39th AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit
Huntsville, AL
20-23 July 2003

AIAA-2003-5051

CFD Process Pre- and Post-processing Automation in Support of Space Propulsion

Suzanne M. Dorney*
NASA Marshall Space Flight Center
Applied Fluid Dynamics Analysis Group
MSFC, AL 35812

ABSTRACT

The use of Computational Fluid Dynamics or CFD has become standard practice in the design and analysis of the major components used for space propulsion. In an attempt to standardize and improve the CFD process a series of automated tools have been developed. Through the use of these automated tools the application of CFD to the design cycle has been improved and streamlined. This paper presents a series of applications in which deficiencies were identified in the CFD process and corrected through the development of automated tools.

INTRODUCTION

Computational Fluid Dynamics (CFD) analyses are currently used in the design and analysis of all the fluid handling devices in rocket engines. Historically, however, there have been only limited efforts to standardize the application of these analyses to the wide range of geometries and flow conditions found in rocket engines. Recently a program was initiated at NASA Marshall Space Flight Center (MSFC) to develop and apply tools that would standardize the major components of the CFD process: code verification, pre-processing, CFD analysis, and post-processing.

This paper describes a set of needs that were identified in the CFD process, the development of tools to address those needs and an explanation of the results of using each tool. In an attempt to error check, standardize and streamline the CDF process as it applies to rocket engines, a series of C++ codes have been developed. These codes are codes that are used at
MSFC: CORSAIR [1-3] and FDNS [4]. CORSAIR is a code used primarily to simulate flow through turbo machinery components and FDNS is used to simulate combustion devices. The goals of each tool are to reduce errors, speed up the application process, and achieve greater insights into the physics being modeled. Detailed descriptions of each tool and how they are being used to enhance various aspects of the CFD process are presented in the following sections.

CODE VERIFICATION

When a new version of a flow code is developed it is vital that all previous capabilities of the flow code are preserved. The process of testing a new version is referred to as code verification. To address the need for FDNS code verification *TestFDNS* was developed. *TestFDNS* automates the validation process by running the new version of the code on a series of test cases from a pre-defined test suite. Once the benchmark test suite has been established a user may select which cases to run in order to validate the latest version of the code. Upon completion of each case the results of the new case are compared to the results from the benchmark test suite. A report referred to as a *DeltaReport* is generated that indicates the level of agreement between the test results and the benchmark results. Since many of these cases require several hours, if not days, to complete an email summary message is also sent to the user when each case completes.

* Computer Scientist

Results

TestFDNS has been used extensively to validate new and modified versions of FDNS. The time savings in being able to run and analyze the results of several test cases automatically is quite significant. Cases that previously would take two days (manually) can now be computed in 20 minutes. *TestFDNS* is also having an impact in the development of new versions of FDNS. When an improvement is incorporated into the flow code *TestFDNS* is being used to test the intermediate results. The *DeltaReport* is then used to determine if the latest set of modifications results in a solution that is closer, or farther from, the known results in the test suite.

The use of *TestFDNS* has become standard practice for all new releases and modifications of the code. An example of a partial *DeltaReport* is shown in Fig 1.

PRE-PROCESSING

As part of an Advanced Space Transportation Program simulations are being performed for a Rocket Based Combined Cycle engine (RBCC). This geometry is represented as a series of grids. Multiple grids are used for one of two reasons. Grid generation of the complete system is often much easier when it can be done in segments. Multiple grids can also be used to take advantage of multiple processors. The solution of each grid (or combination of grids) can be accomplished on a separate processor. Of particular interest in a simulation are the boundary conditions. Boundary conditions control the physical behavior of the flow model. The edges of each grid are processed differently than internal points. In a single grid simulation the boundary conditions are relatively simple to specify, yet in the RBCC simulation where 22 grids are modeled the situation is significantly more complicated. Over 2000 separate values are needed to specify the boundary conditions. These 2000 values need to be verified before the simulation is computed and the results are evaluated. This verification was previously done manually. Because of the sheer volume of values this process was error prone and ambiguous.

PreViewer is an interactive visualization tool specifically designed to address the issues of boundary conditions. It can be used to visually specify boundary conditions, examine boundary conditions, as well as error check boundary conditions. An example panel from *PreViewer* used to visually inspect the interface between two grids is shown in Fig. 2.

Results

The impact of *PreViewer* was immediate. PreViewer was able to detect several errors in cases that were currently in the final stages of running, and in one case the final stage of post-processing. This particular simulation consisted of 44 separate grids and had taken several days to error check all of the boundary specifications. Through the use of *PreViewer* within seconds the error report indicated one error still existed in the specification of the boundary conditions. *PreViewer* is now run as standard practice prior to starting any FDNS simulation. Not only does it detect errors but it ensures the validity of the boundary condition specification. Shortly after its release a batch version of the error reporting capability of *PreViewer* was added so that the error report could be generated without initiating an interactive session.

In addition to error checking the input files for an FDNS simulation, *PreViewer* has evolved into a full pre-processing tool. To take advantage of parallel CFD codes and multiple processors it is often advantageous to break domains into multiple grids. Codes exist that can automatically break up a grid system, except once again the boundary conditions would need to be specified by hand. Figure 3 shows a vortex core chamber that was originally discretized using 9 grids and was then broken up into 19 grids. This almost doubled the number of boundary conditions. Before *PreViewer*, most engineers would not undertake such a task because the time needed to specify and verify the boundary conditions was too great. With the decomposition option in *PreViewer* this process is now automated and the net result of breaking up the grids in this case was a speed up of a factor of 5.

To aid in post-processing *PreViewer* has the capability to combine grids after a simulation is completed.

INTERIM-PROCESSING

When a physical experiment is being performed engineers have access to several sensor panels used to monitor the experiment. In this manner they can determine if the experiment is exhibiting the expected behavior. In a similar manner it is often useful to monitor the interim results of a computational simulation during its processing. To assist in this area two tools, *Monitor* and *FlowShow* have been developed. These tools can be used to construct a

series of contour, vector, and line plots that are generated from the intermediate solutions being generated. *Monitor* is a package that will generate a collection of plots in an interactive window. Each image in the window is updated when the next solution file is written. Figure 4 shows such a collection of plots. *FlowShow* can be used to generate an animation of a single plot. Each frame of the animation is generated from an intermediate solution. This tool is particularly useful when the time between printing solutions is quite large. A single frame of an animation generated with *FlowShow* is shown in Fig. 5. Through the use of these tools it is possible for an engineer to track the progress of a solution while it is forming. Of particular interest is determining when the simulation can be terminated because a solution has converged or reached a time-periodic state.

Results

The use of *Monitor* met with some initial resistance as users felt that it would draw too much on the already overloaded computer systems. However this was not the case. *Monitor* was used to watch several cases and allowed for the early detection of problems in the simulation, as well as convergence. *Monitor* has suffered, in a sense, from its own success. *Monitor* was designed to allow the user to track a particular value while the solution is forming. This ability proved to be so useful that the use of "monitor" points within the flow solver itself has become highly utilized.

FlowShow was used in those cases where a final animation of a simulation was more useful than an updating image. In particular when a new restart file was only generated once or twice a day it was more beneficial to view a single animation at the end of a simulation than to constantly track the simulation. One case where this was particularly useful was a simulation of a vortex chamber in which a flow instability was occurring and through the use of *FlowShow* it was visualized.

POST-PROCESSING

Although there are several commercially available post-processing visualization packages available there is still the need for engineers to be able to generate animations quickly and easily. This was particularly difficult with time dependent simulations. To address the issues of generating animations quickly and easily *Animator* was developed. *Animator* is a batch-processing tool that can be used to generate animations of time dependent solutions. In addition to traditional contour and vector plots *Animator* also has automated feature extraction capabilities [5-6]. A single frame from an animation of a shock is shown in Fig 6.

Parametric studies are being performed on a regular basis as a way to determine optimal component geometries. Because of the volume of results generated the management of a parametric study can be overwhelming. To deal with this volume of data many engineers reduce the data to scalars such as efficiency for their final analysis of the different simulations. To address the need to manage and interpret the results of a parametric study the tool *AutoPlot* was developed. This tool allows for specific values to be determined from the solution such as the location of complete combustion or the maximum temperature along a wall. *AutoPlot* has the capability to be used as a calculator and determine quantities not present in the solution file itself. The specific values are then organized into a single report. *AutoPlot* can also be use to generate a set of standard contour and line plots for each case in the parametric study. The primary use of *AutoPlot* is to generate values and plots in a consistent manner for each case run as part of a parametric study.

Results

The results of using *Animator* have been fairly dramatic. Although the use of animations is not new, the ability to generate simple animations in batch mode without the need for extensive training on intricate animation packages has been quite useful. In addition to its obvious uses in presentations the ability to generate animations in batch mode from partial solutions enables flow features to be evaluated and errors to be detected before large amounts of time and resources are invested.

AutoPlot has been used extensively in the preliminary phases of an injector parametric study. In the initial phase 50 simulations were performed and analyzed. By using *AutoPlot* it was possible to organize, monitor, and compare the results of each case. Since *AutoPlot*'s ability to calculate values is not limited to a pre-defined set of variables it can be used in many different ways. In particular, *AutoPlot* was used in a pre-processing phase to check the geometries of each case prior to starting the simulation. One of the variable parameters was the size of the injector opening. *AutoPlot* was used to examine each geometry file and calculate the opening in each simulation. These calculations were compared, within *AutoPlot*, to a set of expected values to determine if each grid was correct. Thus examining a simple

report was all that was needed to confirm that each simulation was set up correctly.

CONCLUSIONS

The effort to standardize the CFD process as it applies to rocket engines has been successful. Through the use of the tools *TestFDNS*, *PreViewer* and *AutoPlot* there is a standard process and system of tools that are being used consistently. The ability to reduce errors and establish confidence in the validity of each simulation prior to starting the computation has yielded a significant improvement in the CFD process. *Animator* has also had a significant impact by allowing engineers to quickly and easily examine the results of time dependent simulations through the use of animations. Other tools such as *Monitor* and *FlowShow* are used less frequently, but allow the CFD user a suite of tools that can be used when special cases arise.

ACKNOWLEDGEMENTS

The author would like to acknowledge the contributions of members of the Applied Fluid Dynamics Analysis Group at NASA Marshall Space Flight Center especially Dr. Daniel Dorney, Dr. Jeff West, and Mr. Fransisco Canabal.

REFERENCES

1. Dorney, D. J., Griffin, L. W., Huber, F., Sondak, D. L., ``Unsteady Flow in a Supersonic Turbine Stage With Variable Specific Heats,'' *AIAA Journal of Propulsion and Power*, Vol. 18, No. 2, March-April, 2002, pp. 493-496.

2. Dorney, D. J., Griffin, L. W., and Huber, F., ``A Study of the Effects of Tip Clearance in a Supersonic Turbine,'' *ASME Journal of Turbomachinery*, Vol. 122, No. 4, October, 2000, pp. 674-673.

3. Griffin, L. W. and Dorney, D. J., ``Simulations of the Unsteady Flow Through the Fastrac Supersonic Turbine,'' *ASME Journal of Turbomachinery,* Vol. 122, No. 2, April 2000, pp. 225-233.

4. Wang, T-S. and Chen, Y.-S., "Unified Navier-Stokes Flowfield and Performance Analysis of Liquid Rocket Engines," *Journal of Propulsion and Power*, Vol. 9, No. 5, Sept-Oct 1993, pp.678-685.

5. Lane, David A., ``Scientific Visualization of Large-Scale Unsteady Fluid Flows,'' Scientific Visualization, IEEE Computer Society Chapter 5, pp 125-145.

6. Lovely, David, Haimes Robert, ``Shock Detection from Computational Fluid Dynamics Results,'' AIAA Paper No. 99-3285, Norfolk VA, June, 1999.

```
Delta Report

base file:
 Results-FDNS500/fort.93
new file:
 Temp-Results-FDNS500/fort.93

file: fort.93 tolerance: 0.050000 (percent)

comparison of rho values:
  values within tolerance: 48000
  values not within tolerance: 0
  total values: 48000
  min and max values from base case:
    min: 0.000000    max: 1.291901
  min and max values from new case:
    min: 0.000000    max: 1.291901
  percent differences from base case:
    min: 0.0      max: 0.0

comparison of rhoU values:
  values within tolerance: 48000
  values not within tolerance: 0
  total values: 48000
  min and max values from base case:
    min: 0.239215    max: 1.257541
  min and max values from new case:
    min: 0.239215    max: 1.257541
  percent differences from base case:
    min: 0.0      max: 0.0
```

Figure 1: Partial DeltaReport

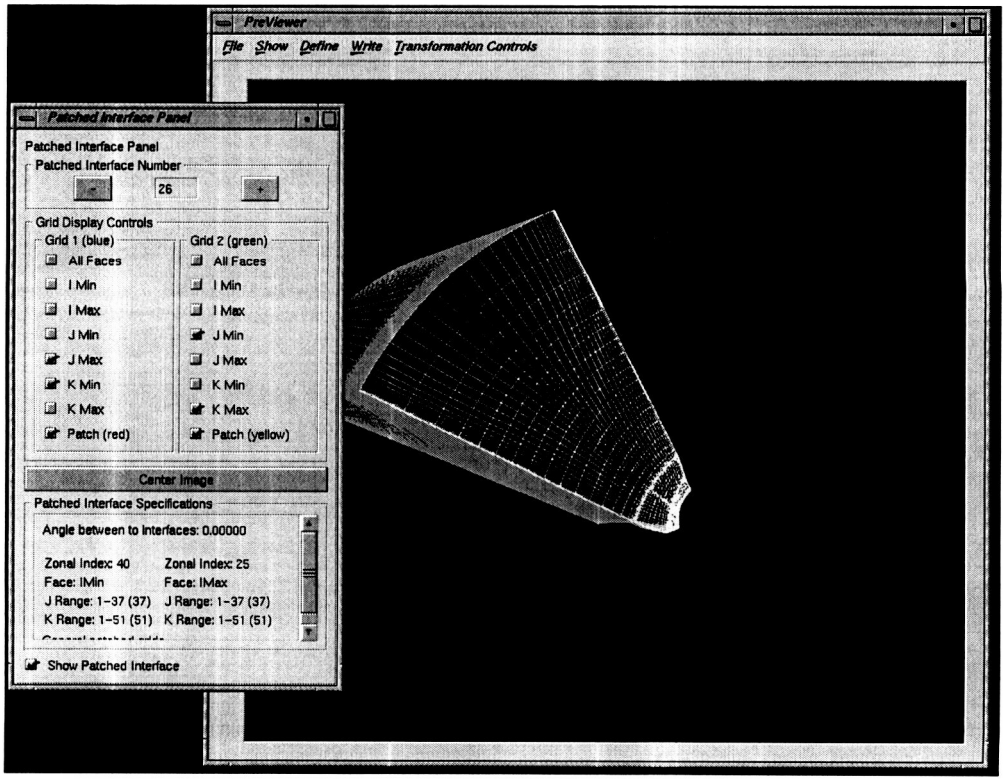

Figure 2: Patched Interface Panel from *PreViewer*

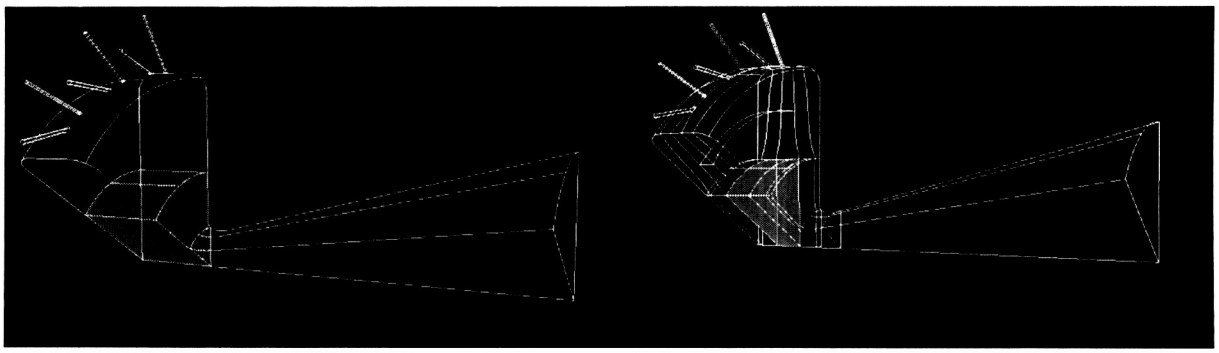

Figure 3: Vortex Chamber with 9 and 19 Grids

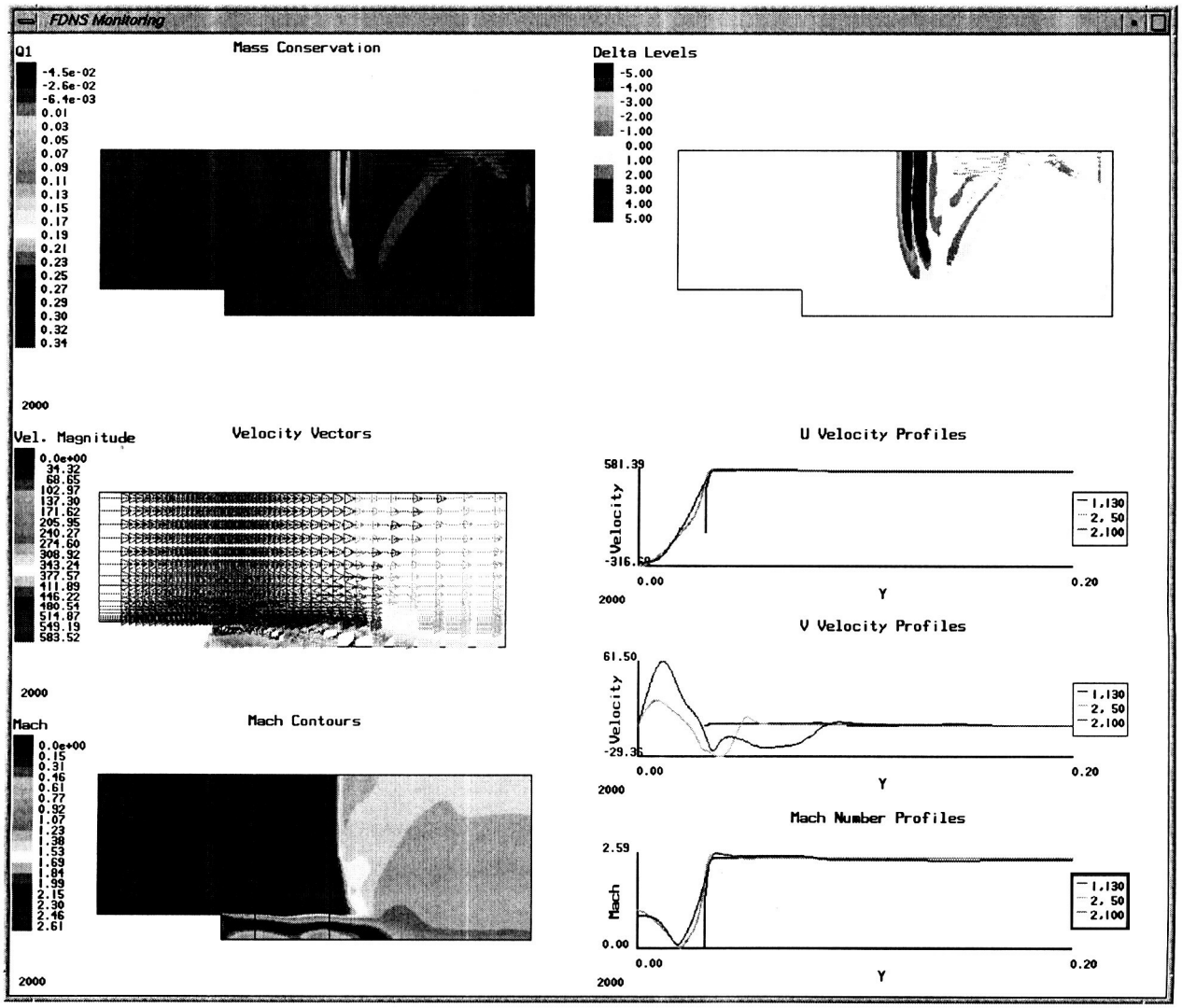

Figure 4: Results of Monitor

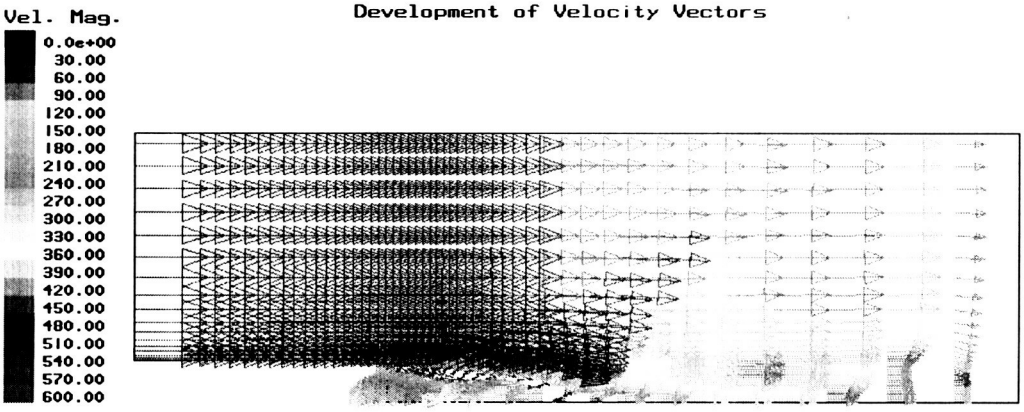

Figure 5: Formation of Flow Solution

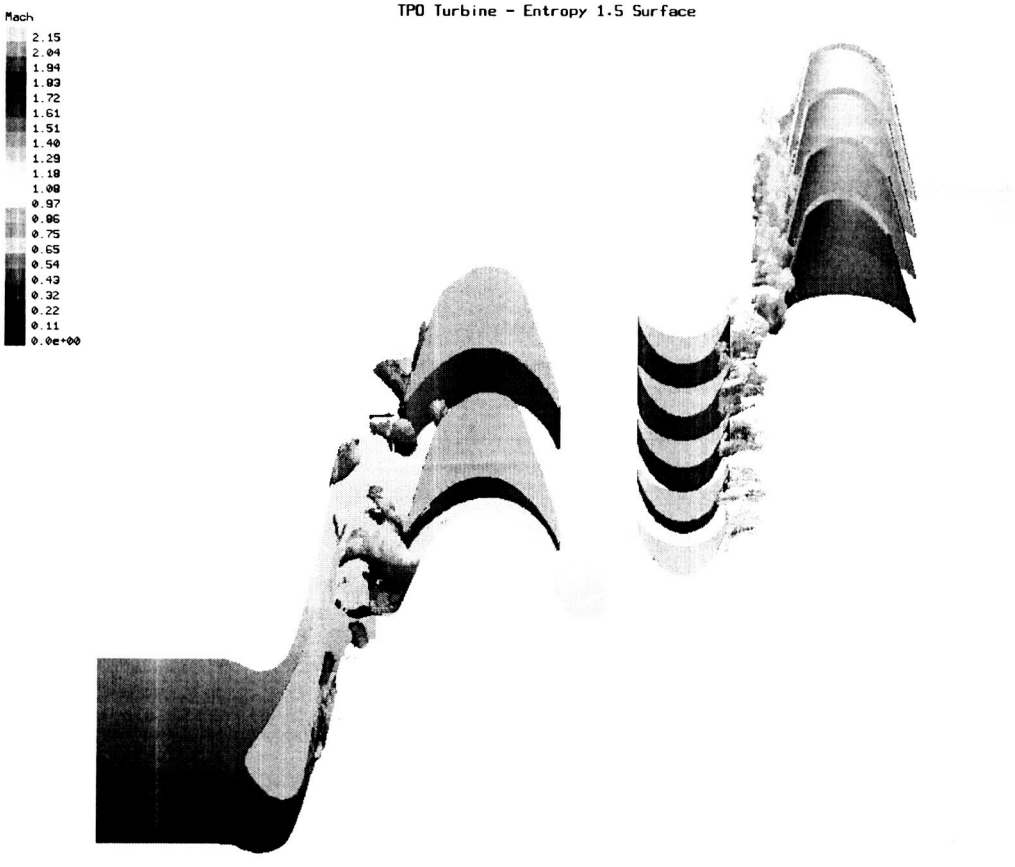

Figure 6: Entropy Contours on a Shock Surface