 3D Cartesian space, (x, y, z). Our gridding system provides a mapping facility to represent this *shape* mapping, see Figure 2. No assumption is made about what mathematical form the mappings may take. Each mapping is treated as a "black box". The advantage of using a black box representation is that it provides greater flexibility by hiding the implementation details from the gridder. In our example, the hull is defined using algebraic formulae, and the keel and winglet are defined using B-spline surfaces.

This mapping facility is not limited to defining shapes. Other geometric and physical values may also be defined. For example, the outward normals of a surface may be defined as a *normal* mapping from the parametric space, (u, v), to 3D vector space. Then in turn based on the *shape* and *normal* mappings, our gridding program can approximate the stream vectors on the surfaces as a *flow* mapping by projecting the *free stream vector*, $(1, 0, 0)$, onto the surface. The free stream vector is the direction the water would flow if the yacht were not

present.

Notice the boundaries of each surface are represented explicitly by directed edges, *arcs*. The arcs in turn are bounded by nodes. Explicit representation of the boundary is useful in that it allows for implicit representation of surfaces. That is, a closed sequence of arcs in parametric space can be used to denote the portion of the surface it encloses. The program adopts the *counter-clockwise rule*. A counter-clockwise, closed sequence of arcs denotes the area bound by the arcs. A clockwise, closed sequence of arcs denotes the area outside of the arcs. This implies the area on the "left-hand side" of an arc is "inside," and area on the "right-hand side" is "outside."

Arcs are also useful in expressing topological information. In our notation two arcs are connected by a dotted line if they are the same line when mapped using *shape* into xyz-space, even though they are distinct in parametric space. For example, in Figure 2 the keel parametric arcs h ($u_{keel} = 0$) and f ($u_{keel} = 1$) are connected by a dotted line, because both of these arcs map to the trailing edge of the keel. Thus in xyz-space it is possible to travel just in the direction of increasing u_{keel} and end up at your starting point. This dotted line together with the dotted line connecting arcs d ($u_{keel} = [0, \ldots, 0.5]$) and e ($u_{keel} = [0.5, \ldots, 1]$) implies the topology of the keel is similar to that of a cylinder with one end closed or a "cup."

Notice that the hull parametric arcs b ($u_{hull} = 1$) and d ($u_{hull} = 0$) are connected to themselves. This is used to show that arcs b and d are degenerate, i.e., they each map to one point in xyz-space. The arc d maps into the trailing point of the hull; the arc b maps into the leading point of the hull.

Also, notice each of the arcs on the winglet is connected to some other arc. This means that in xyz-space the winglet surface does not have any boundaries. Of the three components the winglet is the only one that actually encloses some finite volume in xyz-space.

BSR provides a set of surface patch manipulation operations, such as intersection of surfaces, and division of patches into sub-patches. Figure 4 depicts the patches after the partitioning step. Reparametrization and distribution operations also are supported, see Figure 5. Now, we can formally define reparametrization as a mapping from a unit square, defined in a new parametric space, say (s, t), to a surface patch in (u, v) parametric space.

4   Streamline-based reasoning

The solution to Laplace's equation depends neither on the current state of the flow nor on time, so the geometry of the object determines the solution. Since streamlines are key characteristics of the solution, analyzing how streamlines interact with geometry provides key insights to qualitative behaviors of Laplace's equation. These insights enable us to determine the topology of streamlines. In turn this topology provides natural boundaries for patches in grids.

The most immediate reasoning problem we encounter in streamline-based reasoning is how to get the initial set of streamlines, since we have not yet run PMARC to generate the solution from which streamlines are extracted. We have experimented with various methods of predicting the streamlines *a priori*. However, we have found the simple projection of the *free stream* vector onto the body surface to be a good approximation of the true streamlines. This the *flow* mapping defined earlier.

4.1 Object classification

Analyzing the pattern of streamlines on the surface of different objects, we define two object classes. This first is the *source/sink node* class. Streamlines on objects from this class all originate from one point on the surface, the source node, and all flow to and terminate at another point on the surface, the sink node. Spheres, ellipsoids and other simple bodies of revolution are objects of this class. These objects have axial-symmetry, so there can only be one source node and one sink node.

The second is the *source/sink line* class. This class is like the previous class, except that the streamlines appear to originate and terminate at lines instead of nodes. For instance, the leading edge of a keel is *source line*, and the trailing edge is *Sink line*. All the streamlines flow from the leading edge to the trailing edge. Any wing shaped object belongs to this class.

Using only these two object classes, one can already construct complex, geometric objects, such as the yacht in this paper. The yacht consists of a source/sink node object (hull), and two source/sink line objects (keel and winglet). New classes can always be defined as the need arises.

4.2 Application to gridding

Based on the *following-streamline* heuristic for gridding, it is reasonable to grid a source/sink node object as a single surface patch, since all the streamlines are flowing in one direction, from the source node to the sink node. A source/sink line object should be gridded as two surface patches with the source line and sink line acting as partitioning lines. Although the streamlines still flow from the source line to the sink line, the streamlines take two different routes. For example, one set of streamlines flows to the sink from the right side of the keel ($u > 0.5$), and the other set flows from the left side ($u < 0.5$). The source/sink lines separate these two flow regions.

Streamlines are also useful in reparametrization. Streamlines can be defined as one family of grid lines. Lines orthogonal to the the streamlines can be defined as the other family. For example, on a sphere these two families correspond to the two spherical coordinate directions, θ and ϕ, where $x = \cos\theta, y = \sin\phi\cos\theta, z = \sin\phi\sin\theta$. Streamlines have constant θ and the orthogonal lines have constant ϕ.
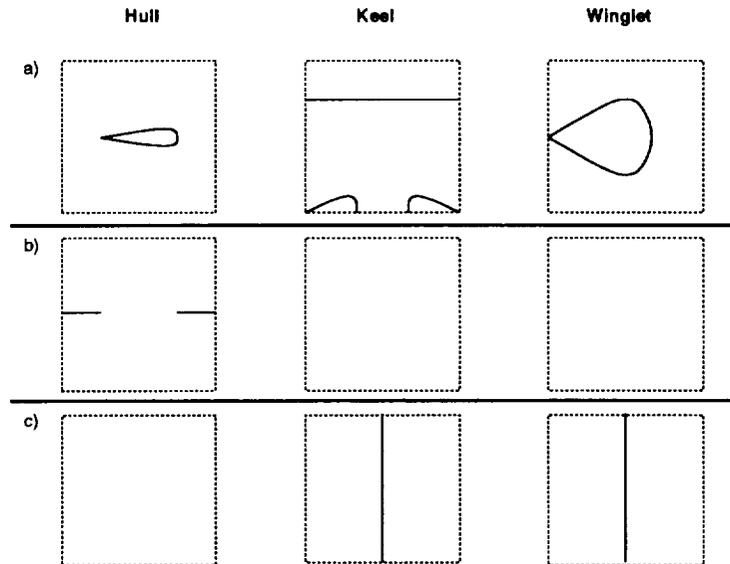
Figure 3: Partition lines: a) intersection lines, b) streamlines to "cut" holes out, and c) source/sink lines.

The sources and sinks provide guidelines on how to distribute the grid lines. The key to distributing grid lines is to highlight the physical features of the domain. That is, put more grid lines in regions where interesting physical changes occur. In the flow domain, the most interesting change is the change in direction and velocity of the flow. This change typically occurs most dramatically around the sources and sinks. So, the grid lines should be distributed more densely around them.

The above discussion deals with idealized objects. In the yacht example, there is a keel attached to the hull, and a winglet attached to the keel. The following sections show how to deal with the topological changes in these idealized objects by going through the three gridding steps in more detail.

5 Partitioning

We break the partitioning step into three sub-steps: 1) Determine the surface partitioning lines, 2) Partition surfaces into surface patches, and 3) Determine real surfaces patches.

5.1 partitioning lines

The partitioning lines that we use can be divided into three categories: surface intersection lines, streamlines, and source/sink lines. Intersection lines provide

588

the boundary between real and fictional surface areas, so they must be present. See Figure 3a for examples.

Notice that the hull-keel intersection line introduces a hole on the hull surface. This hole needs to be cut out, because of the *simply-connected* constraint. Using streamline-based reasoning, the logical way to "cut" out the hole is by cutting along streamlines. We search for a leading point and a trailing point along the intersection. From the leading point we trace a streamline *backward* along the hull surface. From the trailing point we trace a streamline *forward* along the hull surface. These two streamlines are shown in Figure 3b.

Source and sink lines are definitely needed, but all the sink lines turn out to be redundant. The source lines are shown in Figure 3c. Notice that source/sink nodes in xyz-space may become source/sink lines in uv parametric space, as in the hull.

5.2 partition the surface patches

We shall not go into detail on how BSR accomplishes the actual partitioning. Basically BSR 1) gathers all the partition lines of a particular surface, 2) intersects the partition lines with each other and with the boundary lines of the surface, 3) breaks all the lines at intersections, 4) forms a wire frame from the broken lines, and 5) forms the surface patches based on the wire frame. The surface patches after partitioning are shown in Figure 4. BSR updates the topological information after the partitioning process. The shaded surface patches are fictional and will not be gridded.

5.3 determine the real surface patches

Real and fictional surface patches can be distinguished by reasoning using the outward *normal* mappings, the *counter-clockwise* rule, and intersection lines. For example, the surface patch $Keel1$, Figure 5, has intersection lines in common with the hull surface (arc 6) and the winglet surface (arcs 2 and 3). The hull outward *normals* along the hull-keel intersection generally point in the negative z-direction. The inward direction of arc 6 as defined by the counter-clockwise rule is in the negative V_{keel} direction, which corresponds to the negative z-direction in xyz-space. This implies that $Keel1$ is outside of the hull. Similar reasoning using arcs 2 and 3 shows $Keel1$ is outside of the winglet. Since $Keel1$ is on the outside of all its neighbor surfaces, $Keel1$ is a real surface patch. If a surface patch is on the inside of one or more of its neighbors, then it is a fictional patch.
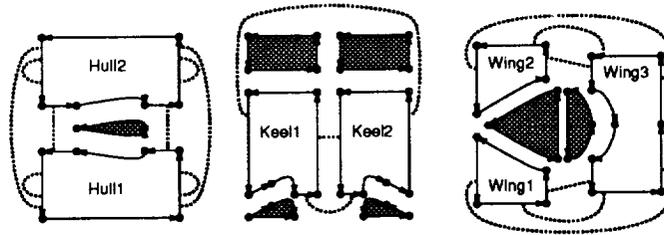
Figure 4: Surface patches after partitioning. Dotted lines across uv-space are not drawn to reduce clutter. Omitted dotted lines would show Keel1 connected to Hull1, Wing1 and Wing3, and would show Keel2 connected to Hull2, Wing2 and Wing3.

6 Reparametrization

Our gridding program is able to invoke several different reparametrization methods, including *transfinite interpolation*, *elliptic* methods, and *variational* methods, [Knupp & Steinberg 1993]. These methods map a unit square on to a quadrilateral. In the case of transfinite interpolation, the mapping is done by interpolating from opposite edges of that quadrilateral. All these method requires the surface patch to be reparametrized to have exactly four sides. But, surface patches tend to have more than four boundary edges, and sometimes less than four. In order to use these methods, we describe a heuristic, streamlined-based method of grouping the boundary arcs of the surface patches into four groups. See Figure 5.

We can classify each arc as either parallel, anti-parallel, or orthogonal with respect to the streamlines. For example, the patch $Keel1$ is bounded by six arcs. Arc 1 is a sink line. Arc 5 is a source line. So, by definition they are orthogonal to the streamlines. Arc 4 is a boundary arc from the original input surface. Arcs 2, 3 and 6 are intersection lines. These four arcs are neither completely parallel nor completely orthogonal to the streamlines. But by sampling different segments of these arcs, we can approximately classify arcs 2 and 4 as parallel, 6 as anti-parallel, and arc 3 as orthogonal. So, six groups are formed, $[(1), (2), (3), (4), (5), (6)]$. But, unlike the graphical depiction in Figure 5, arc 3 is very short when compared to its neighbors, arc 2 and 4. So, heuristically merging arc 3 with its neighbors, we get four groups, $[(1), (2, 3, 4), (5), (6)]$. We call this a *valid sequence* of groups, because it follows a [orthogonal, parallel, orthogonal, anti-parallel] pattern.

If there are not enough arcs to form four groups, then additional degenerate, point arcs may be introduced to construct a valid sequence of groups. For example, in the case of three arcs with classification [orthogonal, orthogonal, anti-parallel], then a parallel, point arc may be add between the two orthogonal
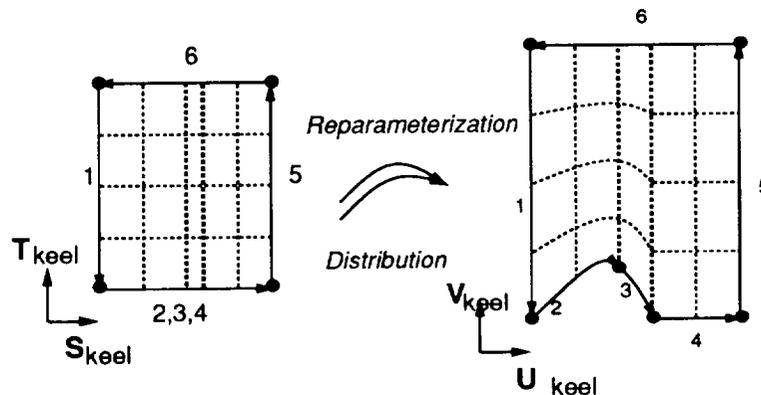
590

Figure 5: Reparametrization and Distribution

arcs.

Our grouping method works well, because the boundary arcs of the surfaces patches tend to be partitioning lines: intersection lines, streamlines, and source/sink lines. Classification of streamlines and source/sink lines are straightforward. In practice intersection lines tend always to be parallel, because an orthogonal intersection line causes too much drag, and would not be used in properly designed streamlined bodies.

This reparametrization step may fail in two cases. The first case occurs when the patch is inherently non-rectangular (e.g., a L-shaped surface patch), and the heuristic method fails to group the boundary arcs into four groups. The second case occurs when the patch is highly non-convex, and the repamaterization method fails to converge or returns a folded grid. In both case the geometry of the surface patch is too complicated, and additional partitioning lines are needed.

7 Distribution

According to streamline-based reasoning, grid lines should be concentrated more densely around sources and sinks. Sources and sinks tend to be at the ends of the surface patches (in Figure 5 arc 1 and arc 5) in our streamline-based gridding method. So, complicated distribution schemes usually are not needed. We have experimented with *cosine* and *hyperbolic tangent* schemes, which distribute more grid lines at the ends and yet distribute them smoothly enough as not to violate the *expansion ratio* constraint. Both schemes work well, but if many grid lines are laid out, *cosine* tends to place grid lines too densely at the ends. This leads to numerical truncation error.

Beside resolving physical features, distribution must also resolve geometric

features. For example, one $S_{keel} = constant$ grid line must be laid out at the intersection of arc 2 and arc 3, and another one grid line at the intersection of arc 3 and arc 4. Grid lines that must be laid out are shown as heavy, dotted lines in Figure 5. The node at the intersection of arc 3 and arc 4 touches three surface patches, $Keel1$, $Keel2$, and $Wing3$. Not laying a grid line at that node would create a gap there so the three patches would not meet.

8 Computational Results

Our gridding algorithms have been implemented in a working program. Figure 6 shows the results of a convergence study in which our gridding program generated a series of grids for PMARC. In the convergence study we ran a series of simulations using grids with the same partitioning and reparametrization, but with increasingly denser grid lines. As the grid becomes denser and grid spacing decreases, output quantities computed by PMARC should converge to their correct values. The output quantity we are most interested in is effective draft, a measure of the efficiency of a sailing yacht's keel. Figure 7 shows how effective draft converges as grid spacing is reduced in our convergence study.

Other values in Figure 6 can also be used as checks on the soundness of the simulation. For example, the maximum C_p (pressure coefficient) should approach 1 as the grid is refined, and the minimum C_p should not become too negative, as very large negative values usually indicate flaws in the grid. [Gelsey 1992] discusses automated evaluation of simulation output quality.

Panels	Draft	Drag	Lift	min C_p	max C_p
286	2.8836	0.0951	-0.2879	-3.1832	0.6191
938	2.5869	0.1255	-0.2532	-4.6730	0.7941
3678	2.4655	0.1479	-0.2418	-4.1283	0.9010
14898	2.4327	0.1676	-0.2139	-4.9392	0.9631

Figure 6: Convergence study

9 Future Work

This work can be extended in various directions. One is to add feedback and local refinement capabilities to the gridder. The streamlines predicted by PMARC may be fed back into the gridder to improve the grid. Also, the gridder can be extended to detect and correct local flaws in the grid based on intermediate values, such as the coefficient of pressure. Another direction is to extend the gridder to other physical domains where PDE simulators are needed. We believe our methodology of identifying key physical features of the domain and of reasoning about how they interact with the geometry is quite general and extensible. For example, in the ingot casting problem of heat transfer the temperature profile
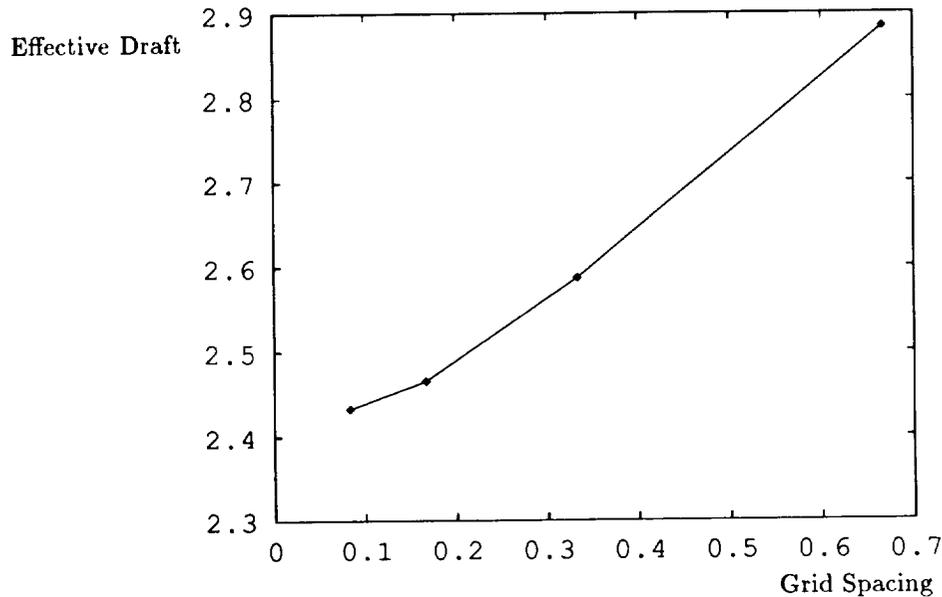
Figure 7: Convergence as spacing is reduced

seems to be the key feature [Ling, Steinberg, & Jaluria 1993]. Temperature profiles tend to change the fastest near sharp corners and in appendages (regions where the surface area to volume ratio is large). This suggests that isotherms should be useful as grid lines, and they should be distributed more densely near corners and appendages.

10 Related Work

Using streamlines is a natural idea. [Chung, Kuwahara, & Richmond 1993] defines a 2D finite-difference method based on streamline-coordinates, instead of Cartesian coordinates. [Chao & Liu 1991] applies streamline-based gridding to 2D flow problems consisting of a single patch. Many geometric modeling systems have been developed, such as Alpha1 by [Riesenfeld 1981] and SHAPES by [Sinha 1992]. [Requicha 1980] provides a good survey. Most of these systems are intended for modeling mechanical components, and provide little support for gridding, like representation of parametric space objects for reparametrization and distribution, and algorithms to manipulate these objects. Previous AI work in gridding includes [Dannenhoffer 1992], and [Santhanam *et al.* 1992]. In the 2D planar flow domain, Dannenhoffer's program is able to do partitioning by merging *templates* of previously-solved cases. So, the set of shapes it can handle is limited. Santhanam identifies several key parameters to modify and improve grids in 1D Euler domain.

11 Conclusion

Numerical simulation of partial differential equations is a powerful tool for engineering design. However, human expertise and spatial reasoning abilities are needed in order to form the spatial grids which PDE solvers require as input. We have developed a geometric modeling language, BSR, capable of expressing geometrical, topological, and physical aspects of the gridding problem, and we have used BSR as a basis for an intelligent automated system for generating the grids required for numerical simulation. The grid generation process involves analyzing the topology of the spatial domain, predicting and classifying the interactions of physics and geometry, and reasoning about the peculiarities of the numerical simulator.

12 Acknowledgments

References

[Chao & Liu 1991] Chao, Y. C., and Liu, S. S. 1991. Streamline adaptive grid method for complex flow computation. *Numerical Heat Transfer, Part B* 20:145-168.

[Chung, Kuwahara, & Richmond 1993] Chung, S. G.; Kuwahara, K.; and Richmond, O. 1993. Streamline-coordinate finite-difference method for hot metal deformations. *Journal of Computational Physics* 108:1-7.

[Dannenhoffer 1992] Dannenhoffer, J. F. 1992. Automatic block-structured grid generation — progress and challenge. In Kant, E.; Keller, R.; and Steinberg, S., eds., *AAAI Fall Symposium Series: Intelligent Scientific Computation*, 28-32.

[Ellman, Keane, & Schwabacher 1992] Ellman, T.; Keane, J.; and Schwabacher, M. 1992. The Rutgers CAP Project Design Associate. Technical Report CAP-TR-7, Department of Computer Science, Rutgers University.

[Gelsey 1992] Gelsey, A. 1992. Modeling and simulation for automated yacht design. In *AAAI Fall Symposium on Design from Physical Principles*, 44-49.

[Gelsey 1994] Gelsey, A. 1994. Automated reasoning about machines. *Artificial Intelligence*. To appear.

[Kao & Su 1992] Kao, T. J., and Su, T. Y. 1992. An interactive multi-block grid generation system. In Smith, R. E., ed., *Software Systems for Surface Modeling and Grid Generation*, number 3143 in NASA Conference Publication, 333–345.

[Knupp & Steinberg 1993] Knupp, P. M., and Steinberg, S. 1993. *The fundamentals of grid generation*. CRC Press.

[Ling, Steinberg, & Jaluria 1993] Ling, S. R.; Steinberg, L.; and Jaluria, Y. 1993. MSG: A computer system for auotmated modeling of heat transfer. *AI EDAM* 7(4):287–300.

[Remotique, Hart, & Stokes 1992] Remotique, M. G.; Hart, E. T.; and Stokes, M. L. 1992. EAGLEView: A surface and grid generation program and its data management. In Smith, R. E., ed., *Software Systems for Surface Modeling and Grid Generation*, number 3143 in NASA Conference Publication, 243–251.

[Requicha 1980] Requicha, A. A. G. 1980. Representations for rigid solids: Theory, methods, and systems. *Computing Surveys* 12(4):437–464.

[Riesenfeld 1981] Riesenfeld, R. F. 1981. Using the oslo algorithm as a basis for CAD/CAM geometric modeling. In *Proc. NCGA National Conf.*, 345–356.

[Santhanam et al. 1992] Santhanam, T.; Browne, J.; Kallinderis, J.; and Miranker, D. 1992. A knowledge based approach to mesh optimization in CFD domain: 1D Euler code example. In Kant, E.; Keller, R.; and Steinberg, S., eds., *AAAI Fall Symposium Series: Intelligent Scientific Computation*, 115–118.

[Schuster 1992] Schuster, D. M. 1992. Batch mode grid generation: An endangered species? In Smith, R. E., ed., *Software Systems for Surface Modeling and Grid Generation*, number 3143 in NASA Conference Publication, 487–500.

[Sinha 1992] Sinha, P. 1992. Mixed dimensional objects in geometric modeling. In *New Technologies in CAD/CAM*.

[Thompson, Warsi, & Mastin 1985] Thompson, J. F.; Warsi, Z. U. A.; and Mastin, C. W. 1985. *Numerical grid generation : foundations and applications*. North-Holland, Amsterdam.