PARALLELIZATION OF THE FLOW FIELD DEPENDENT VARIATION SCHEME FOR SOLVING THE TRIPLE SHOCK/BOUNDARY LAYER INTERACTION PROBLEM

Richard Gregory Schunk
NASA/Marshall Space Flight Center

T. J. Chung
University of Alabama in Huntsville

ABSTRACT

A parallelized version of the Flowfield Dependent Variation (FDV) Method is developed to analyze a problem of current research interest, the flowfield resulting from a triple shock/boundary layer interaction. Such flowfields are often encountered in the inlets of high speed air-breathing vehicles including the NASA Hyper-X research vehicle. In order to resolve the complex shock structure and to provide adequate resolution for boundary layer computations of the convective heat transfer from surfaces inside the inlet, models containing over 500,000 nodes are needed. Efficient parallelization of the computation is essential to achieving results in a timely manner. Results from a parallelization scheme, based upon multi-threading, as implemented on multiple processor supercomputers and workstations is presented.

INTRODUCTION

The Flowfield Dependent Variation (FDV) Method is utilized to analyze a problem of current research interest, the flowfield produced from a triple shock/boundary layer interaction. Flow fields of this nature are often encountered in the inlets of high speed vehicles such as the scramjet engine of NASA's Hyper-X research vehicle. For this analysis, the numerical results are compared to experimental wind tunnel measurements made by Garrison, Settles, and Horstman [1,2]. The objective of the FDV analysis is to resolve the major flowfield structures observed during the experiment while demonstrating an efficient parallelization scheme based upon multi-threaded programming.

FLOWFIELD DEPENDENT VARIATION (FDV) THEORY

The original idea of FDV methods began from the need to address the physics involved in shock wave turbulent boundary layer interactions [3-5]. In this situation, transitions and interactions of inviscid/viscous, compressible/incompressible, and laminar/turbulent flows constitute not only the physical complexities but also computational difficulties. This is where the very low velocity in the vicinity of the wall and very high velocity far away from the wall coexist within a domain of study. Transitions from one type of flow to another and interactions between two distinctly different flows have been studied for many years both experimentally and numerically. Traditionally, incompressible flows were analyzed using the pressure-

based formulation with the primitive variables for the implicit solution of the Navier-Stokes system of equations together with the pressure Poisson equation. On the other hand, compressible flows were analyzed using the density-based formulation with the conservation variables for the explicit solution of the Navier-Stokes system of equations. In dealing with the domain of study which contains all speed flows with various physical properties where the equations of state for compressible and incompressible flows are different, and where the transitions between laminar and turbulent flows are involved in dilatational dissipation due to compressibility, we must provide very special and powerful numerical treatments. The FDV scheme has been devised toward resolving all of these issues.

To this end, let us consider the Navier-Stokes system of equations in conservation form,

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}_i}{\partial x_i} + \frac{\partial \mathbf{G}_i}{\partial x_i} = \mathbf{B} \tag{1}$$

In expanding $\mathbf{U}^{n+1}$ in a special form of Taylor series about $\mathbf{U}^n$, we introduce the variation parameters s_1 and s_2 for the first and second derivatives of $\mathbf{U}$ with respect to time, respectively

$$\mathbf{U}^{n+1} = \mathbf{U}^n + \Delta t \frac{\partial \mathbf{U}^{n+s_1}}{\partial t} + \frac{\Delta t^2}{2} \frac{\partial^2 \mathbf{U}^{n+s_2}}{\partial t^2} \tag{2}$$

where

$$\frac{\partial \mathbf{U}^{n+s_1}}{\partial t} = \frac{\partial \mathbf{U}^n}{\partial t} + s_1 \frac{\partial \Delta \mathbf{U}^{n+1}}{\partial t} \qquad 0 \le s_1 \le 1 \tag{3a}$$

$$\frac{\partial^2 \mathbf{U}^{n+s_2}}{\partial t^2} = \frac{\partial^2 \mathbf{U}^n}{\partial t^2} + s_2 \frac{\partial^2 \Delta \mathbf{U}^{n+1}}{\partial t^2} \qquad 0 \le s_2 \le 1 \tag{3b}$$

with $\Delta \mathbf{U}^{n+1} = \mathbf{U}^{n+1} - \mathbf{U}^n$. Substituting (3) into (2),

$$\Delta \mathbf{U}^{n+1} = \Delta t \left(\frac{\partial \mathbf{U}^n}{\partial t} + s_1 \frac{\partial \Delta \mathbf{U}^{n+1}}{\partial t} \right) + \frac{\Delta t^2}{2} \left(\frac{\partial^2 \mathbf{U}^n}{\partial t^2} + s_2 \frac{\partial^2 \Delta \mathbf{U}^{n+1}}{\partial t^2} \right) \tag{4}$$

Notice that s_1, associated with the first time derivative, is intended to provide variations as appropriate to the convection and diffusion processes of the flowfield, whereas s_2, involved in the second time derivative, is to control adequate application of artificial viscosity as required in accordance with the flowfield.

In the conservation form of the Navier-Stokes system of equations, $\mathbf{F}_i$ and $\mathbf{B}$ are functions of $\mathbf{U}$, and $\mathbf{G}_i$ is a function of $\mathbf{U}$ and its gradient $\mathbf{U}_{,k}$. Thus, by the chain rule of calculus, the first and second derivative of $\mathbf{U}$ with respect to time may be written as follows:

$$\frac{\partial \mathbf{U}}{\partial t} = -\frac{\partial \mathbf{F}_i}{\partial x_i} - \frac{\partial \mathbf{G}_i}{\partial x_i} + \mathbf{B} \tag{5a}$$

$$\frac{\partial^2 \mathbf{U}}{\partial t^2} = -\frac{\partial \mathbf{F}_i}{\partial \mathbf{U}} \frac{\partial}{\partial x_i} \left(\frac{\partial \mathbf{U}}{\partial t} \right) - \frac{\partial \mathbf{G}_i}{\partial \mathbf{U}} \frac{\partial}{\partial x_i} \left(\frac{\partial \mathbf{U}}{\partial t} \right) - \frac{\partial \mathbf{G}_i}{\partial \mathbf{U}_{,k}} \frac{\partial^2}{\partial x_i \partial x_k} \left(\frac{\partial \mathbf{U}}{\partial t} \right) + \frac{\partial \mathbf{B}}{\partial \mathbf{U}} \left(\frac{\partial \mathbf{U}}{\partial t} \right) \tag{5b}$$

We denote the convection Jacobian $\mathbf{a}_i$, the diffusion Jacobian $\mathbf{b}_i$, the diffusion gradient Jacobian $\mathbf{c}_{ik}$, and the source Jacobian $\mathbf{d}$ as

$$\mathbf{a}_i = \frac{\partial \mathbf{F}_i}{\partial \mathbf{U}} \quad , \qquad \mathbf{b}_i = \frac{\partial \mathbf{G}_i}{\partial \mathbf{U}} \quad , \qquad \boldsymbol{c}_{ik} = \frac{\partial \boldsymbol{G}_i}{\partial \boldsymbol{U}_{,k}} \quad , \qquad \mathbf{d} = \frac{\partial \mathbf{B}}{\partial \mathbf{U}}$$

For the purpose of generality, we assume here that the source terms arise from additional equations for chemical species equations.

The second derivative of **U** with respect to time may now be written in terms of these Jacobians by substitution into (5b),

$$\frac{\partial^2 \mathbf{U}}{\partial t^2} = \frac{\partial}{\partial x_i}\left(\mathbf{a}_i + \mathbf{b}_i\right)\left(\frac{\partial \mathbf{F}_j}{\partial x_j} + \frac{\partial \mathbf{G}_j}{\partial x_j} - \mathbf{B}\right) + \frac{\partial^2}{\partial x_i \partial x_k}\mathbf{c}_{ik}\left(\frac{\partial \mathbf{F}_j}{\partial x_j} + \frac{\partial \mathbf{G}_j}{\partial x_j} - \mathbf{B}\right) - \mathbf{d}\left(\frac{\partial \mathbf{F}_j}{\partial x_j} + \frac{\partial \mathbf{G}_j}{\partial x_j} - \mathbf{B}\right) \tag{6}$$

Substituting (5a) and (6) into (4), and assuming the product of the diffusion gradient Jacobian with third order spatial derivatives to be negligible, we have

$$\begin{aligned}
\Delta \mathbf{U}^{n+1} = {} & \Delta t\left[-\frac{\partial \mathbf{F}_i^n}{\partial x_i} - \frac{\partial \mathbf{G}_i^n}{\partial x_i} + \mathbf{B}^n + s_1\left(-\frac{\partial \Delta \mathbf{F}_i^{n+1}}{\partial x_i} - \frac{\partial \Delta \mathbf{G}_i^{n+1}}{\partial x_i} + \Delta \mathbf{B}^{n+1}\right)\right] \\
& + \frac{\Delta t^2}{2}\left\{\left[\frac{\partial}{\partial x_i}\left(\mathbf{a}_i + \mathbf{b}_i\right)\left(\frac{\partial \mathbf{F}_j^n}{\partial x_j} + \frac{\partial \mathbf{G}_j^n}{\partial x_j} - \mathbf{B}^n\right) - \mathbf{d}\left(\frac{\partial \mathbf{F}_i^n}{\partial x_i} + \frac{\partial \mathbf{G}_i^n}{\partial x_i} - \mathbf{B}^n\right)\right]\right. \\
& + s_2\left[\frac{\partial}{\partial x_i}\left(\mathbf{a}_i + \mathbf{b}_i\right)\left(\frac{\partial \Delta \mathbf{F}_j^{n+1}}{\partial x_j} + \frac{\partial \Delta \mathbf{G}_j^{n+1}}{\partial x_j} - \Delta \mathbf{B}^{n+1}\right)\right. \\
& \left.\left.\left. - \mathbf{d}\left(\frac{\partial \Delta \mathbf{F}_i^{n+1}}{\partial x_i} + \frac{\partial \Delta \mathbf{G}_i^{n+1}}{\partial x_i} - \Delta \mathbf{B}^{n+1}\right)\right]\right\} + \mathrm{O}\left(\Delta t^3\right)
\end{aligned} \tag{7}$$

The variation parameters s_1 and s_2 which appear in (7) may be accorded with appropriate physical roles by calculating them from the flowfield-dependent quantities. For example, if s_1 is associated with the temporal changes (Δ terms, henceforth called *fluctuations*, not meant to be turbulent fluctuations) of convection, it may be calculated from the spatial changes of Mach number between adjacent nodal points so that $s_1 = 0$ would imply no changes in convection fluctuations. Similarly, if s_1 is associated with the fluctuations of diffusion, then it may be calculated from the spatial changes of Reynolds number or Peclet number between adjacent nodal points such that $s_1 = 0$ would signify no changes in diffusion fluctuations. Therefore, the role of s_1 for diffusion is different from that of convection. Similarly, the role of s_1 for the fluctuation of the sources (such as reaction rates and heat generation) should be different from convection and diffusion. For example, we may define the fluctuation quantities associated with s_1 as

$$s_1\left(\frac{\partial \Delta \mathbf{F}^{n+1}_{\ i}}{\partial x_i} + \frac{\partial \Delta \mathbf{G}^{n+1}_{\ i}}{\partial x_i} - \Delta \mathbf{B}^{n+1}\right) \quad \Rightarrow \quad s_{1c}\frac{\partial \Delta \mathbf{F}^{n+1}_{\ i}}{\partial x_i} + s_{1d}\frac{\partial \Delta \mathbf{G}^{n+1}_{\ i}}{\partial x_i} - s_{1s}\,\Delta \mathbf{B}^{n+1}$$

$$= \frac{\sqrt{M^2_{\max} - M^2_{\min}}}{M_{\min}}\frac{\partial \Delta \mathbf{F}_i^{n+1}}{\partial x_i} + \frac{\sqrt{Re^2_{\max} - Re^2_{\min}}}{Re_{min}}\frac{\partial \Delta \mathbf{G}^{n+1}_{\ i}}{\partial x_i} - \frac{\sqrt{Da^2_{\max} - Da^2_{\min}}}{Da_{\min}}\Delta \mathbf{B}^{n+1} \tag{8}$$

where it is seen that the variation parameter s_1 originally adopted as a single mathematical or numerical parameter has now turned into multiple physical parameters such as the changes of Mach numbers, Reynolds numbers (or Peclet numbers), and Damkohler numbers (Da), between adjacent nodal points. The magnitudes of fluctuations of convection, diffusion, and source terms are dictated by the current flowfield situations in space and time. Similar assessments can be applied to the variation parameter s_2 as associated with its corresponding fluctuation terms of convection, diffusion, and source. Thus, in order to provide variations to the changes of convection, diffusion, and source terms differently in accordance with the current flowfield situations, we reassign s_1 and s_2 associated with convection, diffusion and source terms as follows:

$$s_1\Delta\mathbf{G}_i \Rightarrow s_{1d}\Delta\mathbf{G}_i = s_3\Delta\mathbf{G}_i \quad , \quad s_1\Delta\mathbf{B} \Rightarrow s_{1s}\Delta\mathbf{B}_i = s_5\Delta\mathbf{B}$$

$$s_2\Delta\mathbf{G}_i \Rightarrow s_{2d}\Delta\mathbf{G}_i = s_4\Delta\mathbf{G}_i \quad , \quad s_2\Delta\mathrm{B} \Rightarrow s_{2s}\Delta\mathbf{B} = s_6\Delta\mathbf{B}$$

with the various variation parameters defined as

$s_{1c} = s_1$ = first order convection variation parameter

$s_{2c} = s_2$ = second order convection variation parameter

$s_{1d} = s_3$ = first order diffusion variation parameter

$s_{2d} = s_4$ = second order diffusion variation parameter

$s_{1s} = s_5$ = first order source term variation parameter

$s_{2s} = s_6$ = second order source term variation parameter

The first order variation parameters s_1, s_3, and s_5 are flowfield-dependent, whereas the second order variation parameters s_2, s_4, and s_6 are exponentially proportional to the first order variation parameters, and mainly act as artificial viscosity.

EXPERIMENTAL MEASUREMENTS

The analytical results are compared to experimental measurements for a triple shock interaction obtained by Garrison, Settles, and Horstman [1,2]. The wind tunnel model used to produce the triple shock/boundary layer interaction consists of two vertical fins and a horizontal ramp as shown in Figure 1. The angle of attack for the fins is 15° and the ramp is inclined at an angle of 10° with respect to the inlet flow. The inlet flow is at Mach 3.85 with a stagnation temperature and pressure of 295K and 1500 kPa, respectively. The fins are 82.5 mm high and are separated by a distance of 96.3 mm. The leading edge of the model is located 21 cm in front of the ramp inlet and produces a turbulent boundary layer with a thickness of 3.5 mm at the inlet to the model. Flow through the model is characterized by three oblique shocks originating from the leading edges of the ramp and the fins. Above the oblique ramp shock, the two inviscid fin shocks intersect and reflect as shown in the figure. For the purposes of this analysis, the ramp is assumed to be 120 mm in length, the distance at which the reflected inviscid fin shocks are just incident upon the exit corners of each fin. According to inviscid flow theory, the fin shocks should intersect approximately 92 mm from the combined ramp and fin entrance. Measurements of the flowfield structure in the x-y plane are made via the Planar Laser Scattering (PLS) technique at various depths upstream of, coincident with, and behind the inviscid fin shock intersection [2].

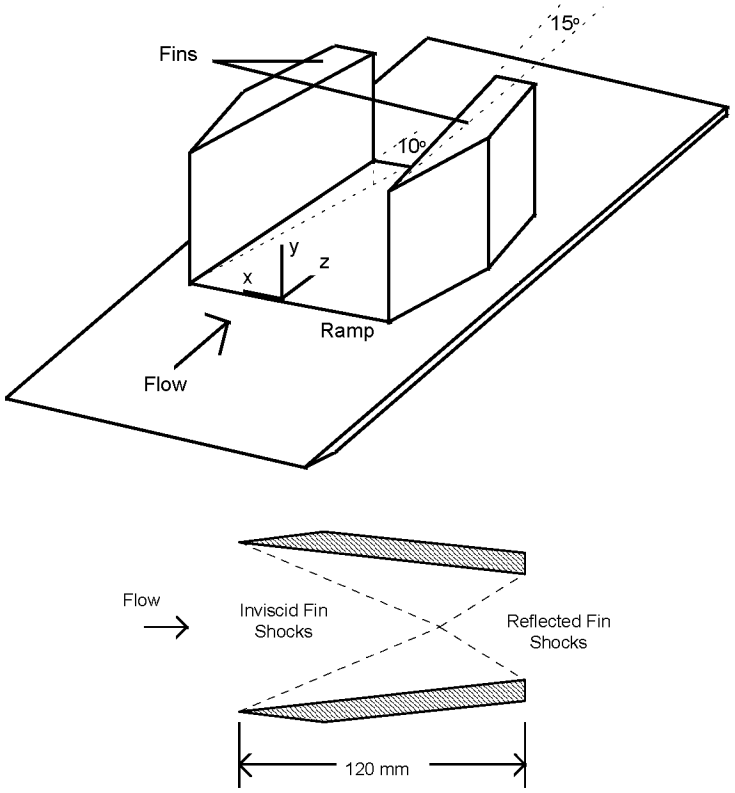
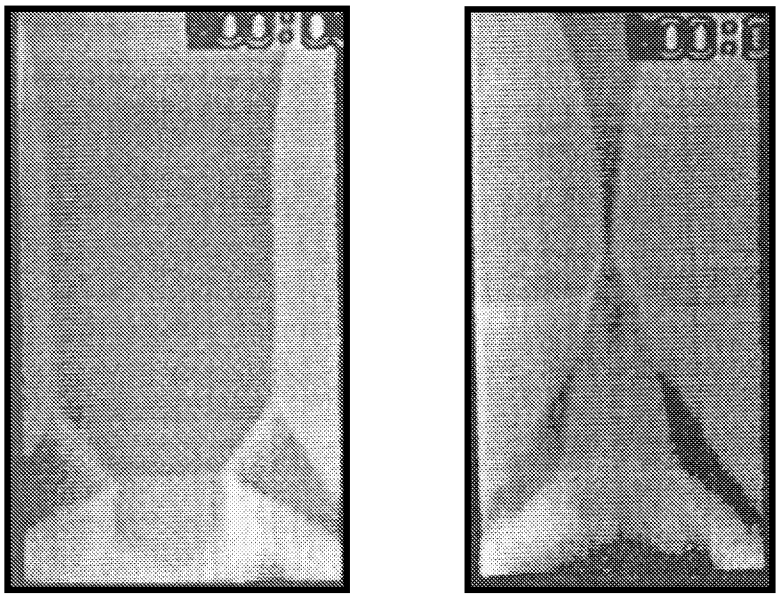


Figure 1: Inviscid Fin Shock Reflection (Top View, X-Z Plane)

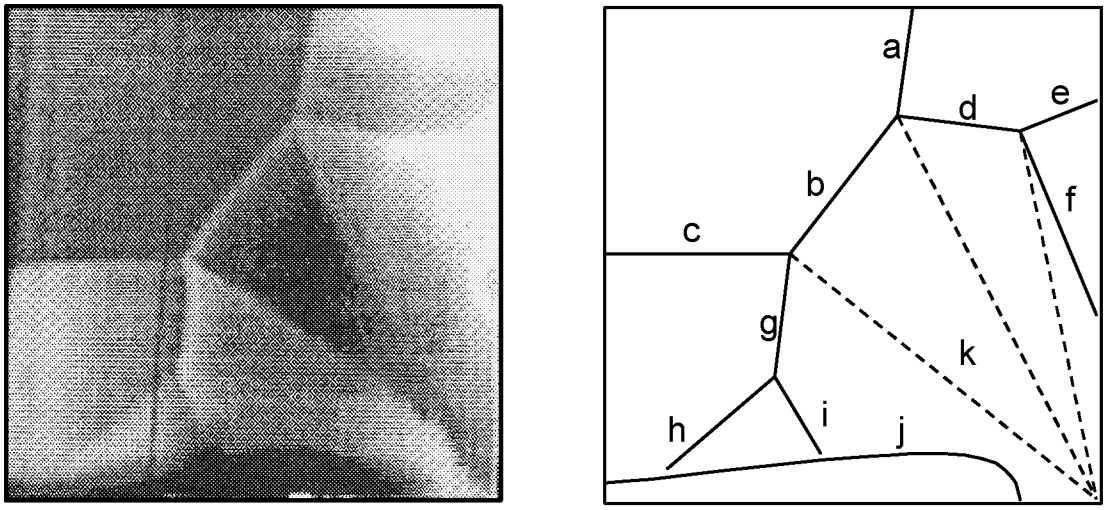
Of particular interest in this analysis is the complex shock/boundary layer interaction produced in the x-y planes perpendicular to the flow direction. Upstream of and coincident with the inviscid ramp shock intersection, the fin and ramp shocks are reflected and interact with the fin and ramp boundary layers to produce the shock structures contained in the PLS images of Figure 2. Upstream of the shock intersection (left), the flow is characterized by the inviscid fin and ramp shocks reflecting to form a corner shock. Slip lines separating the shock induced flow regions are also visible in the image. The flow separation from the ramp underneath the embedded fin shock is also visible in the image. At the shock intersection (right), the inviscid fin shocks merge, the ramp shock dissapears, and the corner shock is reflected to form the structure shown. The curvature of the fin shocks become more pronounced and a large separation region is observed underneath the reflected corner shocks. This is attributed to the curvature of the inviscid shocks to the finite height of the fins (i.e. there would be no shock curvature with fins of "infinite" height) [1].



(a) Z=6.7 cm (b) Z=9.2 cm

Figure 2: Fin/Ramp Shock Structure in the X-Y Plane Before (Left) and Co-incident (Right) the Inviscid Fin Shock Intersection [1]

A detailed PLS view of the corner shock reflection physics is shown in Figure 3. [1]. As shown in the figure, the inviscid fin (a) and ramp (b) shocks reflect to form the corner (c) shock. Both the embedded ramp (d) and fin (g) shocks split into separation (e,h) and rear (f,i) shocks above the ramp and fin boundary/separation layers. The ramp separated region (j) and the slip lines (k) dividing the different velocity regions as induced by the shock structure are also visible in the image.



PLS Image of Corner Flow **Corresponding Flowfield Structure**

Figure 3: Fin/Ramp Shock Structure in the X-Y Plane [1], a) Inviscid Fin Shock, b) Corner Shock, c) Inviscid Ramp Shock, d) Embedded Ramp Shock, e) Ramp Separation Shock, f) Ramp Rear Shock, g) Embedded Fin Shock, h) Separation Fin Shock, i) Rear Fin Shock, j) Separated Region, k) Sliplines

COMPUTER MODEL

Since the two fins are symmetric about the centerline, only half of the wind tunnel model is included in the computational model. Two finite difference computational grids, varying in resolution, are developed for the FDV analysis. The coarse grid model, consisting of a non-uniform nodal resolution of 31 x 41 x 55 (in the x, y, and z directions) is shown in Figure 4. The viscous grid is clustered close to the fin and ramp surfaces. Results from the coarse grid analysis are used as the starting condition for the fine grid model. The fine grid model is obtained by interpolating the flow variables against the coarse mesh. Doubling the number of grid points in each direction produces a fine grid with over 538,000 nodal points (61 x 81 x 109). Recall that the most important aspect of the FDV theory is that the shock capturing mechanism and the transition and interaction between compressible/incompressible, viscous/inviscid, and laminar/turbulent flows are incorporated into the FDV formulation. No special treatments are required to simulate these physical phenomena. Thus, the finite difference discretization requires no special schemes. Simple central differences can be used to discretize the FDV.

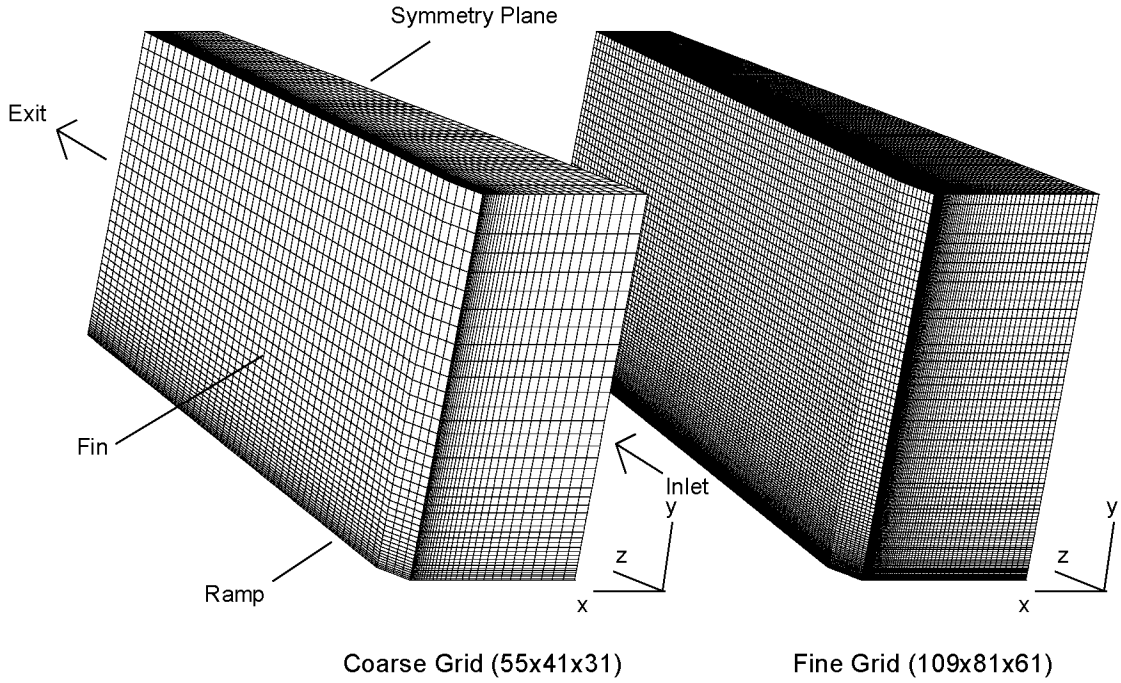


Figure 4: Three Dimensional Finite Difference Models

The inlet conditions to the model are fixed with the freestream conditions described above (M=3.85, Po=1500 kPa, and To=295 K) and include a superimposed boundary layer 3.5 mm in height[1]. At the fin and ramp surfaces, no-slip velocity boundary conditions are imposed and the normal pressure and temperature gradients are set to zero. In the symmetry plane and for the bounding surface on top (x-z plane), all of the flow variables are computed such that the normal gradients vanish except for the normal flux, which is explicitly set to zero. At the exit, all of the flow variables are extrapolated from interior grid points.

PARALLELIZATION STRATEGY: MULTI-THREADED PROGRAMMING AND DOMAIN DECOMPOSITION

Multi-threaded programming is utilized to take advantage of multiple computational elements on the host computer. Typically, a multi-threaded process will spawn multiple threads which are allocated by the operating system to the available computational elements (or processors) within the system. If more than one processor is available, the threads may execute in parallel resulting in a significant reduction in execution time. If more threads are spawned than available processors, the threads appear to execute concurrently as the operating system decides which threads execute while the others wait. One unique advantage of multi-threaded programming on shared memory multiprocessor systems is the ability to share global memory. This alleviates the need for data exchange or message passing between threads as all global memory allocated by the parent process is available to each thread. However, precautions must be taken to prevent deadlock or race conditions resulting from multiple threads trying to simultaneously write to the same data.

Threads are implemented by linking an application to a shared library and making calls to the routines within that library. Two popular implementations are widely used: the Pthreads library [6] (and its derivatives) that are available on most Unix operating systems and the NTthreads library that is available under Windows NT. There are differences between the two implementations, but applications can be ported from one to the other with moderate ease and many of the basic functions are similar albeit with different names and syntax.

Domain decomposition methods [7] can be used in conjunction with multi-threaded programming to create an efficient parallel application. The sub-domains resulting from the decomposition provide a convenient division of labor for the processing elements within the host computer. In this application, an Additive Schwarz domain decomposition [7] method is utilized. The method is illustrated below (Figure 5) for a two dimensional square mesh that is decomposed into four sub-domains. The nodes belonging to each of the four sub-domains are denoted with geometric symbols while boundary nodes are identified with bold crosses. The desire is to solve for each node implicitly within a single sub-domain. For nodes on the edge of each sub-domain this is accomplished by treating the adjacent node in the neighboring sub-domain as a boundary. The overlapping of neighboring nodes between sub-domains is illustrated in Figure 6. Higher degrees of overlapping, which may improve convergence at the expense of computation time, are also used.

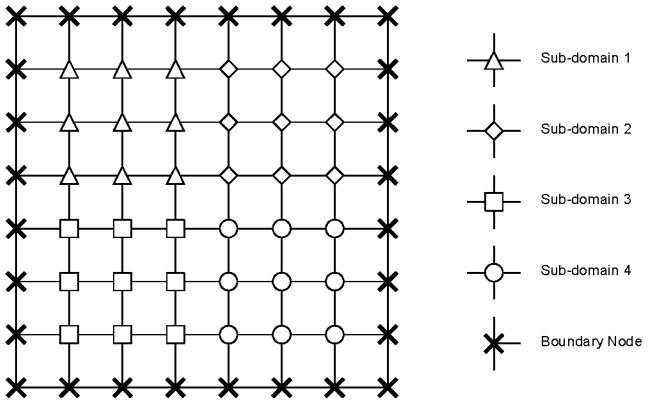


Figure 5: Multiple Subdomains

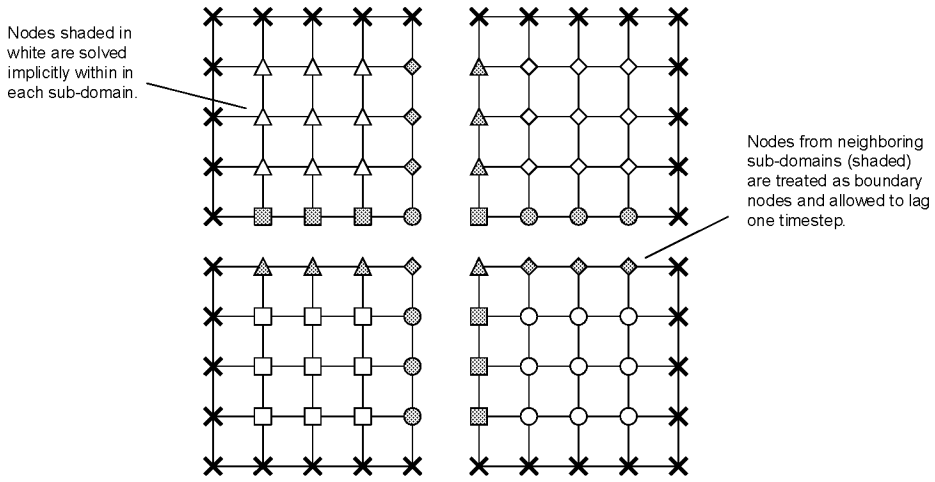


Figure 6: Domain Decomposition

In a parallel application, load balancing between processors is critical to achieving optimum performance. Ideally, if a domain could be decomposed into regions requiring an identical amount of computation, it would be a simple matter to divide the problem between processing elements as shown in Figure 7 for four threads executing on an equal number of processors.

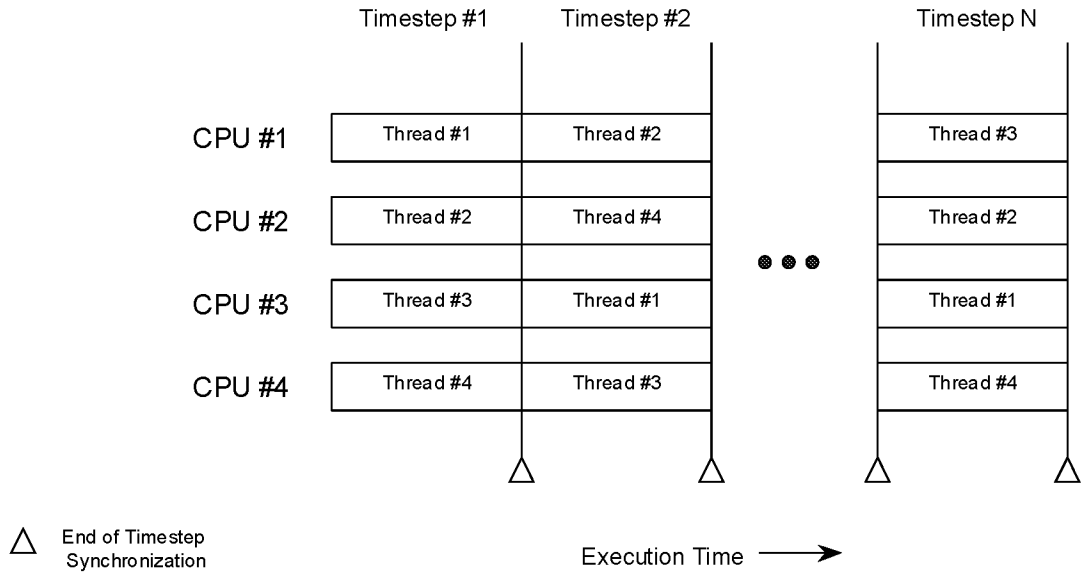


Figure 7: Ideal Load Balancing

Unfortunately, in a "real world" application the domain may not be decomposed such that the computation for each processor is balanced, resulting in lost efficiency. If the execution time required for each sub-domain is not identical, the CPU's will become idle for portions of time as shown in Figure 8.

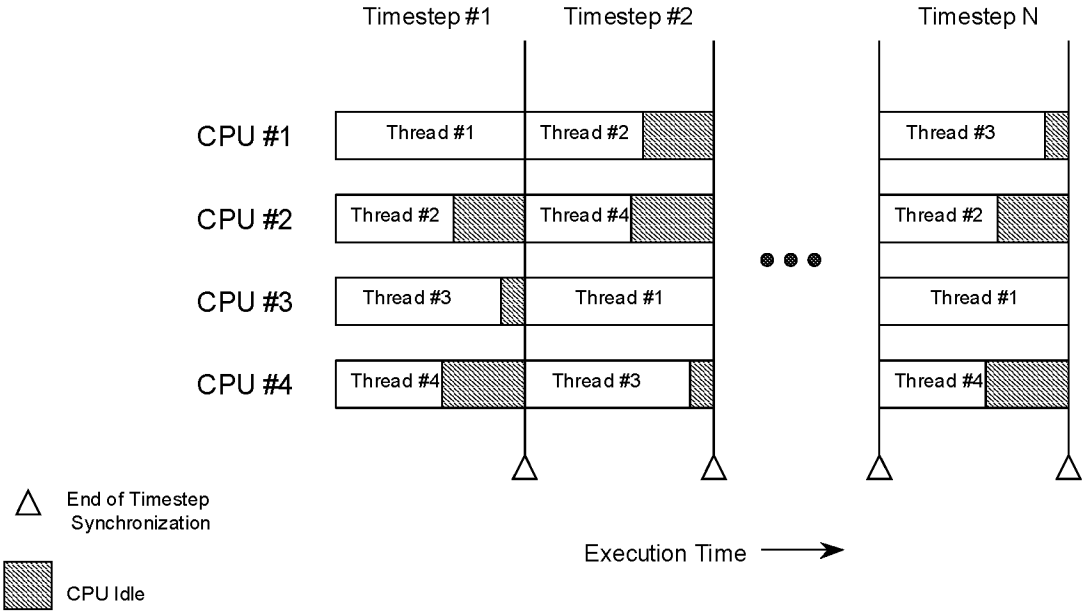


Figure 8: "Real World" Load Balancing

One approach to load balancing, as implemented in this application, is to decompose the domain into more sub-domains than available processors and use threads to perform the computations within each block. The

finer granularity permits a more even distribution of work amongst the available processing elements as shown in Figure 9.

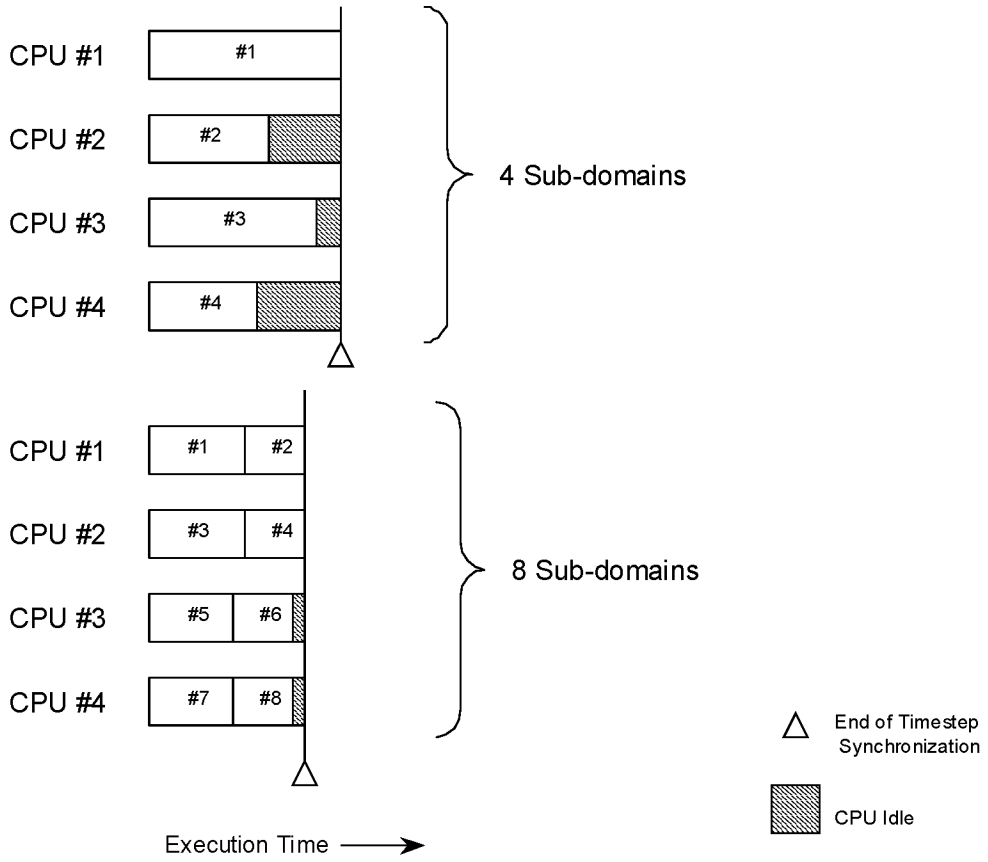
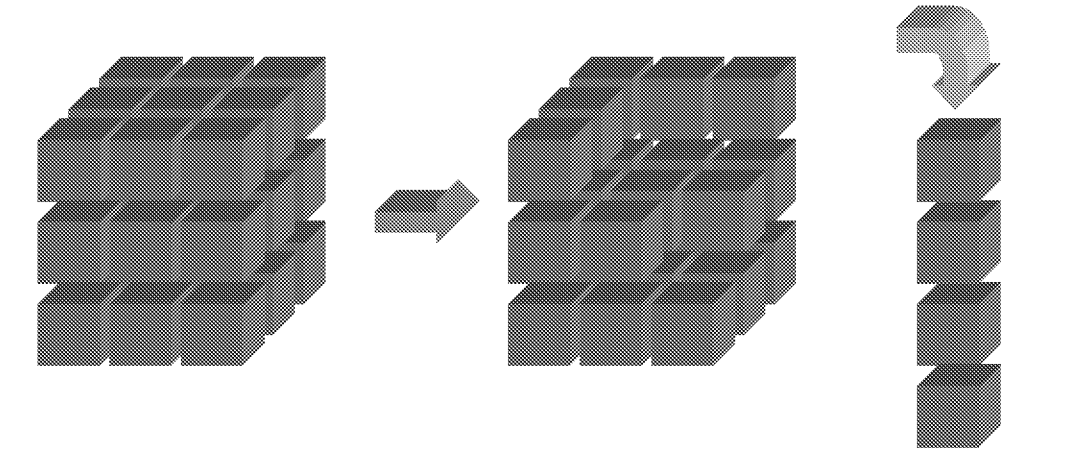


Figure 9: Domain Decomposition Improves Parallelism

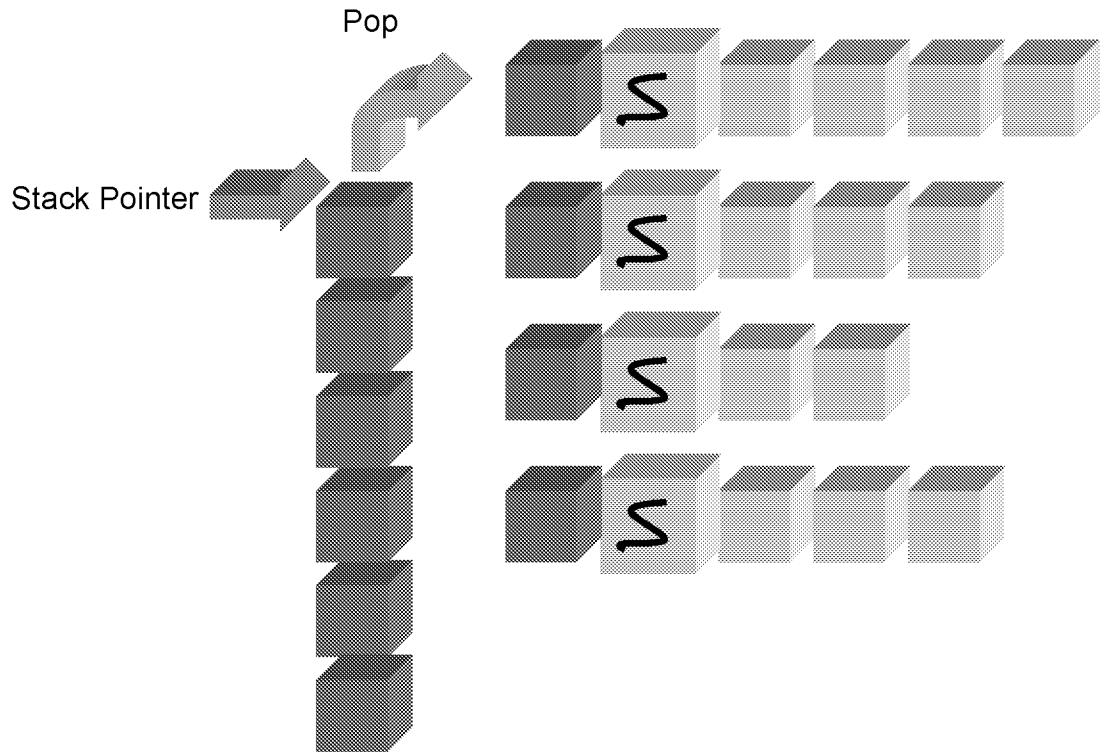
In this approach, the number of threads spawned is equal to the number of available processors with each thread marching through the available sub-domains (which preferably number at least two times the number of processors), solving one at a time in an "assembly-line" fashion. A stack is employed where each thread pops the next sub-domain to be solved off of the top of the stack. Mutual exclusion locks are employed to protect the stack pointer in the event two or more threads access the stack simultaneously. Each thread remains busy until the number of sub-domains is exhausted. If the number of sub-domains is large enough, the degree of parallelism will be high although decomposing a problem into too many sub-domains may adversely affect convergence. This approach is illustrated in Figures 10 and 11.



1) Decompose the domain

2) Push each sub-domain onto a software stack

Figure 10: Decompose the Domain and push onto Stack



3) Spawn threads and execute until stack is exhausted

Figure 11: Allow Threads to Process each Sub-domain

The coarse mesh computations were performed on a four processor Alpha™ based workstation located at the University of Alabama in Huntsville and on a dual processor Pentium™ II workstation located at the Marshall Space Flight Center. The fine mesh computations were conducted on SGI™ Origin 2000 and Power Challenge XL supercomputers (each containing twelve processors) located at the Marshall Space

Flight Center. The FDV application solver is based upon the Generalized Minimum Residual (GMRES) algorithm described by Shakib [8]. The application is coded to be multi-threaded to take advantage parallelism in the host computer. The number of threads is specified at run time and is based upon the expected number of available CPU's. The results for three different architectures are provided in Table 1. Typical utilizations (defined as CPU time/elapsed time) range from 180% to 380% for two to eight threads. It should be noted that both the number of threads and number of processors impose theoretical limits on the maximum performance gain. Obviously, the normalized performance increase can not exceed the number of threads and, aside from tertiary performance issues (such as on processor cache), nor can the normalized performance increase exceed the number of processors. For the coarse mesh model, actual speed increases range from 1.77 to 3.44 for 2 to 4 processors. The results are encouraging when considering the CPU contention between multiple users on the host machines. For the coarse mesh model on a dual processor Pentium II workstation (with no other users) a CPU utilization of 196% is observed with a real time speedup of 1.92. The four processor machine did exhibit a significant amount of overhead when moving beyond a single thread. Utilizing four threads for the fine mesh model resulted in CPU utilizations of 357% and 370% for a domain decomposed into 27 and 64 regions, respectively. The fine mesh model was not run with a single processor or thread so no relative speed-up data is available. The CPU utilization is encouraging considering the high CPU contention on the twelve processor machine.

Table 1. Computational Performance Summary

Threads	Grid	Decomposition	CPU Time (hours)	Elapsed Time (hours)	CPU Utilization	Speed-up	Processor	Number of Proc
1	55x41x31	4x4x4	5.05	5.05	100%	1.00	Pentium II	2
2	55x41x31	4x4x4	5.13	2.62	196%	1.93	Pentium II	2
1	55x41x31	4x4x4	4.69	4.72	99%	1.00	Alpha	4
2	55x41x31	4x4x4	5.19	2.66	195%	1.77	Alpha	4
4	55x41x31	4x4x4	5.30	1.42	373%	3.32	Alpha	4
6	55x41x31	4x4x4	5.30	1.40	378%	3.36	Alpha	4
8	55x41x31	4x4x4	5.16	1.37	377%	3.44	Alpha	4
4	109x81x61	3x3x3	37.40	10.47	357%	NA	R10000	12
4	109x81x61	4x4x4	52.76	14.25	370%	NA	R10000	12

Density contours for the inviscid shock interaction (x-z plane, as viewed from above the wind tunnel model) are shown in Figure 12. The 15° fins produce inviscid shocks that are predicted to intersect and reflect approximately 92 mm from the ramp entrance. The reflected shock does not intersect with the exit corner of the ramp as expected. Two cross sections, located at 67 mm and 92 mm, respectively, from the entrance are noted on the plot.

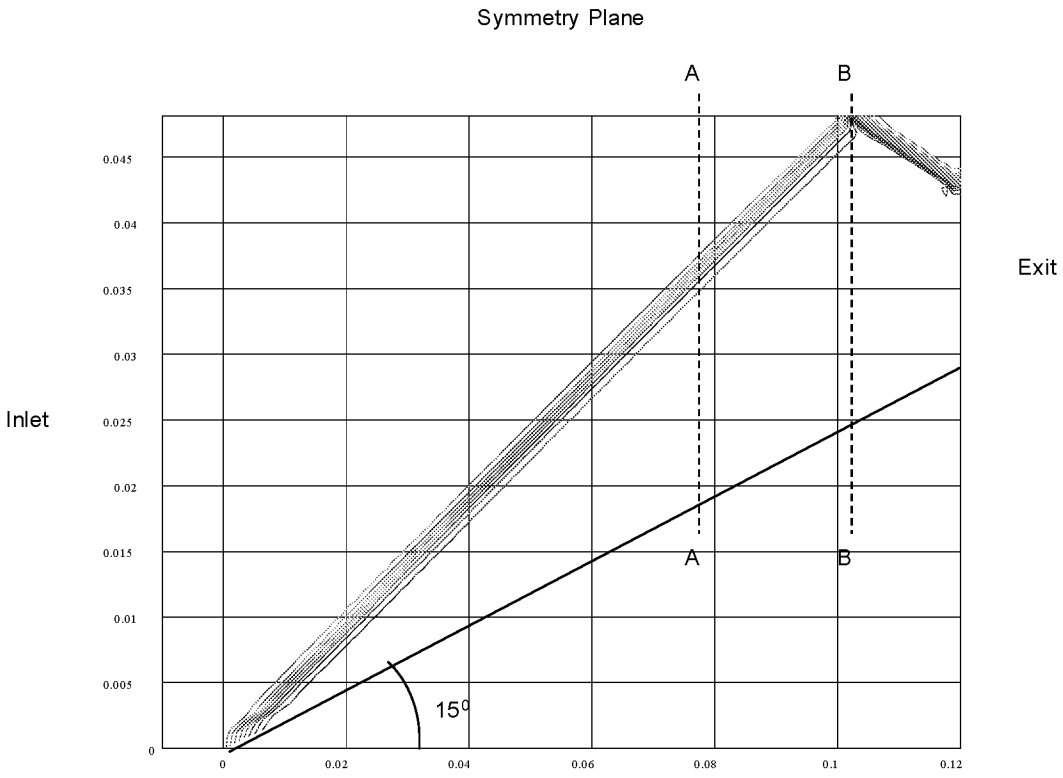


Figure 12: Density Contours for X-Z Cross Section (Top), Slip Boundary

Density contours for the flow in x-y planes located 67 mm (upstream of the inviscid shock intersection) and 92 mm (coincident with the inviscid shock intersection) from the combined fin/ramp entrance are shown in Figure 13. It appears that the upstream predictions correlate well with the experimental images. The inviscid ramp and fin shocks, as well as the corner reflection, are easily discernible in the upstream figure (see left). Interestingly, it appears that the triangular shaped slip lines are present in the numerical results of the upstream plane. Since the slip-lines divide constant pressure regions with differing velocities, this feature is not visible in the static pressure plots. As in the experimental imagery, the inviscid fin shocks merge together in the symmetry plane at the point where the inviscid shocks intersect (see right). No curvature of the inviscid fin shock intersection is observed in the numerical predictions. The reflection of the corner shock about the symmetry plane is observed, but the ramp embedded shock is lower relative to the height of the fin than in the experimental results.

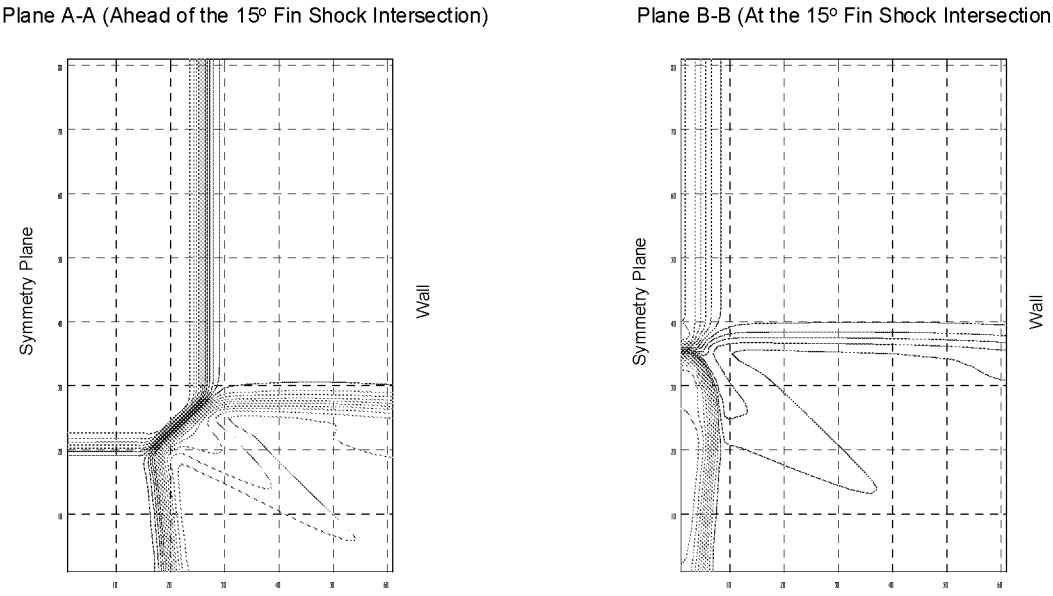


Figure 13: Density Contours for Y-Z Cross Section, Slip Boundary

CONCLUSIONS

The comparison of the FDV method to the actual measured flowfield for the triple shock interaction is encouraging. Many of the flowfield features observed in the experimental imagery are resolved in the computation including the inviscid shock corner reflections. Particularly good results are obtained for the shock structure in the cross sectional x-y planes upstream of the inviscid shock intersection. The numerical results did not exhibit the shock curvatures evident in the experimental images, but this may be rectified through increased grid resolution or a different boundary condition for the top surface (x-z plane) may be required. It is concluded that the multi-threaded domain decomposition approach provides an efficient strategy for parallelizing the FDV code and it is expected to be implemented on problems of increasing size in the future.

ACKNOWLEDGEMENTS

The authors wish to acknowledge Professor Krishna Kavi and members of the "Crash" group at the University of Alabama in Huntsville Electrical and Computer Engineering Department for their support.

REFERENCES

1. Garrison, T. J., Settles G. S., Naranyanswami, N., Knight D. D., and Horstman, C. C., "Flowfield Surveys and Computations of a Crossing Shock Wave/Boundary Layer Interaction", AIAA Paper 94-2273, June 1994.

2. Garrison, T. J., Settles G. S., and Horstman, C. C., "Measurements of Triple Shock Wave/Turbulent Boundary Layer Interaction", AIAA Paper 94-2274, June 1994.

3. Yoon, K. T.and Chung, T. J., "Three-dimensional mixed explicit-implicit generalized Galerkin spectral element methods for high-speed turbulent compressible flows", Computer Methods in Applied Mechanics and Engineering, 135,343-367 (1996)

4. Yoon, K. T., Moon, S. Y., Garcia, G. A., Heard, G. W., and Chung, T. J., "Flowfield dependent mixed explicit-implicit (FDMEI) methods for high and low speed and compressible and incompressible flows", Computer methods in Applied Mechanics and Engineering, 151, 75-104 (1998)

5. Chung, T. J., "Transitions and interactions of inviscid/viscous, compressible/incompressible and laminar/turbulent flows", Int. J. Numer. Meth. Fluids, in press.

6. Nichols B., Buttlar D., and Farrell J., "Pthreads Programming", O'Reilly and Associates, Inc., (1996).

7. Demmel, J.W., "Applied Numerical Linear Algebra", SIAM, Philadelphia, 347-351, (1997).

8. Shakib, F., Hughes, T., and Zdenek J.,"A Multi-element Group Preconditioned GMRES Algorithm for Non-symmetric Systems Arising in Finite Element Analysis", Computer Methods in Applied Mechanics and Engineering (1989).