S2-61

N91-21064

p. 18

GENERATION AND USE OF UNSTRUCTURED GRIDS FOR TURBOMACHINERY CALCULATIONS

Dana R. Lindquist
Michael B. Giles
Massachusetts Institute of Technology
Cambridge, MA 02139

ORIGINAL CONTAINS
COLOR ILLUSTRATIONS

MJ 700802

ABSTRACT

This paper presents a wavefront mesh generator for two dimensional triangular meshes as well as a brief description of the solution method used with these meshes. The interest is in creating meshes for solving the equations of fluid mechanics in complex turbomachinery problems, although the mesh generator and flow solver may be used for a larger variety of applications. The focus is on the flexibility and power of the mesh generation method for triangulating extremely complex geometries and in changing the geometry to create a new mesh. Two turbomachinery applications are presented which take advantage of this method: the analysis of pylon/strut and pylon/OGV interaction in the bypass of a turbofan.

MESH GENERATOR

In recent years the use of unstructured triangular meshes in computational fluid dynamics has grown in popularity. The main reason for going to triangular cells is the ability to compute the flowfield around complex geometries since in these cases it is easier to create a triangular mesh than a quadrilateral mesh, if a quadrilateral mesh can even be created. The first question that one encounters when deciding to work with triangles is how to create the mesh. This paper presents a mesh generator which was originally based on the work of Lo [1] and extended by Peraire et al [2,3]. The method works by advancing a front through the domain to be triangulated, creating points as they are needed. The result is a very powerful and flexible mesh generator.

The computational domain is specified as a number of closed objects inside an outer boundary. Each of these curves are defined as piecewise cubic splines based on a set of points given by the user. These boundaries are divided into segments which represent a node distribution on the boundary and compose the initial front. This front then moves into the domain to be triangulated, creating triangles as it goes. A new triangle is made up of two nodes on the front and either a newly created node or another node on the front. The front moves inward until it totally collapses in on itself. An example of the process is shown in Figure 1 where the wave front is denoted by the darker line.

For the mesh generator to determine the size of the triangles throughout the region, a desired mesh size must be given for each point in the triangulated region. Here this is done by creating a background mesh of large triangles with mesh parameters given at the nodes. A general point in the region will lie inside a background mesh triangle, and the local mesh parameters are found by a linear interpolation of the values at the background nodes.

Some less desirable triangles can be created by the process, for example where the front finally collapses, therefore the final mesh is smoothed. The smoothing of the final grid is done in two steps. First, the triangulation is made to conform to the rules for a Delaunay mesh. This is done by examining all sets of two adjacent triangles. The face between the triangles is rotated if the current face location is on the longer diagonal as shown in Figure 2. Next the nodes are displaced slightly so the mesh is relaxed. The new location of a node is given by

$$x_i^{new} = x_i + \frac{\omega}{n}\sum_{k=1}^{n}(x_k - x_i) \tag{1}$$

where i is the node to be smoothed and the sum is over the n nodes surrounding node i. A value of $\omega = 0.3$ is used for the relaxation factor. The combination of these two steps in smoothing the grid is quite effective in eliminating any undesirable cells as can be seen in Figure 1.

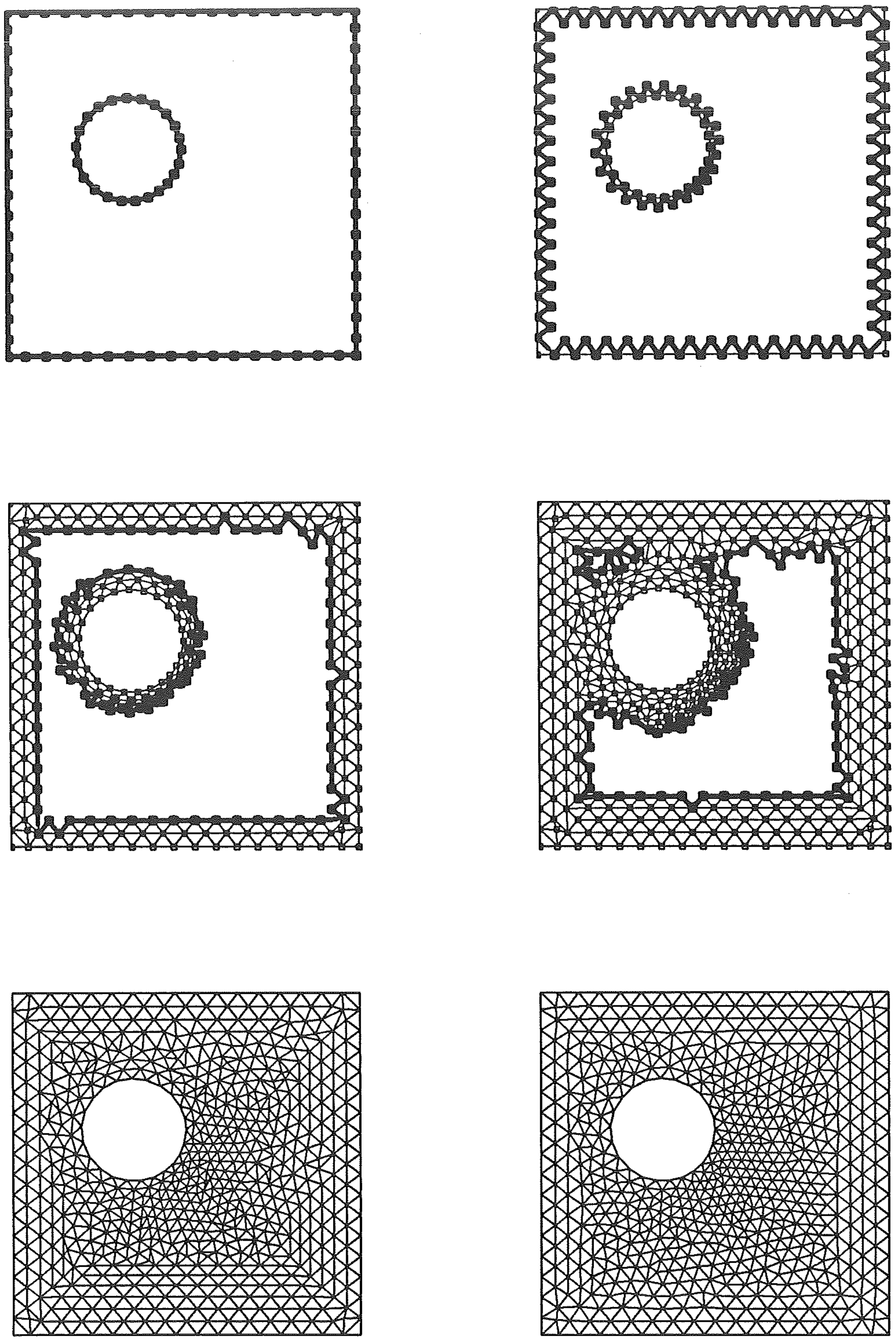


Figure 1: Mesh at various stages in the advancing front method.

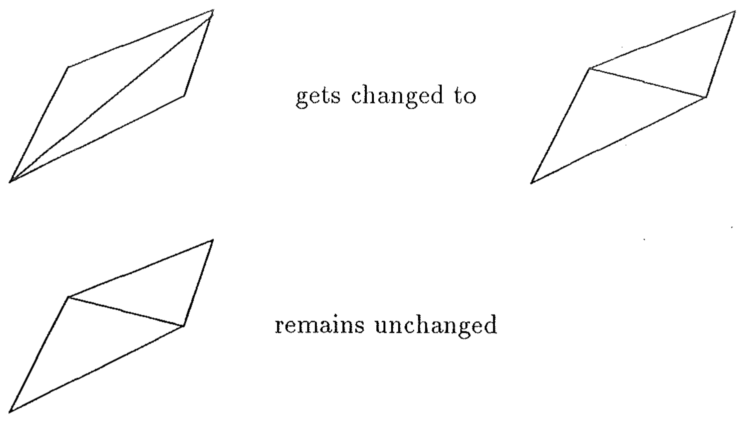


Figure 2: Mesh smoothing to create Delaunay triangulation.

The focus of the discussion here will be on the flexibility and power in generating grids for extremely complex geometries with very little user time required in the definition of the problem. Interested readers are referred to the papers by Peraire et al [2,3] for a discussion on the exact algorithm for front advancement. It would suffice to say that the 2D mesh generator consists of over 3000 lines of code of which most involve conditional statements. The only part of the code which will vectorize is a few lines involving a search. Peraire et al [4] have successfully extended this method to three dimensions. The biggest difficulty from a user interface point of view in three dimensions is setting up a background mesh size distribution. Instead of using a background mesh, some other method which defines the distribution on the boundaries and smooths it through the region could be used.

To illustrate the ease of creating a mesh, the specifics involved in creating the mesh in Figure 1 will be shown as an example. As previously stated, the outer boundary of the domain and the objects inside the domain are specified by the user as a set of point which define a closed curve. Two other files are required, one which describes the orientation of the objects and another which describes the background mesh.

Objects:

Here, two objects are used: a square which is defined from $x = -1$, 1 and $y = -1$, 1, and a circle with center at the origin and radius 1. The points defining these objects are in files OBJECT.SQUARE and OBJECT.CIRCLE respectively.

The object files contain additional information about the type of boundary condition to be applied to segments of the boundary. The boundary condition could correspond to an inlet, outlet, solid wall or a set of periodic surfaces.

Orientation:

The file FGRID.SQUARE is shown below along with the geometry for the problem. This file describes the orientation of the objects in the domain. The several lines of the file are:

Line 1: title for the mesh

Line 2: name of the object file which in this case is OBJECT.SQUARE

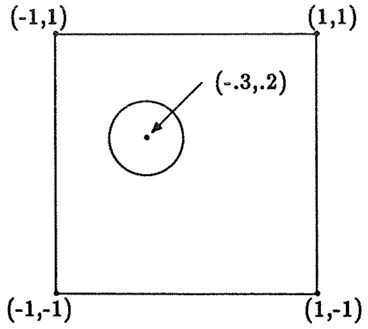
Line 3: tells whether the mesh will be created inside or outside the object where a positive number means the mesh is inside

Line 4: four real values which give the factor by which the data is scaled, the angle by which the data is rotated about the origin in the data, and the amount in the x and y directions by which the data is translated

Last 3 lines: the same kind of information given above about the data in file OBJECT.CIRCLE

FGRID.SQUARE

```
circle inside a square
OBJECT.SQUARE
1
1. 0. 0. 0.
OBJECT.CIRCLE
-1
0.3  0. -.3  .2
```



It is clear that objects can be easily rotated, scaled and placed anywhere in the domain. This means it is easy to change the location of an object or drop another object into the domain.

Background:

The information about the background mesh for this case is given in file BACK.SQUARE which is shown below. This file has information giving the node locations and the mesh parameters at the nodes as well as the way the nodes are connected to create a mesh. The several lines of the file are:

Line 1: number of nodes in the background mesh which in this case it 5

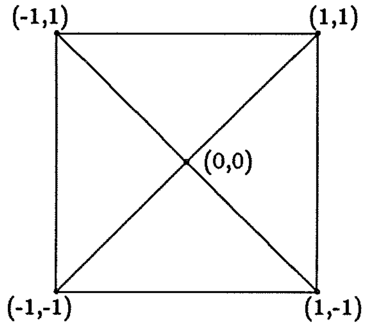
Next 5 lines: each node is given a consecutive numbers from the top of the list and has three real values which give the x and y locations of the nodes and the desired standard length of the cell

Line 7: number of cells in the background mesh which in this case is 4

Next 4 lines: each line contains the three node numbers listed counter-clockwise which make up the background cells

BACK.SQUARE

```
  5
-1.  -1.   .1
 1.  -1.   .1
 1.   1.   .1
-1.   1.   .1
 0.   0.  .05
  4
  1    2    5
  2    3    5
  3    4    5
  4    1    5
```



The current version of the code requires the user to set up the background mesh by hand. This turns out to be a little tedious when a complicated mesh size distribution is desired. In the future an interactive method of creating the background mesh will be created where the user can place and move points using a mouse and connect these points again using a mouse.

A few other examples of meshes created using this method are shown in Figures 3 and 4. In Figure 3 a very unusual mesh was created. The point which should be most emphasized in connection with this example is that once the object files were created for the letters C, F and D, it was a simple matter to get the mesh. In Figure 4 a more standard mesh was created about a T7 turbine blade. The background mesh is shown to illustrate how course a background mesh can be to get a realistic cell size distribution. It can be seen that resolution is obtained around the leading edge where it is needed, but the cells are coarser in the rest of the region.

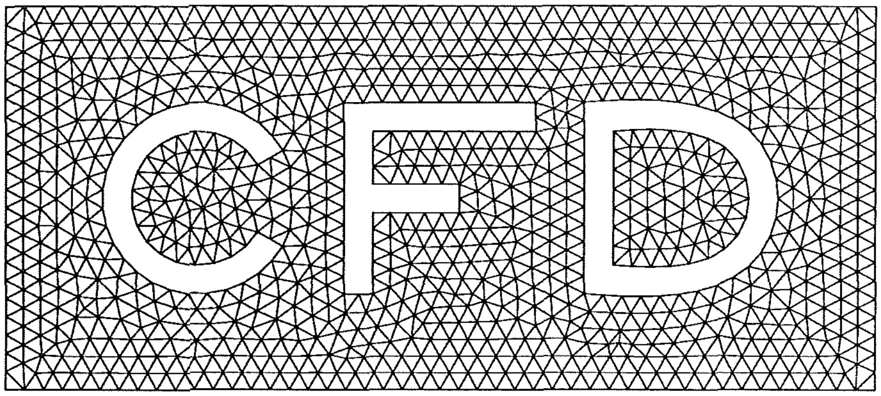


Figure 3: Mesh around an unusual configuration.

FLOW SOLVER

The calculations were performed using UNSFLO [5], a Ni-type, Lax-Wendroff, Euler program which has the capability of computing on an unstructured triangular or quadrilateral mesh, or a mesh consisting of a mix of these cell types. The algorithm is fairly straight forward, but the numerical smoothing used has been found to have a large effect on the accuracy of the solution. The dissipative operator which we call numerical smoothing is composed of two parts, a fourth difference operator throughout the flowfield and another operator which is required to capture shocks and other discontinuities. Due to its importance to the accuracy of the solution, the fourth difference operator will be discussed here.

To compute the fourth difference operator, a second difference of a second difference is found. These operators are not necessarily the same, and in UNSFLO they are not. The first is a relatively simple operator which gives a non-zero second difference for a linear function on an irregular mesh. The second operator is more complex, but results in a zero second difference for a linear function. By examining the effect of the second difference operator on a linear function the accuracy of the operator is tested, since for second order or higher accuracy the contribution must be zero.

A typical cell is shown in Figure 5 with corresponding nodes labeled 1, 2 and 3. The operators will be described in terms of a contribution from the cell to one of the nodes. The total contribution to a node comes from all the cells surrounding the node.

The low-accuracy second difference operator is not dependent on the location of the nodes surrounding the node for which the second difference is computed, but merely on the function values at these nodes. For a triangular mesh the contribution from cell A to the second difference at node 1 is

$$(D^2S)_{1A} = (S_3 + S_2 - 2S_1) \tag{2}$$

where S is the variable for which the second difference is computed. This second difference is conservative since the total contribution of each cell to its nodes is zero.

The high-accuracy second difference operator consists of finding the first derivative for each cell and then combining the derivatives on the cells surrounding a node to form a second difference. Unlike the low-accuracy second difference operator, this operator is dependent on the mesh geometry. Referring to Figure 5 the first derivative with respect to x is found for cell A

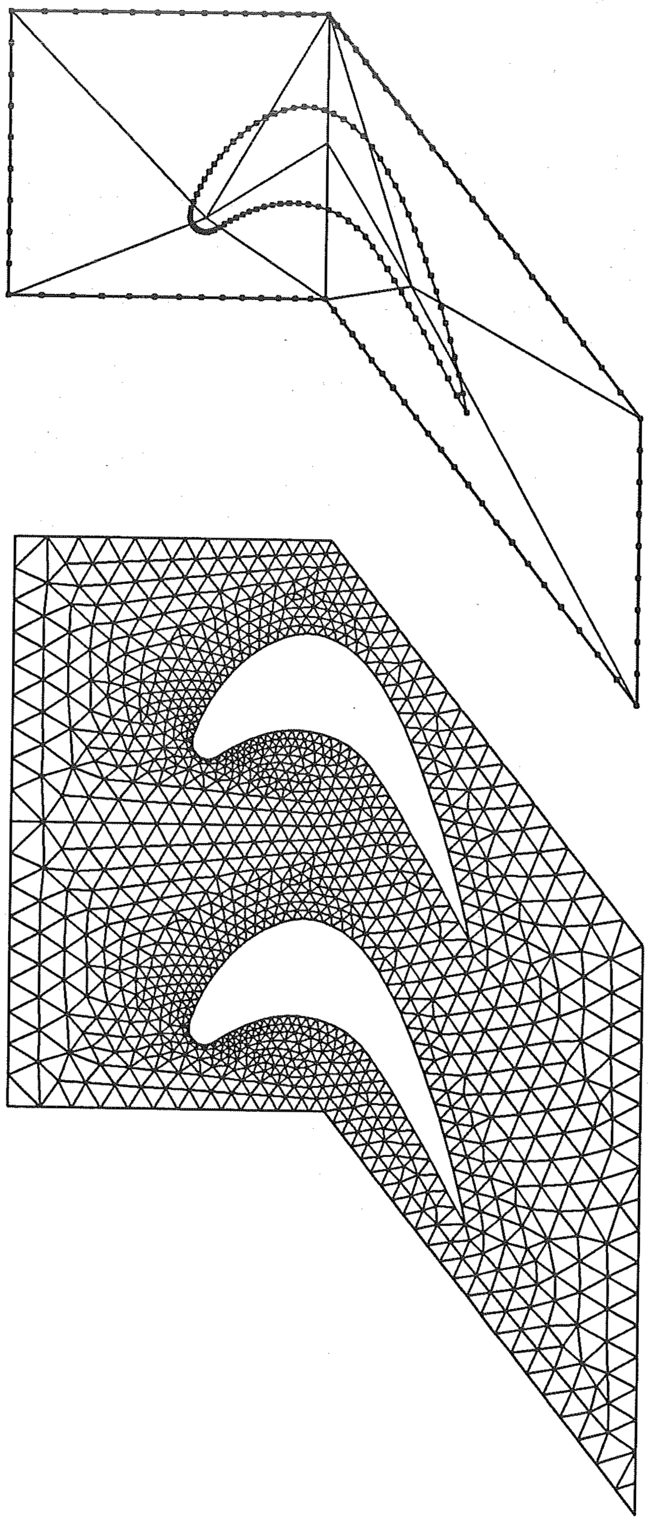


Figure 4: Mesh around T7 turbine blade and background mesh. The mesh contains 1127 cells and 652 nodes.

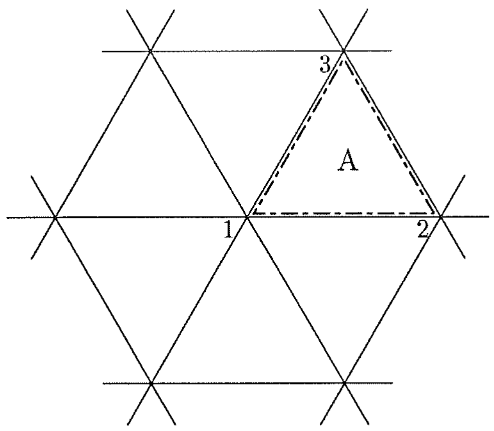


Figure 5: Triangular cell

$$\begin{aligned}
(S_x)_A &= \frac{1}{A_A}\iint_{cell A}\frac{\partial S}{\partial x}dx\,dy \\
&= \frac{1}{A_A}\int_{1-2-3} S dy \\
&= \frac{1}{2\,A_A}[(S_2+S_1)(y_2-y_1)+(S_3+S_2)(y_3-y_2)+(S_1+S_3)(y_1-y_3)] \qquad (3)
\end{aligned}$$

and similarly for the derivative with respect to y. A similar process is performed to create a second difference. The integration is taken around all the triangles which surround the node for which the second difference is computed, using the derivative values calculated at the cells. To get a second difference instead of a second derivative, there is no division by the area of the integrated region. The contribution to the second difference at node 1 from cell A is

$$\begin{aligned}
(D^2S)_{1A} &= \int_{2-3}[(S_x)_A dy - (S_y)_A dx] \\
&= (S_x)_A(y_3-y_2)-(S_y)_A(x_3-x_2) \qquad (4)
\end{aligned}$$

This second difference operator is conservative since again the total contribution of each cell is zero, but unlike the previous operator it is second order accurate.

On all boundaries, solid wall or farfield, boundary conditions must be implemented for the second difference operators. For the low-accuracy operator the contribution to a node on the boundary is simply the contribution from the cells surrounding that node which are inside the domain as given in Equation (2). The high-accuracy operator involves a line integral in Equation (4) which must be closed when considering a node on the boundary. To do this the integral is continued along the boundary faces on either side of the node in question using the value of (S_x) and (S_y) from the cell directly inside the boundary.

Clearly, to get the fourth difference operator either of these second difference operators could be used. Two methods were examined by Lindquist [6,7]. The first method is to use the low-accuracy second difference twice by operating first on the state vector and then operating on this second difference. This fourth difference is conservative, but is second order accurate only on a uniform mesh since the second difference operator used is only second order accurate on a uniform mesh. The second method is to compute a second difference of the state vector using the high-accuracy method and operate on this second difference with the low-accuracy second difference. This operator is second order accurate since the first operator is second order accurate and conservative since the second operator is conservative.

The second method is more expensive than the first, but the effect per iteration is an increase of only 5-10% which is a small increase for the gain in accuracy. The fourth difference is multiplied by a coefficient, between 0.0001 and 0.01, to control the amount of smoothing which is added to the scheme.

A modification to the high accuracy second difference operator was made by Holmes and Connell [8] which is particularly useful in creating a more accurate operator when the aspect ratio of the cells is high. This modification adds a weighting factor the the operator which is based on the geometry of the mesh.

APPLICATIONS

There are applications in turbomachinery where a standard grid generator is useless. In particular, two problems arise: the grid generator assumes a particular geometry configuration or grid clustering is desired in a specific locality. Two cases will be discussed here which involve both these situations. The problem is the analysis of pylon/strut and pylon/OGV interaction in the bypass of a turbofan. The configuration is clearly complex and high grid resolution is only required around the struts or OGVs, particularly in the leading edge region. In both cases UNSFLO, the solver described in the previous section, was used.

The first application is a pylon/strut interaction. The geometry for this case is similar to that published in Reference [9]. Figure 6 shows the computational domain, which represents the complete bypass annulus unwrapped into a two-dimesnional domain. At the center is the top pylon, and at the top and bottom of the domain are the two surfaces of the bottom pylon plus a section of periodic boundary. The eight struts all have a NACA 0012 profile and are inclined at angles of $-10°$, $0°$, $7°$, $20°$, $-20°$, $-7°$, $0°$, $10°$, listed from top to bottom. The specification of this geometry required just three files, one describing the outer boundary (the two pylons plus the inflow, outflow and periodic boundaries), one describing a NACA 0012 airfoil of unit chord, and one describing how the full gemoetry is composed from these two files by the appropriate scaling, rotations and translations. To change the inclination of any of the struts requires only a minor change to the last file. Figure 6 also shows the background mesh which is used to control the mesh spacing, as well as the initial grid points on the boundaries. The final mesh shown in Figure 6 has cells which are three times the size used in the computation since the final mesh spacing is quite small, but the blowup if the blade nearest the pylon is of the actual mesh. For obvious reasons, the mesh spacing has been controlled so that the grid is very fine around the struts, particularly their leading edges, and is fairly coarse around the pylons. Overall, this case uses approximately 120,000 cells and 60,000 nodes.

The high stagger angles of the struts was chosen to generate lift such that the lift-related potential field of the struts would approximately cancel the blockage-related potential field of the pylons, and hence reduce the unsteady upstream interaction with the fan. The flow calculation, with the inflow being at zero incidence and $M_\infty = 0.386$ reveals a problem with this approach. Because of the high lift, each of the two struts which are most inclined has a strong normal shock with a peak Mach number of 1.8 just behind the leading edge as shown in Figures 7, 8 and 9. The loss at this shock is so large that the accompanying vorticity leads to an inviscid separation near the trailing edge which is shown in Figure 10. The strong shock loss in this case suggests that it is a better idea to design the struts to be non-lifting, and instead tailor the OGV's, which are just upstream of the struts but are not included in this calculation, to prevent the potential field of the pylons from interacting with the fan.

The second application is a pylon/OGV interaction. The proper geometry in this case has 28 OGV's around the annulus but to reduce the computational cost the calculation was performed with 14 OGV's, maintaining the size and position of the OGV's and the pylon, and therefore doubling the relative blockage effect of the pylon. There are two reasons for presenting this case. The first is that it is another complicated example of unstructured grid generation with good control over grid spacing which varies by over two orders of magnitude. The second reason is that the results of the flow calculation exhibit a self-excited propagating flow instability which greatly ressembles rotating stall. Rotating stall has previously been calculated in two dimensions by a coupled vortex-boundary layer method [10] and by a Navier-Stokes calculation [11]. In the former case a propagating stall cell with large blockage was calculated, whereas in the latter case the blockage was not very severe and could almost be described as an unsteady boundary layer separation rather than a propagating stall. These two computations used five blade passages, which clearly places some restrictions on the stall cell character due to the periodicity constraints. In comparison to these other two calculations, the present calculation is believed to be the first solving the Euler equations which predicts a large stall cell blockage, and uses sufficiently many blade passages that the effect of the periodicity assumption is believed to be minimal.

The flow calculation was begun in a steady-state mode in which the Euler equations are time-marched using local timesteps. This did not converge to a steady-state, but produced results that looked very much like a single rotating stall cell. The computation was then switched into a time-accurate unsteady mode with a constant timestep, and continued until it settled into a periodic solution in which there were two very similar 'stall cells', approximately three blade passages in size. Figure 12 shows contour plots of entropy at four different instants. It clearly shows the high entropy of the 'stalled' fluid and the downward propagation of the 'stall cell'. Figure 13 shows an enlarged view of the 'stall cell' with velocity vectors at each grid point.

Although it must be emphasised that this is an inviscid calculation, the basic unsteady propagation mechanism is very similar to that of a two-dimensional stall cell [12]. As the stall cell approaches a particular vane, the blockage due to the reduced mass flow through the stalled passage causes an increase in the flow incidence on the new vane. This increased incidence leads to a strong normal shock, and the vorticity this produces leads to an inviscid separation. As the incidence increases, the shock strengthens and both the shock and the separation point move forward towards the leading edge until it develops into a leading edge separation with a free shear layer. The transfer of momentum across the shear layer due to numerical smoothing causes the separated fluid to grow into a strong passage. This is the part of the cycle which is probably most incorrectly modelled by the Euler equations; in reality, the retarding viscous force at the blade surface would prevent the growth of such a strong vortex. In the computation the passage vortex grows until it blocks most of the passage. At this time, the passage is near the rear of the rotating stall cell. The blockage due to the other stalled passages now reduces the incidence and supresses the leading edge separation. The flow reattaches at the leading edge and then progressively drives the passge vortex downstream, and the passage returns ultimately to its unstalled state. The ratio of the stall propagation speed to the mean inflow axial velocity is approximately 0.3, a value which is within the range of experimental data for stall propagation.

CONCLUSIONS

The wavefront method of mesh generation has been found to be extremely powerful. It provides a straightforward method of defining the geometry of the computational domain and the ability to easily modify that geometry. The variation of cell size can be specified and changed to fit the current problem. Most of the complexity of the mesh generator is in the code which lets the user concentrate on the current application. Only two dimensional problems are described here, but the method has been successfully extended to three dimensions by Peraire et al [4]. In complicated cases the majority of the user time spent in grid generation is in the specification of the background mesh controlling the grid spacing. It is thought that in three dimensions this may become a problem which deserves attention.

The two applications which are presented demonstrate the ability to analyze complex geometries. The mesh generator and the flow solver for these problems were the same as would be used for a single blade problem. This greatly reduces the need for several flow solvers or grid generation codes. The flowfield solutions provide useful insights into the design of bypass struts and the calculation of rotating stall.

REFERENCES

[1] S. H. Lo, "A New Mesh Generation Scheme for Arbitrary Planar Domains," *International Journal for Numerical Methods in Engineering* Vol. 21, pp. 1403-1426, 1985.

[2] J. Peraire, M. Vahdati, K. Morgan and O. C. Zienkiewicz, "Adaptive Remeshing for Compressible Flow Computations," *Journal of Computational Physics*, Vol. 72, No. 2, October 1987.

[3] J. Peraire, J. Peiro, K. Morgan and O. C. Zienkiewicz, "Finite Element Mesh Generation and Adaptive Procedures for CFD," GAMNI/SMAI Conference on Automated and Adaptive Mesh Generation, October 1-2, 1987.

[4] J. Peraire, J. Peiro, L Formaggia, K. Morgan, O. C. Zienkiewicz, "Finite Element Euler Computations in Three Dimensions," *AIAA 26th Aerospace Sciences Meeting*, AIAA-87-0032, January 1988.

[5] M. B. Giles, "UNSFLO: A Numerical Method For Unsteady Inviscid Flow In Turbomachinery," Technical Report #195, MIT Gas Turbine Laboratory, October 1988.

[6] D.R. Lindquist, "A Comparison of Numerical Schemes on Triangular and Quadrilateral Meshes," SM thesis, Massachusetts Institute of Technology, May 1988.

[7] D. R. Lindquist and M. B. Giles, "A Comparison of Numerical Schemes on Triangular and Quadrilateral Meshes," 11th International Conference on Numerical Methods in Fluid Dynamics, June 1988.

[8] D. G. Holmes and S. D. Connell, "Solution of the 2D Navier-Stokes Equations on Unstructured Adaptive Grids," *Proceedings of the AIAA 9th Computational Fluid Dynamics Conference*, AIAA Paper 89-1932, June 1989.

[9] H. Kodama and S. Nagano, "Potential Pressure Field by Stator/Downstream Strut Interaction," *Journal of Turbomachinery*, Vol 111, pp. 197-203, April 1989.

[10] P. R. Spalart, "Simulation of Rotating Stall by the Vortex Method," *Journal of Propulsion*, Vol. 1, No. 3, pp. 235-241, May-June 1985.

[11] F. Davoudzadeh, N.-S. Liu, S. J. Shamroth and S. J. Thoren, "A Navier-Stokes Study of Rotating Stall in Compressor Cascades," *AIAA/ASME/SAE/ASEE* 24th *Joint Propulsion Conference*, AIAA Paper 88-3265, July 1988.

[12] F. E. Marble, "Propagation of Stall in a Compressor Blade Row," *Journal of the Aeronautical Sciences*, Vol. 22, 1955.

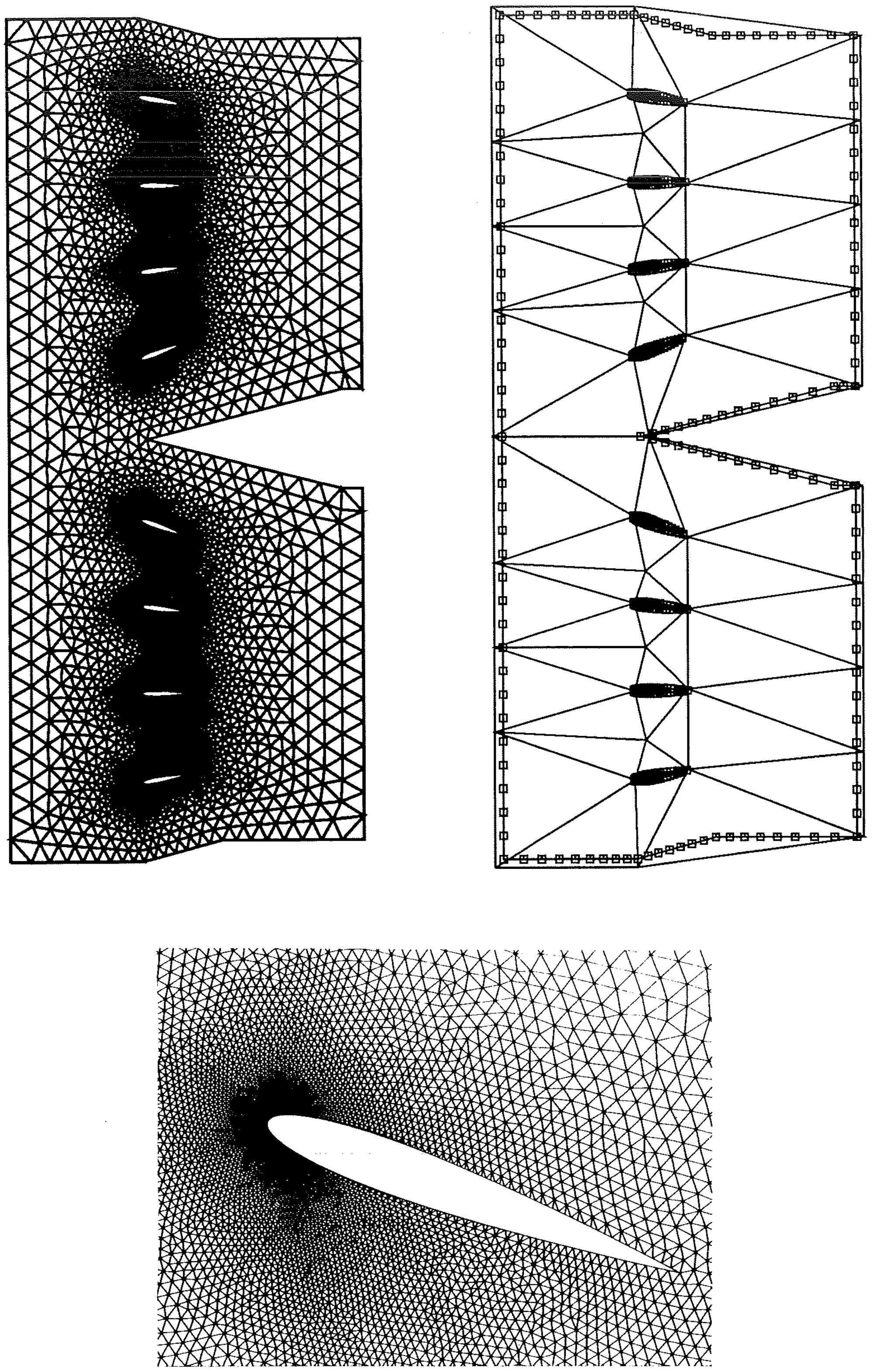
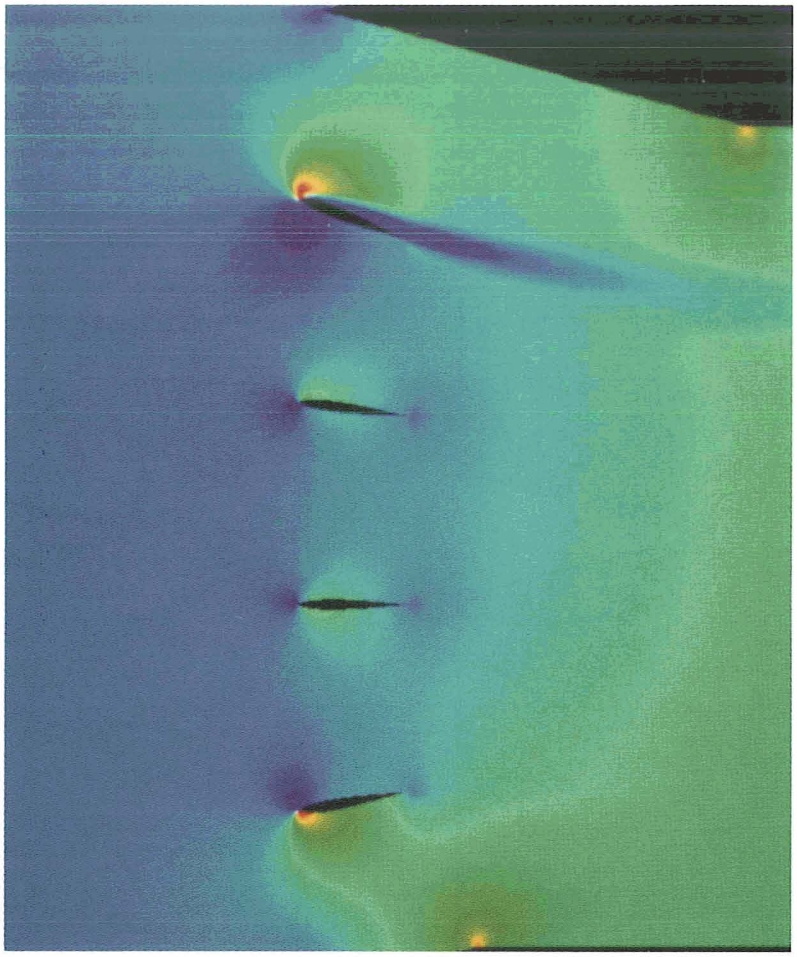


Figure 6: Pylon/strut - Background mesh and the complete mesh with three times the actual mesh cell size and a blowup around one of the blades from the actual mesh.



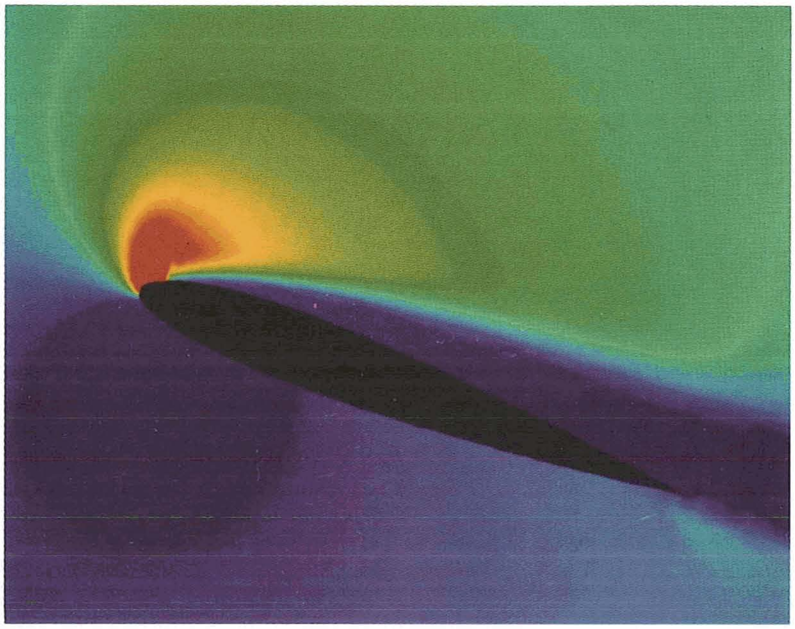
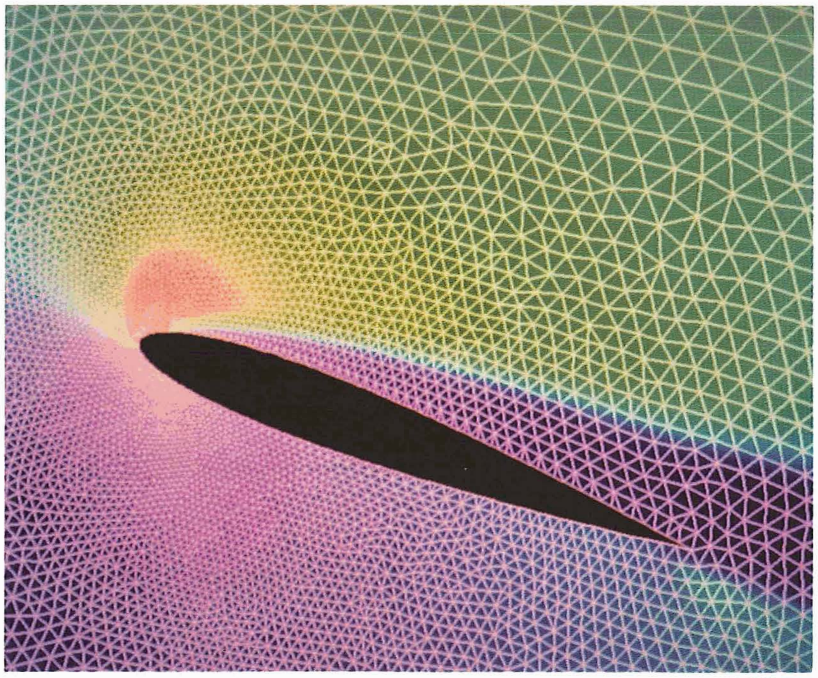


Figure 7: Pylon/strut - Mach number contours with a blowup around the strut nearest the large pylon.

ORIGINAL PAGE IS
OF POOR QUALITY



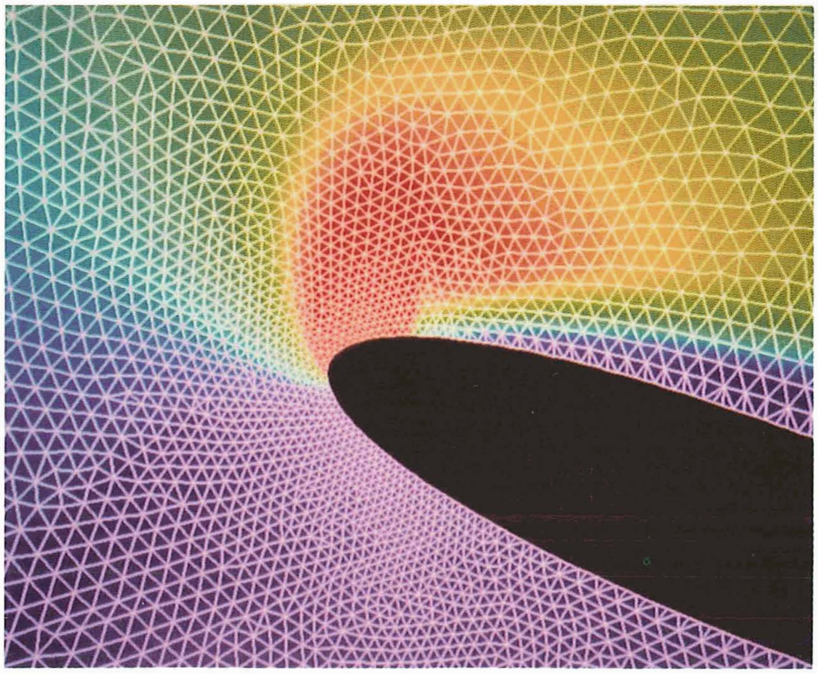


Figure 8: Pylon/strut - Mach number contours and computational grid of blowups around the strut nearest the large pylon.

ORIGINAL PAGE IS
OF POOR QUALITY

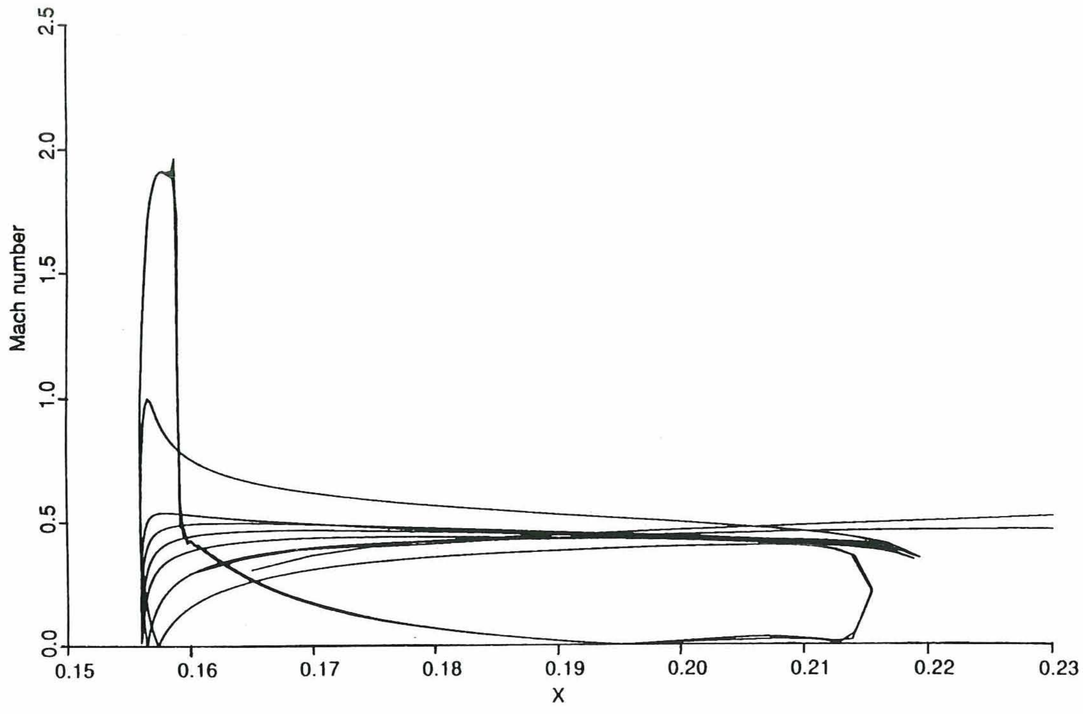


Figure 9: Pylon/strut - Mach number distribution on the pylons and struts.

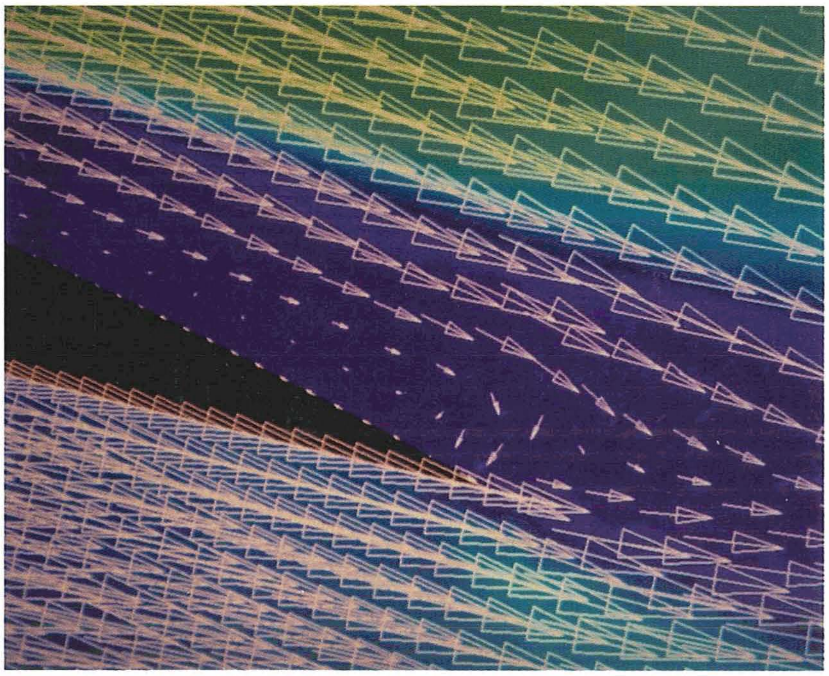


Figure 10: Pylon/strut - Mach number contours and flow vectors showing separation around the trailing edge of the strut nearest the large pylon.

ORIGINAL PAGE IS
OF POOR QUALITY

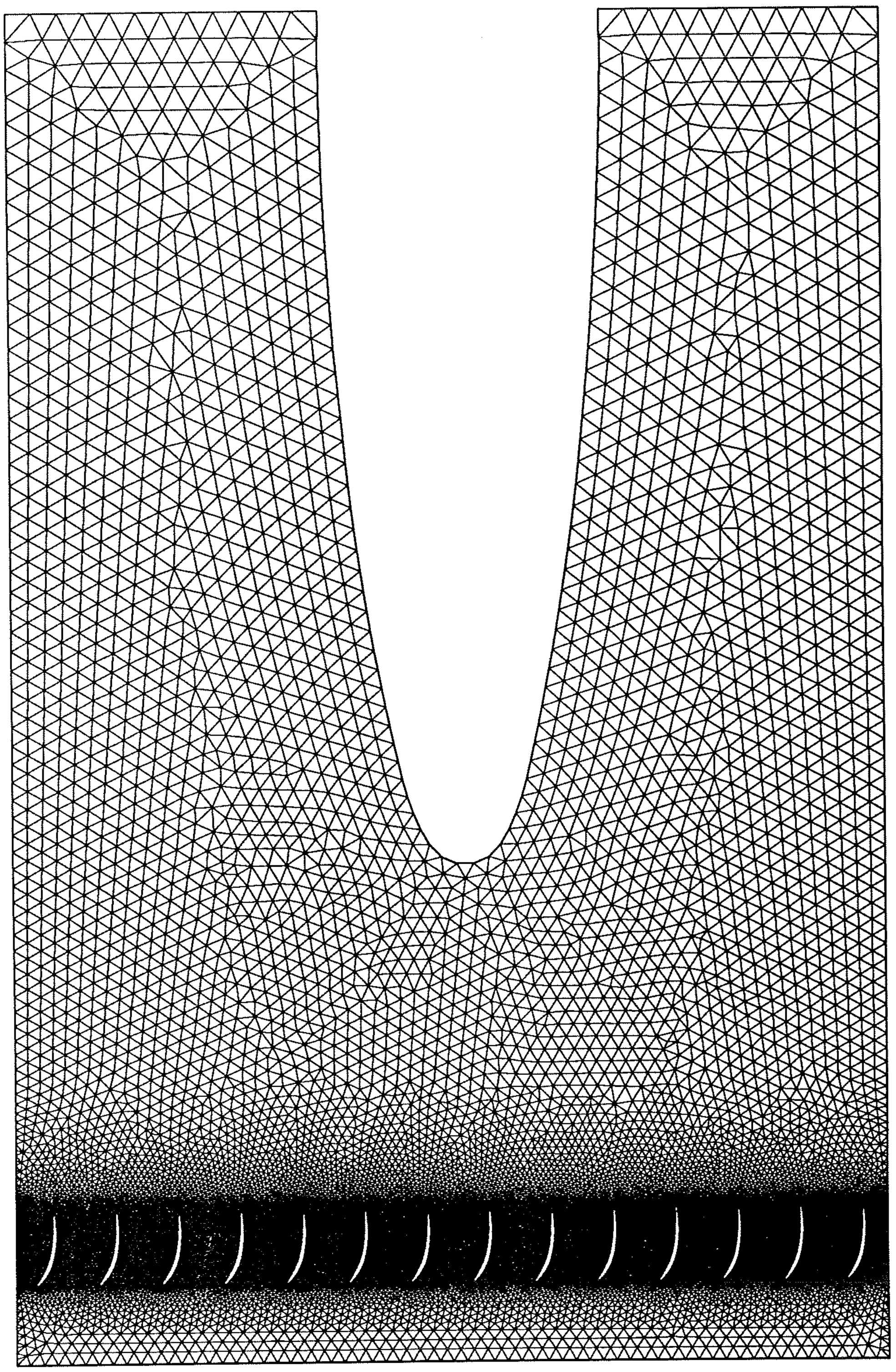


Figure 11: Pylon/OGV - Computational grid of complete domain.

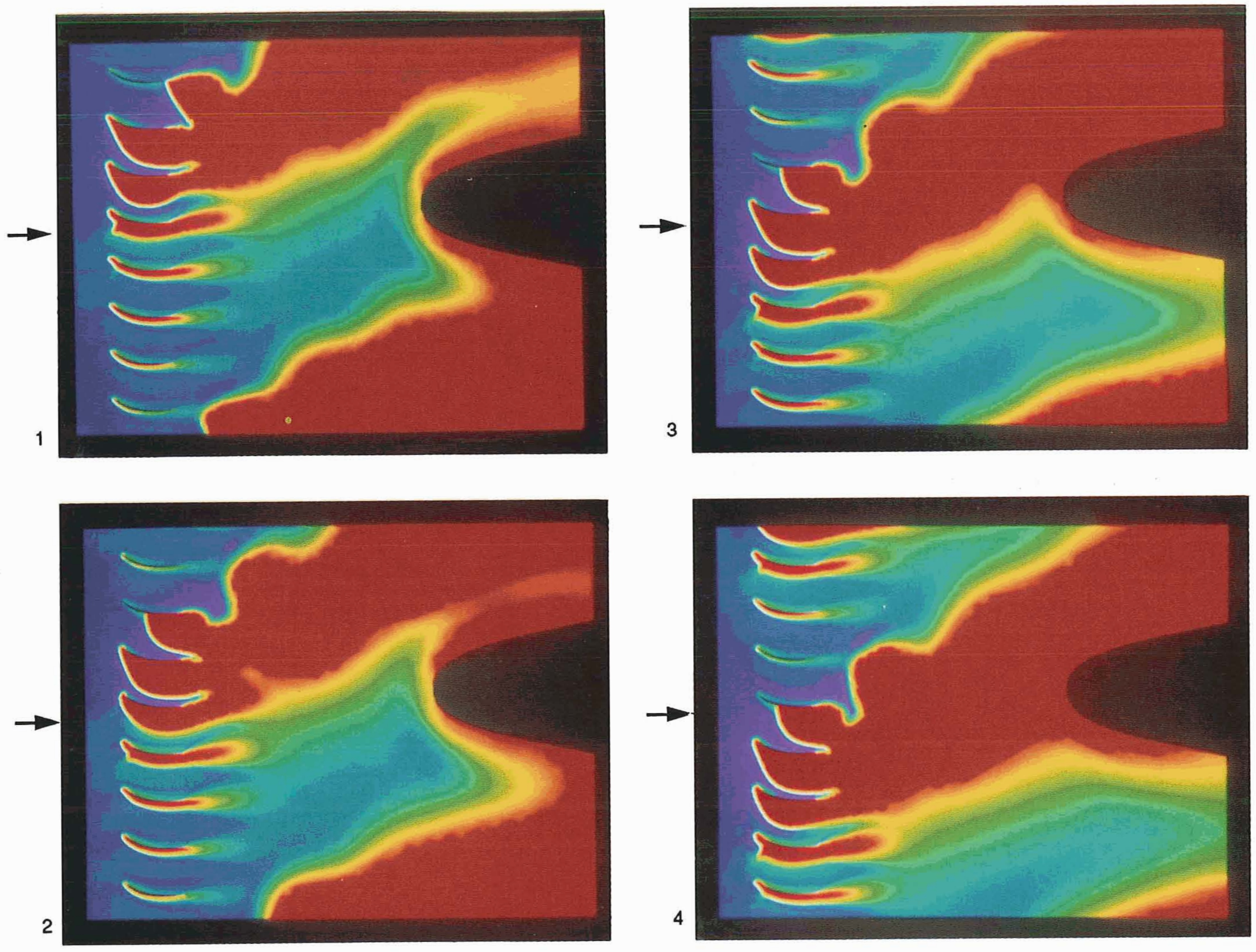


Figure 12: Pylon/OGV - Entropy contours at four different stages in a cycle.

ORIGINAL PAGE IS
OF POOR QUALITY

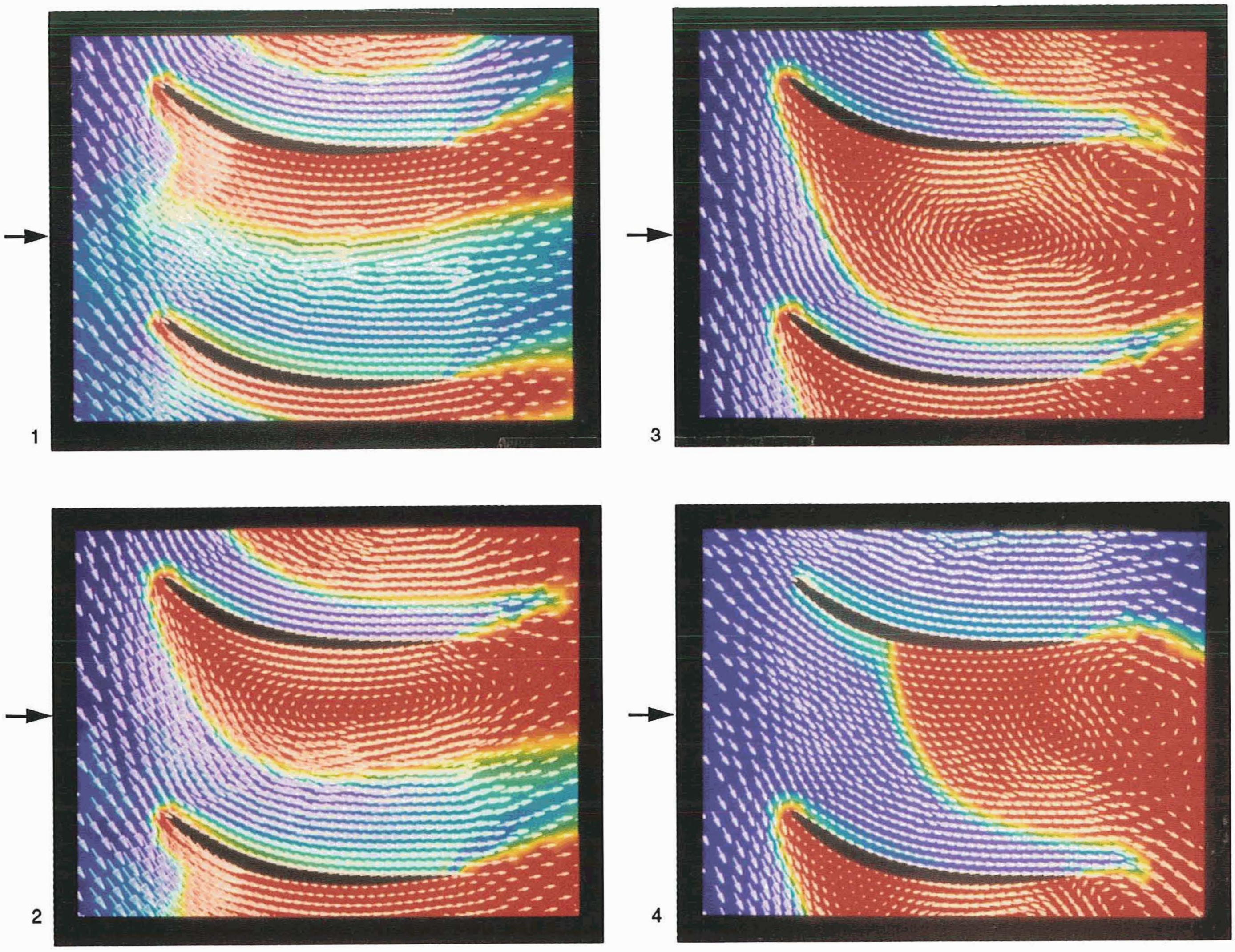


Figure 13: Pylon/OGV - Blowup of entropy contours with flow vectors from Figure 12.

ORIGINAL PAGE IS
OF POOR QUALITY