Automated Fluid Feature Extraction from Transient Simulations

Final Progress Report

Robert Haimes
haimes@orville.mit.edu

December 20, 2000

NASA Ames Research Center
Agreement: NCC2-985

Department of Aeronautics and Astronautics
Massachusetts Institute of Technology
77 Massachusetts Avenue
Cambridge, MA 02139

1 Introduction

In the past, feature extraction and identification were interesting concepts, but not required in understanding the physics of a steady flow field. This is because the results of the more traditional tools like; iso-surfaces, cuts and streamlines, were more interactive and easily abstracted so they could be represented to the investigator. These tools worked and properly conveyed the collected information at the expense of a great deal of interaction. For unsteady flow-fields, the investigator does not have the luxury of spending time scanning only one "snap-shot" of the simulation. Automated assistance is required in pointing out areas of potential interest contained within the flow. This must not require a heavy compute burden (the visualization should not significantly slow down the solution procedure for co-processing environments like **pV3**). And methods must be developed to abstract the feature and display it in a manner that physically makes sense.

The following is a list of the important physical phenomena found in transient (and steady state) fluid flow:

1.1 Shocks

The display of shocks is simple; a shock is a surface in 3-space. As the solution progresses, in an unsteady simulation, the investigator can view the changing shape of the shock surfaces. Some previous work has been done at MIT (as well as other places) on this problem. This early work [Darmofal91] developed the following algorithm:

First determine the normal direction to the shock. Across a shock, the tangential velocity component does not change; thus, the gradient of the speed at a shock is normal to the shock. The exact location of the shock is then determined by calculating the magnitude of the Mach vector, in the direction of the speed gradient, at all points in the domain. The normal Mach number is defined as the Mach vector dotted into the speed gradient. Thus, a positive normal Mach number indicates streamwise compression and a negative normal Mach number indicates expansion. If this value is 1.0 then a shock has been found (or possibly an isentropic recompression through Mach one). This entire iso-surface can be displayed to show the shock, but must be thresholded to remove the surfaces associated with the recompression and some stray portions of the flow field where the normal Mach number happen to be 1.0. The magnitude of the speed gradient was found to be an effective threshold.

1.2 Vortex Cores

Finding these features is important for flow regimes that are vortex dominated (most of which are unsteady) such as flow over delta wings and flow through turbine cascades. Tracking the core can give insight into controlling unsteady lift and fluctuating loadings due to core/surface interactions.

There has been much work done in the location of these features by many investigators. Again, there has been some success at MIT. This particular algorithm has been designed so that no serial operations are required, it is parallel, deterministic (with no 'knobs'), and the output is minimal. The method operates on a cell at a time in the domain and disjoint lines are created where the core of swirling flow is found. Only these line segments need to be displayed, reducing the entire vector field to a tiny amount of data.

This technique, although satisfying, is not without problems. These are:

1. Not producing contiguous lines.
   The method, by its nature, does not produce a contiguous line for the vortex core. This is due to two reasons; (1) for element types that are not tetrahedra the interpolant that describes point location within the cell is not linear. This means that if the core passes through these elements the line can display curvature. By subdividing pyramids, prisms, hexahedra and higher-order elements into tetrahedra for this operation produces a piecewise linear approximation of that curve. And (2) there is no guarantee that the line segments will meet up at shared faces between tetrahedra. This is because the eigenvector associated with the real eigenvalue will not be exactly the same in both neighbors, so when this vector is subtracted form the vector values at the shared nodes each tetrahedra sees a differing velocity field for the face.

2. Locating flow features that are not vortices.
   This method finds patterns of swirling flow (of which a vortex core is the prime example). There are other situations where swirling flow is detected, specifically in the formation of boundary layers. Most implementations of this technique do no process cells that touch solid boundaries to avoid producing line segments in these regions. But this does not always solve the problem. In some cases (where the boundary layer is large in comparison to the mesh spacing) this boundary layer generation is still found.

3. Sensitive to other non-local vector features.
   Critical point theory gives one classification for the flow based on the local flow quantities. 3D points can display a limited number of flow topologies including swirling flow, expansion and compression (with either acceleration or deceleration). The flow outside this local view may be more complex and have aspects of all of these components. The local classification will depend on the strongest type. Also if there are two (strong) axes of swirl, the scheme will indicate a rotation that is a combination of these rotation vectors based on the relative strength of each. This has been reported by [Roth96] where the overall vortex core strength was not much greater that the global curvature of the flow. The result was that the reported core location was displaced from the actual vortex.

1.3 Regions of Recirculation

Recirculation is a difficult feature to locate, but a simple one to visualize. A surface exists that separates the flow (in steady-state) so that no streamlines seeded from one side of this surface penetrate the other side. Some work has been done in locating this feature by computing the stream function. Also it is possible to use vector field topology to find the extent of this region and then draw a series of streamlines connecting the critical points. These lines can be tessellated to create this separation surface.

These methods do not work for transient problems. Like a series of instantaneous streamlines can be misleading in unsteady flow regimes, using techniques based on streamlines will not represent the regions of older fluid. By using the concept of time, recirculation can be identified as regions of fluid that are old in comparison with the *core* flow. Therefore instead of looking directly at flow topology we can calculate Residence Time. This is the Eulerian view of unsteady particle tracing (a

Lagrangian operation). A simple partial differential equation can be solved on the same mesh along with the flow solver. (NOTE: This is possible when performing co-processing; the CFD solver and Residence Time calculation have similar time limit constraints.) An iso-surface can be generated through the result so that regions of old fluid can be separated from newer fluid elements.

Process engineers (in particular for injection molding) and those individuals concerned about environmental pollutants use the concept of Residence Time. This formulation is from the statistical standpoint. We can not find a rigorous definition in the fluid dynamics literature.

1.4 Boundary layers

Boundary layers are features that are very important in most complex fluid flow regimes. The size and shape of the boundary layer are used to determine such values as lift and drag in external aerodynamics. For turbomachinery the size of the boundary layers determine the effective solidity. With regions of recirculation, the boundary layers determine the blockage. In all cases the boundary layer edge can be constructed as a surface (some distance away from solid walls) in 3D flows.

There have been no successes in any known work to robustly determine the surface that represents the extent of the boundary layer from traditional CFD solutions. Fundamentally, this is a very difficult problem. The edge is poorly defined numerically and is more a subtle transition that an abrupt feature.

Accurately knowing the edge of the boundary layer has many numerical benefits for the solver. Turbulence models can be more accurately applied. Grid adaptation can place nodes where they are needed. Split solvers (Euler in core flow, Navier-Stokes in boundary layers) will be more stable and accurate when the position of the edge of the boundary layer is known.

1.5 Wakes

The merging of boundary layers down stream from a body usually generates wakes. Like boundary layers, these features are important for both internal and external flows. Knowing where, and under what circumstances, the wakes impinge on other bodies can have a changing effect on the structural and thermal loads experienced on those surfaces. Again, there has been no real success in finding this feature.

2 Progress Made under this Contract

The goal of this work is to develop a comprehensive software feature extraction tool-kit that can be used either directly with CFD-like solvers or with the results of these types of simulations (i.e. data files). The output of the feature "extractors" are produced in such a manner that it could be rendered within most visualization systems. Much effort will be placed in further quantifying these features so that the results can be applied to grid generation (for refinement based on the features), databases, knowledge based and design systems. This requires two distinct phases; (1) the research into algorithms that will accurately and reliably find these features and (2) the design and construction of the software tool-kit.

2.1 Algorithms and Software

A web-site has been constructed that gives individuals access to both the written papers and available software that has resulted from this grant. This site also contains a small picture and movie gallery of the feature extraction imagery generated by these algorithms. Please look at the URL **http://raphael.mit.edu/fx**.

2.1.1 Shocks

The procedure explained above has been re-examined. First, much effort was placed in examining algorithms that find discontinuities in scalar fields. These techniques can be thought of as the 3D analogue to the methods used in image processing. This approach failed in finding shocks for the following reasons:

- Sharpness.
  Most CFD solvers that perform differences to compute derivative and flux quantities do not suppress saw-tooth oscillations in the solution. These can become unstable even in quiescent flow (for numerical reasons) and will blow-up in the presence of discontinuities. For this reason these CFD solvers "smooth" the flow field. This obviously reduces the ability to find sharp discontinuities since they have been removed. Even for solvers that can handle abrupt changes in the flow field, a shock will probably be smeared across 2 to 3 cells.

- Derivative quantities.
  There tends to be noise generated when derivative quantities are computed from local (cell based) operators. Using operators with larger stencils are possible in structured block meshes but difficult in unstructured grids. This noise problem is amplified when second derivatives are required.

Therefore the shock finder that requires looking for the inflection point – where the laplacian of the laplacian of pressure (the second derivative) is zero is doomed in CFD solutions.

A shock finder has been developed that is a modification of the early work described above. For steady state solutions, the normalized pressure gradient is used instead of the speed gradient – this is less susceptible to other flow features such as boundary layers. It has been found that no

thresholding is required. There is also an extension for transient solutions. See "Shock Detection from Computational Fluid Dynamics Results", AIAA paper 99-3285.

This paper discusses the shock location theory in detail and presents methods for shock classification. Details are presented for computing the shock strength and type (normal, oblique, bow and etc.). This paper also has an analysis of how to determine the shock speed for transient cases.

It was found that the filtering of false positives as discussed in this paper needed to be modified to work well for all flow regimes. The filtering is necessary for the transient correction (due to the derivatives required) and for both steady state and unsteady supersonic flows. The supersonic problem exists because the steady state marker has the normalized pressure gradient as a term. In areas of the flow field where there is a zero (or close to zero) pressure gradient the direction of this gradient can oscillate. This produces the false positives because the Mach number is greater than one. A simple filtering algorithm was implemented to remove these spurious hits:

1. Coalesce the regions.
   Based on whether the cell has any nodes above one for the test function mark the cell to be included in a region. By looking at the neighbors of this element, collect touching cells that are also marked. This can be thought of as the 3D analogue of the flood operation used in many *paint* programs.

2. Filter based on number of cells.
   Remove regions (groups of cells) that do not contain a specified minimum number of elements.

3. Filter based on pressure difference.
   Remove regions that have a pressure difference below a minimum *delta P*. A shock displays a significant pressure jump.

2.1.2 Vortex Cores

The original algorithm produces a series of disjoint line segments. When displayed, the eye puts together (or closes) the line, for a single core, (when the strength of the core is large). This is not acceptable for off-line uses (the first problem listed above) in that it is not possible to trace the full extent of the core. This issue is now resolved. Enforcing the cell piercing to match at cell faces insures that the line segments generated will produce a contiguous core. This was first attempted via the following modification to the algorithm:

1. Compute the $\underline{V}$ (the velocity gradient tensor) at each node.
   This requires much more storage – 9 words are needed for each node in the flow field. This has the advantage that the stencil used for the operation is larger than the cell and therefore the result will be generally smoother.

2. Average the node tensors (on the face) to produce a face-based $\underline{V}$.
   This insures that the same tensor is produced for the two cells touching the face.

3. Perform the eigen-mode analysis on the face tensor.
   If the system signifies swirling flow, determine if the swirling axis cuts through the face by the scheme used in the current method. If, so mark the location on the face.

This scheme worked at the expense of memory and a much higher CPU load. Four eigen-mode calculations are required for each tetrahedron instead of just one. In general, this can be reduced to two per tetrahedron, by the additional storage of face results (about 3 words per face). Note: there are about 2 times the number of faces as cells in a tetrahedral mesh.

This was not a good result, in particular for structured blocks, where each individual hexahedron is broken up into 6 tetrahedra (5, the minimum does not promote face matching). This means that for each element in the mesh a minimum of 12 eigen-mode analyses are required.

These performance problems suggested another, related, technique:

1. Compute the $\underline{V}$ at each node.
2. Perform the eigen-mode analysis on the node tensor.
   The tensor can be overwritten with the critical point classification and the swirl axis vector for rotating flow.
3. Average the swirl axis vectors for the nodes that support the tetrahedral face.
   This should only be done if all nodes on the face indicate swirling flow. Some care needs to be taken to insure that the sense of the axis vectors are the same. Determine if the swirling axis cuts through the face, and if so, mark the location on the face.

For tetrahedral meshes, the reduction of compute load is by a factor of 5 to 6 over the original method (there are roughly 5.5 tetrahedra per node in 'good' unstructured grids). For structured blocks, where the number of nodes is about equal to the number of hexahedra, the number of eigen-mode analyses required is on the order of one per cell.

For coherent collected cores to be produced, all the disjoint lines are used to build threaded lists. The end points now match at tetrahedron faces. Unfortunately, do to the fact that some tetrahedra do not have 2 pierce points, special processing is required:

- 1 intersection
  The point is used as a seed point for streamline integration. This integration only persists until a face is hit (in the original cell – not the decomposed tetrahedron). If there is an intersection on that face the connection is made. If the element was a tetrahedron to begin with, or no match can be found then it is assumed that the core as either started or ended.
- 3 & 4 intersection points
  These connections are deferred. At the end, threads are put together to produce the longest (most number of points) cores. This is a recursive procedure.

All resultant core segments that have less than 4 disjoint segments are culled.

It should be noted that these situations occur because CFD results are, by their very nature, not smooth. Saw-tooth oscillations in the vector field can produce *noisy* results locally.

This new algorithm is still linear and will produce incorrect results when the flow is under 2 or more (relatively equal) rotating influences (the third problem listed above). A paper [Roth98] was presented that, on first viewing, appears to solve this problem. The authors suggest that by examining the higher-order derivatives of the velocity field one can capture curvature. They first recast the technique described above then apply the higher-order correction:

- Parallel alignment
  An intersection point on a face is where the *reduced velocity* is zero. Therefore the velocity vector is parallel to the real eigenvector of $\underline{V}$ (the velocity gradient tensor). $\vec{u}$ (the velocity vector) is an eigenvector of $\underline{V}$ and therefore a solution of $\underline{V}\vec{u} = \lambda\vec{u}$. This suggests that looking for parallel alignment is the same as the current vortex algorithm.

- Second Order
  Computing the velocity second derivatives ($\mathbf{T}$) produces a $3x3x3$ tensor. Checking for alignment of the second derivative following a particle produces:
  $\underline{VV}\vec{u} + \mathbf{T}\vec{u}\vec{u} = \kappa\vec{u}$

In practice, this does not seem to work well. The velocity field generated by most CFD solvers is not smooth. As a result this technique produces both many false positives and also misses intersections due to the local saw-tooth variations. The other problem is the storage requirements. For large data sets, requiring 27 words/node of the rank 3 tensor and 9 words/node for the rank 2 tensor becomes prohibitive.

The issue of alignment coupled with the linear limitations of the classification and results of the eigen-analysis hint at an alternative technique. This approach uses $eig(\underline{\Omega})$ instead of $eig(\underline{V})$. The result is invariably one zero and a complex conjugate pair (λ_c) of eigenvalues – always the indication of swirling flow. The eigenvector associated with the zero eigenvalue can be used with the face-piercing algorithm to produce core segments. This technique has the effect of removing the bulk and shear components of stress from the analysis.

The effect of this algorithm is, at times, longer and more contiguous cores. For example, tracking a horseshoe vortex in a turbine cascade, the core line would disappear as the flow accelerated around the crown of the suction surface. This was due to a change in classification from swirl to non-swirling flow. With the irrotational components removed the core can continue to be extracted. The influence of the irrotational terms also explains why the core segments would break-up when the core strength would diminish. Under these conditions the stress components end up corrupting the direction of the eigenvector used for the swirl axis.

It should be noted that the eigenvector associated with the zero eigenvalue of $eig(\underline{\Omega})$ is just $\vec{\Omega}$. This means that the storage requirements are no longer 9 words per node but 3 and no eigen-analyses are required. The result is that vortex cores can simply be found where $\vec{\Omega}$ aligns with $\vec{u}$.

2.1.3 Regions of Recirculation

The recirculation algorithm, Residence Time, described above requires close integration with the flow solver. The choice is either that solver writer completely incorporates this equation by adding one more equation to the state-vector or some co-processing system (like the visualization suite **pV3**) is used.

Obviously, the best place for this PDE is within the solver. For the second choice, an API for solving this PDE is being developed so that there is access to all of the required data. A Lax-Wendroff scheme has been implemented for time integration. Therefore if some implicit or high-order explicit time integration scheme is used for the solver care must be taken in selecting the time-step so that

the solving of the Residence Time equation is stable. There is a call in the API return the maximum stable time-step.

AIAA paper 99-3291, "Using Residence Time for the Extraction of Recirculation Region", describes, in detail, both the theory and implementation of the Residence Time concept.

2.1.4 Boundary layers and Wakes – Phase 1

An algorithm has been constructed that allows the use of iso-surfacing to separate the boundary layers and wakes from core flow. The method stems from the fact that these features display both rotating flow and fluid under shear stress. This is why, sometimes the vortex core technique gives a false-positive for locations in boundary layers. Therefore, with a boundary layer finder we should be able to mask out these finds in the boundary layer and only display those lines that trace back from the outer flow.

To numerically define these quantities we again start with the $\underline{V}$ (the velocity gradient tensor) at each node:

- Rate of Rotation.
  This quantity is related to vorticity. A skew-symmetric tensor is produced by subtracting the transpose of $\underline{V}$ from $\underline{V}$. The result has zero on all of the diagonal terms and the off-diagonal terms are symmetric but have opposite signs across the diagonal. These values are coordinate system invariant. For this application, the norm of the upper (or lower) terms is used for the rotation scalar. This is a measure of the rate of solid-body rotation.

- Rate of Shear Stress.
  A symmetric tensor can be produced from $\underline{V}$ by adding it to its transpose. This defines the Rate of Deformation tensor. The matrix represents both the bulk and shear stresses and is dependent on the coordinate system. To extract a single scalar that is coordinate system invariant and has the bulk terms removed it is necessary to diagonalize this tensor. The result produces a vector, which signifies the 'principle axis of deformation'. By employing techniques from Solid Mechanics, the norm of the second principal invariant of the 'stress deviator' can be used as a measure of the shear and employed as the scalar.

This scheme initially looked promising but these problems have not been resolved:

- A node based scalar function of shear and rotation has not been constructed that can based on theory.

- The value of this function will probably be dimensional, having units of inverse time (the same as shear and rotation).
  This means that the iso-surface value used to define the edge of the layer changes from case to case. This scalar needs to be multiplied by some characteristic time associated with the problem.

- The value used for the iso-surface also needs to be coupled to theory.

2.1.5 Boundary layers and Wakes – Phase 2

The lack of progress using this avenue has prompted examining another approach. For this attempt we start from Boundary Layer Theory.

The goal is to find a general method to calculate the displacement and momentum thickness of boundary layers near solid surfaces in the model. These quantities are less subjective measures of the boundary layer thickness. The displacement thickness is related to the blockage effect caused by the viscosity near the body, and the momentum thickness is related to the drag on the body; both of which are important to designers. It may also be possible to calculate the blockage effect of viscosity at a section in the flow passage of turbomachinery with these methods.

- Theory

  The method that has been investigated is to integrate the vorticity from the solid surface out to the free stream in a direction normal to the surface. The vorticity is being used as an approximation to the change in velocity normal to the wall, and is useful because it goes to zero in the free stream, allowing for a simple marking for the termination of integration. Using the vorticity eliminates the need to determine the free stream velocity, U, and the boundary layer thickness at each point on the wall. These variables are normally used to define the displacement and momentum thickness, as shown in the following equations 1 and 2, where δ is the boundary layer thickness, δ_1 is the displacement thickness and δ_2 is the momentum thickness.

$$\delta_1 = \int_0^{\delta} (1 - \frac{u}{U}) dn \tag{1}$$

$$\delta_2 = \int_0^{\delta} \frac{u}{U}(1 - \frac{u}{U}) dn \tag{2}$$

  The approximation to the displacement thickness is shown in equation 3 where ω is the vorticity, and the integration proceeds from the wall surface to where the vorticity is approximately zero.

$$\delta_1 \approx \int_0^{\omega=0} dy - \frac{\int_0^{\omega=0} u dy}{\int_0^{\omega=0} \omega dy} \tag{3}$$

  The procedure that is being followed is outlined below:

  1. Determine curves that are normal to the surface and do not intersect one another other. This is being done by actually solving Laplace's equation on the domain of the model, and using the gradient of this solution to define normal vectors.
  2. Calculate the vorticity from the velocity components given from the CFD solution.
  3. Integrate the vorticity from the surface along the normal curves until the vorticity drops below a certain threshold. This integration yields the free stream velocity at that surface point.
  4. Use the free stream velocity value to calculate the displacement and momentum thickness at that point.

- Result

  The first test was to use analytic methods to determine if this method duplicated the results

of Blasius for a flat plate in uniform flow at zero angle of attack. Calculating the free stream value by integrating the vorticity yields a free stream value within 1% of the actual value.

The method was also tested by imposing a Blasius solution on a rectangular mesh, then using a numerical implementation to find the displacement thickness along the plate. The numerical results closely followed the Blasius solution of the boundary layer thickness quantities.

After successfully working through this algorithm in 2D it was found that:

- Numerical integration was slow
  A great deal of CPU time is taken to integrate the vorticity upstream. This is due to the numerical stability and therefore the small time-steps required. The scheme outlined above could be considered for steady state post-processing, but for co-processing transient solutions, each time-step must be reconsidered and recomputed from scratch.
- Non-boundary layer generated vorticity
  This scheme treats all vorticity the same. If a free vortex exists (for example, the vortices above a delta wing flying at angle of attack) this is included in the calculation. This vorticity will clearly corrupt the resultant boundary layer thickness.
- Wakes
  There is no body to integrate from!

2.1.6 Boundary layers and Wakes – Phase 3

Perhaps the most obvious approach to finding viscous regions is to run an inviscid Euler simulation in parallel with a viscous Navier-Stokes simulation and track the regions where the computed flow differs. Unfortunately this is not practical. Boundary layers of significant thickness, including regions of separated flow, displace the regions of inviscid flow away from the walls and destroy similarity between viscous and inviscid solutions. This displacement is one of the properties we originally wished to find.

Boundary layers are generated by viscous phenomena. In this attempt we take the view that a fundamental indicator of the presence of viscous effects is the group of viscous terms in the Navier-Stokes equations. We can integrate these terms over each discretized cell volume in the simulation to determine the local level of viscous influence. Because these terms are the difference between the Navier-Stokes and Euler equations, this method remedies the approach mentioned above by embracing a localized perspective.

$$\rho\frac{D\vec{V}}{Dt} + \nabla p = \nabla \cdot \tau_{ij} \tag{4}$$

$$\rho\frac{Dh}{Dt} = \frac{Dp}{Dt} + \nabla \cdot (k\nabla T) + \tau_{ij}\frac{\partial u_i}{\partial x_j} \tag{5}$$

The last term of the momentum (4) and energy (5) equations above is the viscous contribution to each, with the viscous portion of the stress tensor τ_{ij} defined as

$$\tau_{ij} = \mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) + \delta_{ij}\lambda\nabla \cdot \vec{V} \tag{6}$$

where the $\nabla \cdot \vec{V}$ terms are generally negligible. Since the viscous terms constitute the entropy-generating portion of the Navier-Stokes equations, an even more fundamental way to view viscous influence in the flow is to look at the rate of entropy generation in the flow. In an isentropic scenario the conservative form of the entropy conservation equation is

$$\frac{\partial \rho s}{\partial t} + \nabla \cdot (\rho \vec{V} s) = 0 \tag{7}$$

where $s = \ln \frac{h^{\gamma/(\gamma-1)}}{p}$. In an entropy production-detecting scheme, however, the right side of the equation would be some value of interest. Both approaches are being compared against analytically known laminar boundary layer profiles before one method is selected for development and experimentation.

Clearly this task is not yet complete. In fact, it may have been bold to suggest that finding the edge of boundary layers and wakes was within the scope of the effort. The boundary layer is an important feature in turbulence modeling. Algebraic models require the edge of the boundary layer and, in general, do not work because of this requirement. More sophisticated models involve one (or more) additional PDEs to be solved in order to close the Reynolds Averaged Navier Stokes Equations. This is not too different than the approach being being currently investigated, leaving one to believe that this may produce a scheme that will work.

3 Presentations and Publications

3.1 In Peer-reviewed Journals

- D. Kenwright and R. Haimes, Automatic vortex core detection. *IEEE Computer Graphics and Applications*, Vol. 18, No. 4, IEEE Computer Society Press, pp. 70-74, July 1998.

3.2 In Conference Proceedings

- D. Lovely and R. Haimes, Shock detection from Computational Fluid Dynamics results. 1999 AIAA Computational Fluid Dynamics Conference, AIAA Paper No. 99-3285, Norfolk, VA, June 1999.
- R. Haimes and D. Kenwright, On the Velocity Gradient Tensor and Fluid Feature Extraction, 1999 AIAA Computational Fluid Dynamics Conference, AIAA Paper No. 99-3288, Norfolk, VA, June 1999.
- R. Haimes, Using residence time for the extraction of recirculation regions, 1999 AIAA Computational Fluid Dynamics Conference, AIAA Paper No. 99-3291, Norfolk, VA, June 1999.
- I. Trotts, D, Kenwright and R. Haimes, Critical Points at Infinity: a missing link in vector field topology. NSF/DoE Lake Tahoe Workshop on Hierarchical Approximation and Geometrical Methods for Scientific Visualization, Oct. 2000.

3.3 Extended Abstracts

- L. Basket, R. Haimes and D. Kenwright, Shear Layer Feature Extraction in Viscous CFD Simulations, submitted to 2001 AIAA Computational Fluid Dynamics Conference.

4 Personnel

David Lovely finished his Masters Degree. His thesis was entitled: *Boundary Layer and Shock Detection in CFD Solutions*, October, 1999. Lawrence Baskett is currently working on Phase 3 of the Boundary Layer detection for his Masters thesis.

5 Awards

- The keynote address given at the 7-th Annual Conference of the CFD Society of Canada was *Automated Feature Extraction from Transient CFD Simulations* given by Robert Haimes.
- At IEEE Visualization 1999 the panel discussion entitled *Automation vs. Interaction: What's Best for Big Data?* was awarded Best Panel at the conference. As a member of the panel, Robert Haimes' position was for automated feature extraction, backed up by the work on going here.

6 Technology Transfer

The software generated under this contract (FX) has been incorporated into these commercial visualization software packages:

1. Intelligent Light (FieldView)
   Intelligent Light was showing feature extraction running under FieldView at the AIAA Aerospace Conference in Reno NV (in January 2000). An image in the Vortex Core gallery on the FX web-site has come from their product. IL has FX included as an integral part of FieldView Rev 7.
2. CEI (Ensight)
   Ensight is used a great deal by ARL and other Army installations. It is also the DOE visualizer of choice. CEI has included portions of FX into their current release.

The following software companies are reviewing FX for inclusion into their software:

1. ICEM-CFD using ICEM-Visual3.
2. Kitware authors of VTK.
3. AVS
4. Amtec Engineering for TECPLOT.