NASA-CR-186033

NASA Contractor Report 186033

NASA-CR-186033
19960009469

A Computational Fluid Dynamics Simulation of the Hypersonic Flight of the Pegasus™ Vehicle Using an Artificial Viscosity Model and a Nonlinear Filtering Method

John Cadiz Mendoza

FOR REFERENCE
NOT TO BE TAKEN FROM THIS ROOM

September 1995

LIBRARY COPY
OCT 3 1995
LANGLEY RESEARCH CENTER
LIBRARY NASA
HAMPTON, VIRGINIA

National Aeronautics
and
Space Administration

NF00260

NASA Technical Library
3 1176 01422 8853

NASA Contractor Report 186033

A Computational Fluid Dynamics Simulation of the Hypersonic Flight of the Pegasus™ Vehicle Using an Artificial Viscosity Model and a Nonlinear Filtering Method

John Cadiz Mendoza
University of California, Los Angeles
Los Angeles, California

1995

National Aeronautics and
Space Administration

Dryden Flight Research Center
Edwards, California 93523-0273

This Page Intentionally Left Blank

Contents

List of Figures iv

List of Symbols vi

Abstract viii

Chapter Page

I. Introduction 1

II. PARC3D 6

III. Engquist Filter 18

IV. MacCormack Scheme 26

V. Test Conditions 31

VI. Results and Discussion 46

VII. Conclusions and Future Work 86

Appendix A: Engquist Filter Programs 88

Appendix B: Shock-Boundary Layer Interactions Convergence Histories 102

References 109

List of Figures

Figure Page

1. Engquist Filter Applied at Node j 21

2. x-t Space Characteristics 29

3. One-Dimensional Shock Tube 32

4. Convergent Duct Geometry 35

5. Shock-Boundary Layer Interaction 35

6. Pegasus Vehicle 39

7. Complete Pegasus Computational Grid 41

8. Magnified View of Fillet Area 43

9. Shock Tube Results at t = 0.00233 s 47

10. 2-D Convergent Duct: CFL = 0.8 49

11. 2-D Convergent Duct: Convergence History 50

12. 2-D Convergent Duct: CFL = 1.25 54

13. Duct Results For Increasing Deflection Angle 55

14. Shock-Boundary Layer Interaction: $P_f/P_i = 1.2$ 57

15. Shock-Boundary Layer Interaction: $P_f/P_i = 1.4$ 62

16. Shock-Boundary Layer Interaction: Strong Shock $P_f/P_i = 2.4$

    16a. Skin Friction Coefficient 66
    16b. Surface Pressure 67
    16c. Skin Friction Coefficient: Expanded View 68
    16d. Surface Pressure: Expanded View 69

17. HRSI Plug Location (Side View of Vehicle) 71

18. Heat Flux at HRSI Plugs:

Mach 3.52, $\alpha = 7.35°$, Flight 001 74

19. Heat Flux at HRSI Plugs:

Mach 3.52, $\alpha = 2.65°$, Flight 002 75

20. Heat Flux at HRSI Plugs:

Mach 6.67, $\alpha = 0°$, Flight 001 76

21. Heat Flux at HRSI Plugs:

Mach 6.67, $\alpha = 0°$, Flight 002 77

22. Heat Flux at HRSI Plugs:

Mach 5.0, $\alpha = 0.5°$, Flight 002 78

23. Temperature Gradient Near Fillet Wall 83

24. Pressure at HRSI Plugs 85

25. Shock-Boundary Layer: Convergence History $P_f/P_i = 1.2$

25a. DIS2 = 0.25 103
25b. DIS2 = 0 No Filter 104
25c. DIS2 = 0 With Filter 105

26. Shock-Boundary Layer: Convergence History $P_f/P_i = 1.4$

26a. DIS2 = 0.25 106
26b. DIS2 = 0 No Filter 107
26c. DIS2 = 0 With Filter 108

List of Symbols

α	=	Angle of Attack (degrees)
CFD	=	Computational Fluid Dynamics
NASA	=	National Aeronautics and Space Administrations
HRSI	=	High-temperature Reusable Surface Insulation
Q	=	Conservation Variable Vector
F_j	=	Inviscid Flux Vector
G_j	=	Viscous Flux Vector
X_j	=	Cartesian Coordinate Direction
t	=	Time (s)
Re	=	Reynolds Number
ρ	=	Density (kg/m^3)
u_i	=	Velocity in the Cartesian Coordinate Direction
E	=	Total Energy per Unit Volume
P	=	Pressure (N/m^2)
δ_{ij}	=	Kronecker Delta
τ_{ij}	=	Shear Force (N/m^2)
q_j	=	Heat Flux in Coordinate Direction (W/m^2)
μ	=	Dynamic Viscosity (kg/m-s)
λ	=	Second Coefficient of Viscosity (kg/m-s)
K	=	Thermal Conductivity (W/m-K)

Pr	=	Prandtl Number
T	=	Temperature (K)
ξ_i	=	Computational Coordinate Direction
γ	=	Ratio of Specific Heats
T_{Suth}	=	Sutherland Reference Temperature (110.3 K)
J	=	Jacobian
A_j	=	Jacobian Matrix
n	=	Time Step n
TVD	=	Total Variation Diminishing
CPU	=	Computer Processing Unit
CFL	=	Courant-Friedrichs-Lewy Number
M	=	Mach Number
P_f	=	Final Pressure at Exit
P_i	=	Initial Pressure at Inlet
DIS2	=	Artificial Dissipation Parameter
P_{stag}	=	Stagnation Pressure (N/m^2)
$e^k_{j+1/2}$	=	k^{th} Right Eigenvector of Jacobian Matrix
$\alpha^k_{j+1/2}$	=	Magnitude of Jump in Flow Quantity Across $e^k_{j+1/2}$

Abstract

The computational fluid dynamics code, PARC3D, is tested to see if its use of non-physical artificial dissipation affects the accuracy of its results. This is accomplished by simulating a shock-laminar boundary layer interaction and several hypersonic flight conditions of the Pegasus™ launch vehicle using full artificial dissipation, low artificial dissipation, and the Engquist filter. Before the filter is applied to the PARC3D code, it is validated in one-dimensional and two-dimensional form in a MacCormack scheme against the Riemann and convergent duct problem. For this explicit scheme, the filter shows great improvements in accuracy and computational time as opposed to the nonfiltered solutions. However, for the implicit PARC3D code it is found that the best estimate of the Pegasus experimental heat fluxes and surface pressures is the simulation utilizing low artificial dissipation and no filter. The filter does improve accuracy over the artificially dissipative case but at a computational expense greater than that achieved by the low artificial dissipation case which has no computational time penalty and shows better results. For the shock-boundary layer simulation, the filter does will in terms of accuracy for a strong impingement shock but not as well for weaker shock strengths. Furthermore, for the latter problem the filter reduces the required computational time to convergence by 18.7%.

Chapter I

Introduction

Hypersonic flow is a fast growing area of interest since some modern vehicles may fly in this regime. However, it is a very difficult task to design vehicles to withstand the large thermal and structural stresses which accompany hypersonic flow. Previous techniques involved analytical tools and wind tunnels. The latter, however, present obstacles in the form of wind tunnel physical limitations and expensive, time-consuming experiments. The former does not do any better because most analytical solutions to hypersonic flow problems are usually limited to simplified geometries unlike those that are encountered on actual flight vehicles. A possible solution to this problem is to use Computational Fluid Dynamics (CFD). With CFD, the Navier-Stokes equations can be solved without as many geometrical and physical limitations.

An important goal in all projects involving CFD is to minimize the amount of computational time necessary to complete the numerical simulation. This is because, although manpower is not required during the computations, computer use is still expensive. This is especially true for CRAY supercomputers because of their high cost, which at one point was on the order of a thousand dollars per hour. In any case, one would rather have the problem solved in one hour than in thirty hours.

Minimizing computational time is not the only goal. Accuracy must also be

taken into account. In CFD, accuracy and time minimization go hand in hand because the primary CFD equations--the Navier-Stokes (N-S) equations--are quite complicated. Solving the N-S equations has resulted in many different computer codes each simplifying the N-S equations in some manner to reduce the complexity of their respective solving algorithms. Some assume that the tangential gradients of velocity are small in comparison to the normal gradients and hence arrive at the thin-layer Navier-Stokes codes or Parabolic Navier-Stokes codes. Others assume an inviscid problem and arrive at Euler equation solvers. All of these assumptions lead to questions about the accuracy of the numerical solutions. Do the assumptions result in discrepancies? If so, how much? Should another approach or code be used? All of these questions are valid and need to be addressed. There are several ways to do this. One is to apply the code to a problem with a known analytic solution. These are usually few and, if they exist, are for geometries so oversimplified that they become too trivial a problem for the numerical scheme. In any case, these analytical solutions also require some assumptions. Thus, a better test is to compare the numerical results against experimental values. This latter method is more apt to provide challenging geometries than those available for analytical solutions and also less assumptions. In this way the answers to the questions can be found and the CFD approach to vehicle design or problem solving can be validated.

The primary code used in this work is PARC3D which, like most of its counterparts, has made changes to the N-S equations to make numerically solving them

more computationally efficient. More will be said about the simplifications in the upcoming sections. PARC3D utilizes a central-differencing scheme and hence requires artificial dissipation to maintain numerical stability. However, since this artificial dissipation, or pseudo-viscosity, is non-physical, it can be argued that it will erroneously affect the resulting solutions of the code. One way to verify the importance of artificial dissipation is to simply remove it entirely from the code. However, this is easier said than done, since, as previously stated, it is important in maintaining computational stability. On the other hand, if it could be replaced by a filter that only removes the numerical fluctuations, stability could be maintained with zero dissipation. The filter that is being considered here is the nonlinear Engquist filter (Engquist et al., 1989).

The experiments used to test PARC3D are taken from flights of the Pegasus™ launch vehicle. The Pegasus is a three-stage launch vehicle designed to carry small payloads to low earth orbit. It is unique for several reasons. One is that it was designed to fit under the wing of an existing B-52 aircraft and to be launched like a missile. Another is that the vehicle itself was designed without the use of wind tunnels. CFD was used to determine the thermal and structural loads that the vehicle was to experience (Mendenhall et al., 1990).

There are five flight conditions chosen for simulation in accordance with NASA Dryden. Two are from the first flight while the other three are from the second flight. NASA Dryden fitted the Pegasus™ with HRSI (High-temperature Reusable Surface

Insulation) plugs which measured surface temperatures from which surface heat fluxes were derived. This is a difficult test for the CFD code since the simulation is for a very large vehicle (approximately 15.32 meters long) while the heat fluxes to be calculated depend primarily on events occurring in the thin boundary layer (on the order of millimeters). Thus, a code coming close to the experimental values in a reasonable amount of time would be an accomplishment. In spite of these difficulties, CFD will be seen to be a more feasible option to expensive and time-consuming wind tunnel experiments.

On the second flight, four of the ten HRSI plugs were fitted with pressure ports. This enabled a more direct comparison with computations since there is no intermediary step to question in the comparisons. However, the pressure data are sparse and hence do not allow a strong conclusion to be made about the accuracy of the filter and PARC3D.

It must be noted that, even with the filter, the author is unable to zero the artificial dissipation parameter in PARC3D. Thus, the parameter is lowered to its lowest possible value while still maintaining stability. It is later learned that the simulations remain stable at the lowered artificial dissipation values even without the filter. Thus, the simulations are repeated with the low value and without the filter to further investigate the effects of artificial dissipation and the filter.

In total, there are four sets of data for each Pegasus flight condition. One set is for a simulation which utilizes an unrecommended large amount of artificial

dissipation. Another is for that which uses the Engquist filter. The third is for the simulation without the Engquist filter but at a lower artificial dissipation value. The last set is the flight data itself. Although the filter helps in discerning the shock, its use is more of a hindrance than a blessing. For this particular problem, it is concluded that the lower artificial dissipation without the filter is the best approach. Furthermore, it is shown that the filter is more suited to problems in which the grids are uniform and the shock more normal as opposed to oblique.

In addition to the above scenarios, the filter is utilized in a MacCormack scheme and then applied to the Riemann and converging duct problem. Furthermore, the two-dimensional shock impingement problem is simulated using PARC3D with the filter implemented. The latter is a laminar case with varying shock strengths. Experimental skin friction data is the primary information compared. This data is less likely to have measurement errors than the heat fluxes in the Pegasus data and so should be a more reliable indicator of the effects of the Engquist filter and the value of the artificial dissipation parameter in PARC3D. In the converging duct problem, the ability of the filter to capture the shock as it becomes more and more normal is investigated. The filtered results improve as the shock nears a normal orientation (aligned to one primary coordinate direction).

Chapter II

PARC3D

PARC3D is a fully implicit three-dimensional code primarily used in steady-state, internal flow problems. It is, however, also applicable to external flow problems due to its robustness. In fact, it has been used many times in the past to simulate flows over aerodynamic vehicles. PARC3D is a finite-difference code which solves the three-dimensional Navier-Stokes equations in delta conservation form expressed in general curvilinear coordinates. The solution scheme used in PARC3D is the Beam and Warming approximate factorization algorithm with implicit Euler time-stepping. It has the ability to simulate laminar and turbulent inviscid or viscous flows. For turbulent flow, the Navier-Stokes equations are treated as being Reynolds-averaged (mass-averaged) with the turbulent viscosity generated from the Baldwin-Lomax algebraic turbulence model. Some of the assumptions found in PARC3D are the use of a calorically perfect gas, constant Prandtl number, the Sutherland viscosity law, and Stokes hypothesis. PARC3D is very user-friendly allowing easy specifications of the boundary conditions and other problem parameters. This is done in one subroutine where one inputs a numerical value that describes the problem. The manual lists and describes all of the available types of boundary conditions and necessary inputs.

Given that a general code may not fulfill all of the requirements of a particular problem, it should be noted that there are two shortcomings of PARC3D that impact the Pegasus analysis. One is the lack of an ablative model. The other is that PARC3D only supports two types of surface boundary conditions for viscous problems; 1) no-slip, isothermal wall and 2) no-slip, adiabatic wall. To obtain a more accurate heat flux calculation at the surface, the surface should be allowed to have a nonuniform surface temperature.

PARC3D: Theory

In PARC3D, the Navier-Stokes equations are given in conservation form.

$$\frac{\partial Q}{\partial t} + \frac{\partial F_j}{\partial X_j} = \frac{1}{Re}\frac{\partial G_j}{\partial X_j}$$

Here, Q is the vector of conservation variables, F_j is the inviscid flux vectors, and G_j is the viscous flux vectors.

$$Q = \begin{bmatrix} \rho \\ \rho u_i \\ E \end{bmatrix}$$

$$F_j = \begin{bmatrix} \rho u_j \\ \rho u_i u_j + P\delta_{ij} \\ (E+P)u_j \end{bmatrix}$$

$$G_j = \begin{bmatrix} 0 \\ \tau_{ij} \\ u_k \tau_{jk} - q_j \end{bmatrix}$$

$$\tau_{ij} = \mu\left(\frac{\partial u_i}{\partial X_j} + \frac{\partial u_j}{\partial X_i}\right) + \lambda \frac{\partial u_k}{\partial X_k}\delta_{ij}$$

$$q_j = -\frac{K}{\beta_r Pr}\frac{\partial T}{\partial X_j}$$

The variables E and δ_{ij} represent the total energy per unit volume and Kronecker delta, respectively, and are defined:

$$E = \rho(e + \tfrac{1}{2}u_k u_k)$$

$$\delta_{ij} = \begin{matrix} 1, & if\ i=j \\ 0, & if\ i \neq j \end{matrix}$$

Throughout the following sections and also above, the following notation will be used:

$$\begin{matrix} X_1 = X & u_1 = u \\ X_2 = Y & u_2 = v \\ X_3 = Z & u_3 = w \end{matrix}$$

$$u_x = \frac{\partial u}{\partial X} \quad u_y = \frac{\partial u}{\partial Y} \quad u_z = \frac{\partial u}{\partial Z}$$

$$\begin{matrix} \xi_1 = \xi & U_1 = U \\ \xi_2 = \eta & U_2 = V \\ \xi_3 = \zeta & U_3 = W \end{matrix}$$

The capitalized velocities, U_j, are called the contravarient velocities and are defined as follows:

$$U_j = \frac{\partial \xi_j}{\partial t} + u_k \frac{\partial \xi_j}{\partial X_k}$$

As stated earlier, PARC3D assumes a calorically perfect gas. Hence, after nondimensionalization, the equations of state are:

$$P = \frac{\rho T}{\gamma} = (\gamma - 1)(E - \frac{1}{2}\rho u_k u_k)$$

$$e = \frac{T}{\gamma(\gamma - 1)}$$

$$T = \gamma(\gamma - 1)(\frac{E}{\rho} - \frac{1}{2}u_k u_k)$$

Further assumptions in PARC3D lead to a constant ratio of specific heats (γ) which leads to a constant β_r ($\beta_r = \gamma - 1$), constant Prandtl number, and the Sutherland viscosity law:

$$K = \mu$$

$$\mu = \frac{T^{\frac{3}{2}}(1 + T_s)}{(T + T_s)}$$

$$T_s = \textit{Nondimensional Sutherland Temperature}$$

To be more widely and easily applicable to flows with irregular grids, PARC3D utilizes a general curvilinear coordinate transformation while maintaining the strong conservation law form of the Navier-Stokes equations. The coordinate transformation is onto a computational space in which

$$\xi_j = \xi_j(X_i, t)$$

However, for this particular code, it is assumed that the grid is not adaptive. In other words, the grid does not move or change with time. Therefore, the above dependence on time is removed.

After transformation, the Navier-Stokes equations become (with **J** representing the Jacobian, the set of first partial derivatives of the curvilinear coordinates with respect to the Cartesian coordinates):

$$\frac{\partial \hat{Q}}{\partial t} + \frac{\partial \hat{F}_j}{\partial \xi_j} = \frac{1}{Re}\frac{\partial \hat{G}_j}{\partial \xi_j}$$

$$\hat{Q} = \frac{1}{J}Q$$

$$\hat{F}_j = \frac{1}{J}\left(\frac{\partial \xi_j}{\partial t}Q + \frac{\partial \xi_j}{\partial X_k}F_k\right)$$

$$\hat{G}_j = \frac{1}{J}\frac{\partial \xi_j}{\partial X_k}G_k$$

The primary solver used by PARC3D is the Beam and Warming algorithm which utilizes an approximate factorization technique to form an ADI (alternate direction implicit) type of scheme. The scheme is implicit and computationally robust. PARC3D uses Euler backward differencing (and so is first-order in time) such that

$$\Delta \hat{Q}^n + \Delta t^n\left(\frac{\partial \hat{F}_j^{n+1}}{\partial \xi_j} - \frac{1}{Re}\frac{\partial \hat{G}_j^{n+1}}{\partial \xi_j}\right) = 0$$

$$\hat{Q}^n = \hat{Q}(\xi_j, t^n)$$

$$\Delta \hat{Q}^n = \hat{Q}^{n+1} - \hat{Q}^n$$

Furthermore, to make the code more computationally efficient, the inviscid flux vectors are time-linearized while the viscous flux vectors are time lagged.

$$\hat{F}_j^{n+1} \approx \hat{F}^n + A_j^n \Delta \hat{Q}^n$$

$$A_j = \frac{\partial \hat{F}_j}{\partial \hat{Q}}$$

A_j is called the Jacobian matrix. Thus, the Navier-Stokes equations become ("I" denotes the identity matrix):

$$\left(I+\Delta t\frac{\partial}{\partial\xi_j}A_j^n\right)\Delta\hat{Q}^n = -\Delta t\left(\frac{\partial\hat{F}_j^n}{\partial\xi_j}-\frac{1}{Re}\frac{\partial\hat{G}_j^n}{\partial\xi_j}\right)$$

To make solving the above equation more computationally inexpensive, PARC3D approximately factors the operators above such that a series of inexpensive scalar matrix equations are solved.

$$\left(I+\Delta t\frac{\partial}{\partial\xi}A_1^n\right)\left(I+\Delta t\frac{\partial}{\partial\eta}A_2^n\right)\left(I+\Delta t\frac{\partial}{\partial\zeta}A_3^n\right)\Delta\hat{Q}^n = -\Delta t\left(\frac{\partial\hat{F}_j^n}{\partial\xi_j}-\frac{1}{Re}\frac{\partial\hat{G}_j^n}{\partial\xi_j}\right)$$

Thus, after substituting the appropriate discretizations for the derivatives, the resulting algorithm follows:

$$RHS = -\Delta t\left[\left(\delta_\xi\hat{F}_1^n+\delta_\eta\hat{F}_2^n+\delta_\zeta\hat{F}_3^n\right)-\frac{1}{Re}\left(d_\xi\hat{G}_1^n+d_\eta\hat{G}_2^n+d_\zeta\hat{G}_3^n\right)\right]$$

$$\left(I+\Delta t\delta_\xi A_1^n\right)\Delta\hat{Q}^{**} = RHS$$

$$\left(I+\Delta t\delta_\eta A_2^n\right)\Delta\hat{Q}^{*} = \Delta\hat{Q}**$$

$$\left(I+\Delta t\delta_\zeta A_3^n\right)\Delta\hat{Q}^n = \Delta\hat{Q}^{*}$$

$$\hat{Q}^{n+1} = \hat{Q}^n+\Delta\hat{Q}^n$$

The central-difference operators are defined, for example, as follows:

$$\delta_\xi \hat{F}_1 = \frac{1}{2}\left[\hat{F}_1(\xi+1,\eta,\zeta) - \hat{F}_1(\xi-1,\eta,\zeta)\right]$$

$$d_\xi \hat{G}_1 = \hat{G}_1(\xi+\tfrac{1}{2},\eta,\zeta) - \hat{G}_1(\xi-\tfrac{1}{2},\eta,\zeta)$$

To reduce computational time further, PARC3D uncouples the equations by diagonalizing them in the following manner.

$$A_j = T_j \Lambda_j T_j^{-1}$$

$$\left(I + \Delta t \frac{\partial A_j}{\partial \xi_j}\right) = \left(T_j T_j^{-1} + \Delta t \frac{\partial}{\partial \xi_j} T_j \Lambda_j T_j^{-1}\right) \approx T_j\left(I + \Delta t \frac{\partial}{\partial \xi_j}\Lambda_j\right)T_j^{-1}$$

$$T_1\left(I + \Delta t \frac{\partial}{\partial \xi}\Lambda_1\right)N_{12}\left(I + \Delta t \frac{\partial}{\partial \eta}\Lambda_2\right)N_{23}\left(I + \Delta t \frac{\partial}{\partial \zeta}\Lambda_3\right)T_3^{-1}\Delta\hat{Q} = RHS$$

$$N_{12} = T_1^{-1} T_2$$

$$N_{23} = T_2^{-1} T_3$$

The eigenvalues of the Jacobian matrix are utilized as a diagonal matrix in the above equations.

$$\Lambda_j = Diag\ [U_j, U_j, U_j, U_j + a|K_i^j|, U_j - a|K_i^j|]$$

$$K_i^j = \frac{\partial \xi_j}{\partial X_i}$$

$$|K_i^j| = \sqrt{\sum_i K_i^j K_i^j}$$

Thus, to advance one time-step, the following steps are followed:

$$T_1 \Delta\hat{Q}^{(6)} = RHS$$

$$[I + \Delta t \delta_\xi \Lambda_1] \Delta\hat{Q}^{(5)} = \Delta\hat{Q}^{(6)}$$

$$N_{12} \Delta\hat{Q}^{(4)} = \Delta\hat{Q}^{(4)}$$

$$[I + \Delta t \delta_\eta \Lambda_2] \Delta\hat{Q}^{(3)} = \Delta\hat{Q}^{(4)}$$

$$N_{23} \Delta\hat{Q}^{(2)} = \Delta\hat{Q}^{(3)}$$

$$[I + \Delta t \delta_\zeta \Lambda_3] \Delta\hat{Q}^{(1)} = \Delta\hat{Q}^{(2)}$$

$$T_3^{-1} \Delta\hat{Q} = \Delta\hat{Q}^{(1)}$$

$$\hat{Q}^{n+1} = \hat{Q}^n + \Delta\hat{Q}$$

PARC3D: Artificial Dissipation

Due to the central-differencing used in PARC3D, spurious oscillations occur that if left untreated would grow in time until the code goes unconditionally unstable. Thus, a method of damping these oscillations had to be found. It is for this reason that PARC3D incorporates an artificial dissipation algorithm. This algorithm is comprised of second-order artificial dissipation for the numerical spikes as well as fourth-order artificial dissipation for improved shock-capturing. The latter dissipation is physically-acceptable in that being fourth-order will not drastically affect the accuracy of the second-order solution. However, the second-order dissipation is questionable since it is of the order of the discretization method. Thus, it is plausible that this artificial dissipation may damp out physical discontinuities thereby diffusing the results. Thus, instead of the sharp discontinuity that symbolizes a shock, one captures (if it is still discernible) a smooth gradient. The shock would now have a larger bandwidth (i.e. occupy more node spaces) than if the artificial dissipation was absent.

The artificial dissipation in PARC3D is similar to the Jameson artificial viscosity (Jameson et al., 1981). The PARC3D version is like the following.

$$\begin{gathered}\nabla_\xi[C_\xi(\epsilon^{(2)}\Delta_\xi - \epsilon^{(4)}\Delta_\xi\nabla_\xi\Delta_\xi)](J\hat{Q}) + \\ \nabla_\eta[C_\eta(\epsilon^{(2)}\Delta_\eta - \epsilon^{(4)}\Delta_\eta\nabla_\eta)](J\hat{Q}) + \\ \nabla_\zeta[C_\zeta(\epsilon^{(2)}\Delta_\zeta - \epsilon^{(4)}\Delta_\zeta\nabla_\zeta\Delta_\zeta)](J\hat{Q})\end{gathered}$$

In the above equation, there are forward and backward difference operators defined as:

$$\Delta_\xi Q = Q(\xi+1,\eta,\zeta) - Q(\xi,\eta,\zeta)$$
$$\nabla_\xi Q = Q(\xi,\eta,\zeta) - Q(\xi-1,\eta,\zeta)$$

The other coordinates are defined similarly. The other nonlinear coefficients are defined by, for example,

$$C_\xi = C(\xi+1,\eta,\zeta) + C(\xi,\eta,\zeta)$$
$$C = (|U| + a\sqrt{\xi_x^2+\xi_y^2+\xi_z^2}) + (|V| + a\sqrt{\eta_x^2+\eta_y^2+\eta_z^2}) + (|W| + a\sqrt{\zeta_x^2+\zeta_y^2+\zeta_z^2})$$

The second-and fourth-order parameters are then defined by

$$\epsilon^{(2)} = K_2 \Delta t f$$
$$\epsilon^{(4)} = Max(0, K_4 \Delta t - \epsilon^{(2)})$$
$$where\ K_2 \le 0.25 \qquad 0 < K_4 \le 0.64$$

$$f = Max(f_\xi, f_\eta, f_\zeta)$$
$$f_\xi = \frac{|P(\xi+1,\eta,\zeta) - 2P(\xi,\eta,\zeta) + P(\xi-1,\eta,\zeta)|}{|P(\xi+1,\eta,\zeta) + 2P(\xi,\eta,\zeta) + P(\xi-1,\eta,\zeta)|}$$

"f" is called the "switch" function and is smoothed (averaged) over immediate neighbor points. The artificial viscosity is then embedded into the solver steps as follows:

$$T_1 \Delta\hat{Q}^{(6)} = RHS$$

$$[I(1+IV_1) + \Delta t \delta_\xi \Lambda_1] \Delta\hat{Q}^{(5)} = \Delta\hat{Q}^{(6)}$$

$$N_{12} \Delta\hat{Q}^{(4)} = \Delta\hat{Q}^{(4)}$$

$$[I(1+IV_2) + \Delta t \delta_\eta \Lambda_2] \Delta\hat{Q}^{(3)} = \Delta\hat{Q}^{(4)}$$

$$N_{23} \Delta\hat{Q}^{(2)} = \Delta\hat{Q}^{(3)}$$

$$[I(1+IV_3) + \Delta t \delta_\zeta \Lambda_3] \Delta\hat{Q}^{(1)} = \Delta\hat{Q}^{(2)}$$

$$T_3^{-1} \Delta\hat{Q} = \Delta\hat{Q}^{(1)}$$

As can be seen, IV_i is the implicit artificial viscosity operator given by

$$IV_1 = \nabla_\xi [C_\xi (\epsilon^{(2)} \Delta_\xi - \epsilon^{(4)} \Delta_\xi \nabla_\xi \Delta_\xi)] J$$

Thus, the equations become a set of pentadiagonal block equations, which can be more efficiently solved using a version of Gaussian elimination than if the original Navier-Stokes equations were iterated until they were converged.

Chapter III

Engquist Filter

The Engquist filter (Engquist et al., 1989) was developed to control numerical oscillations generated by central-differencing. Central-differencing is a very desirable discretization technique because it is second-order accurate and easy to use. However, it has difficulties in terms of stability. Spurious oscillations usually result around discontinuities if central-differencing is used due to the inability of adjacent nodes to communicate between themselves. These oscillations grow in time until the solutions diverge to the extent that one calculates negative pressures, densities, etc.. To take advantage of the higher order of accuracy of central-differencing, one must first deal with its stability problems. The Engquist filter does just this. It is perfectly suited to removing spurious oscillations resulting from central-differencing. Thus, stability is maintained and the benefits of central-differencing are gained.

Other more stable methods have been derived. The most popular ones today are those called Essentially-Non-Oscillating (ENO). These schemes are usually of higher order but also require a considerable amount of computational time because eigenvectors must be calculated for the entire flowfield in contrast to the Engquist filter which calculate eigenvectors only at extrema. However, these schemes do not develop the spurious oscillations that plague a number of finite-difference schemes.

The scalar version of the Engquist filter works in a very simplistic manner. There are also many different versions of it (Engquist et al., 1989). The easiest to extend to a system of equations is Algorithm 2.1. This algorithm is simple in theory and can be applied either as a postprocessor or within the solving algorithm. The former means that the filter is applied to the solution after the solver has finished its iteration while the latter implies that the filter is embedded within the solver. Thus, after the solver iterates at a given node point, the filter is called to see whether or not the node point needs to be filtered.

The scalar Engquist filter works in the following manner. It finds an extremum, say at node j, and adjusts this extremum such that the extremum is removed or, at the least, decreased. To maintain conservation, the amount of decrease (or increase) is then added to (or subtracted from) the adjacent node point with the larger gradient (in Figure 1, this is node j+1). The change in the adjacent node is such that the adjacent node does not become another extremum.

The first step in the filtering process is to see if the following inequality is satisfied:

$$\begin{aligned} \Delta_+ u_j &= u_{j+1} - u_j \\ \Delta_- u_j &= u_j - u_{j-1} \\ (\Delta_+ u_j)(\Delta_- u_j) &< 0 \end{aligned}$$

If it is not, then the next node point is evaluated. However, if it is, then an extremum has been found at node j. The next step is to determine in which direction the larger gradient faces. If Δ_+u_j is the larger of the two gradients, then the node point j+1 is deemed the adjacent node to be affected. The amount added to (subtracted from) node j is also subtracted from (added to) node j+1 (Figure 1). In this way global conservation of the flow property u is maintained. Similarly, if the larger gradient is that for Δ_-u_j, then the adjacent node point that will be corrected is that of j-1. The amount of correction that is applied to node j is such that the corrected node point will not become another extremum. To ensure this, the Engquist filter takes and compares the two neighboring gradients. The larger of the two is then halved and compared to the smaller one. Between the latter two values, the smaller is taken to be the amount of acceptable change that can be applied to node point j and the corresponding adjacent node. Once the extremum at node j is dealt with, the filter continues down the line to find other extrema and repeat the entire process.

The above algorithm does have its limitations. For one, it is not total-variation-diminishing (TVD). Thus, the total variation may increase which implies the probability of the presence of oscillations near a shock. According to Engquist, these oscillations normally have very small amplitudes. From the author's numerical experiments, the oscillations have been observed but, as Engquist has noted, have been of small magnitude. Another drawback of Algorithm 2.1 is that it cannot detect an extremum composed of more than one node point. Furthermore, in smooth regions,

Engquist Filter Applied at Node j

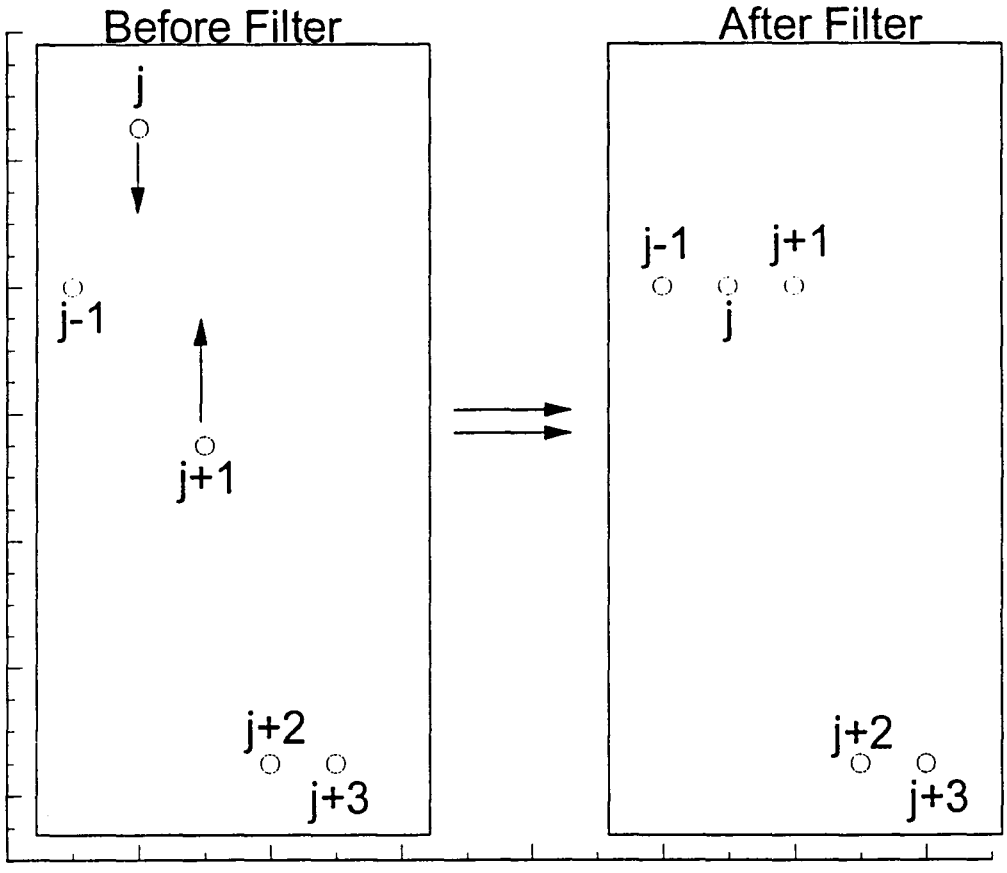


Figure 1

it flattens extrema that are not the result of overshooting and so results in a lower order of accuracy in the smooth regions even though the primary solver has a higher order. Engquist resolves these shortcomings with other algorithms which he discusses in depth in his paper. However, for the present problem, Algorithm 2.1 suffices since it is the most feasible algorithm for expansion to a system of equations. It must be noted that, although it may appear that Algorithm 2.1 is insufficient as a filter, it does its job well, as shown by Engquist.

The Engquist filter as applied to a system of equations is similar to Algorithm 2.1. The primary difference is that in the scalar version only the magnitude is important whereas in the system case the magnitude and direction are both integral factors. Other than this, the scalar and system forms are the same in theory. In the paper by Engquist, the filter for a system of equations is described by Algorithm 4.1.

The form of the equation that the filter is applied to is the following one-dimensional wave equation:

$$u_t + f(u)_x = 0$$

Here the subscripts imply differentiation with respect to the variable in the subscript. For example u_t implies the derivative of u with respect to time. Also, the flux function is written as a function of the conservative variables (u). In this case, the hyperbolic equation can be decomposed to obtain an independent set of equations:

$$f(u)_x = f_u u_x$$
$$A = f_u$$
$$u_t + A u_x = 0$$

The matrix A is the Jacobian matrix which can be decomposed into independent fields by finding its eigenvectors and eigenvalues. It is here that the generalization to systems takes place. First the eigenvectors of the Jacobian matrix are found. Letting m be the number of equations (i.e. m=dimension of problem + 2, such that m=3 describes a one-dimensional problem), u the (m by 1) matrix of conservative variables, $e^k_{j+½}$ the kth right eigenvector of A at the cell face dividing nodes j and j+1, and $\alpha^k_{j+½}$ the magnitude of the jump in associated flow property across the kth eigenvector, Engquist arrives at the following relation for the difference gradients:

$$\Delta_+ u_j = \sum_{k=1}^{m} \alpha^k_{j+\frac{1}{2}} e^k_{j+\frac{1}{2}}$$
$$\Delta_- u_j = \sum_{k=1}^{m} \alpha^k_{j-\frac{1}{2}} e^k_{j+\frac{1}{2}} .$$

In association with the scalar version of the filter, the α's can be viewed as the magnitudes of the gradients while the e's are their directions. The right-hand-side of the above equation is calculated at j+1/2 as opposed to a proper node point such as j or j+1 because Engquist suggests the use of averaging between the flow values at j and j+1. His suggestion leads to a Roe-averaging technique (Roe, 1981) for finding the eigenvector values.

$$u^k_{j+\frac{1}{2}} = \frac{\sqrt{\rho_{j+1}}\,u^k_{j+1} + \sqrt{\rho_j}\,u^k_j}{\sqrt{\rho_{j+1}} + \sqrt{\rho_j}}$$

As with the scalar version of the filter, the magnitudes of the gradients ($\alpha_{j+1/2}$ and $\alpha_{j-1/2}$) are compared to one another. The required changes are then applied to the flow variable at node j and the corresponding adjacent node point in each field.

To maintain conservation, the changes have to be added or subtracted in the same field for all of the node points involved. In other words, the magnitude of change associated with the direction of a particular eigenvector would be added by first multiplying it by its associated eigenvector. This maintains the concept of vectorial addition. In the scalar version, only the corrected scalar amount δ is added to the respective node points. In the system case this becomes:

$$\Delta_+ u_j = \sum_{k=1}^{m} \alpha^k_{j+\frac{1}{2}} e^k_{j+\frac{1}{2}} + \delta\, e^1_{j+\frac{1}{2}}$$

$$\Delta_- u_j = \sum_{k=1}^{m} \alpha^k_{j-\frac{1}{2}} e^k_{j-\frac{1}{2}} - \delta\, e^1_{j+\frac{1}{2}}$$

Note that the magnitude δ added to $\Delta_- u_j$ is multiplied by the first eigenvector $e^1_{j+1/2}$ instead of $e^1_{j-1/2}$ in order to maintain conservation.

After all of the incremental changes are added to the flow variables at node j, the filter moves on to the next node point. There, the test for an extremum is again

applied. If one is found, the filter alters that node point following the procedures described above. If no extremum is found, the filter checks the following node point and so on until the last node point of the present coordinate direction is reached. Thus, the filter does not have to go through the computationally intensive calculation of eigenvectors at each node point.

For multi-dimensional problems, the filter is split by applying the filter successively in each direction. For example, for the two-dimensional Euler equations (x-y space), the problem is treated like two one-dimensional wave equations. The spatial derivative of the first wave equation is taken with respect to y and the second is taken with respect to x. This dimensional-splitting, though, becomes computationally time consuming. It was found by the author that for most cases the amount of CPU increase incurred by applying the filter to each direction is not substantiated by any impressive increase in accuracy. It suffices to apply the filter in the flow direction to which the discontinuity is primarily normal. Also, it was found that the filter is most effective for a one-dimensional problem or for one in which the discontinuity is primarily in one coordinate direction. Fortran listings for the one-, two-, and three-dimensional forms of the Engquist filter used in this thesis are located in Appendix A.

Chapter IV

MacCormack Scheme

The explicit MacCormack scheme is a two-step method which is second-order accurate. Much can be learned about the method by applying it to a scalar problem--the linear wave equation.

$$u_t + cu_x = 0$$
$$c = constant$$

As shown by Anderson et al. (1984), the scheme has a truncation error of the order of $[(\Delta x)^2, (\Delta t)^2]$, and is stable for $|\nu| = 1$ (the Courant-Friedrichs-Lewy or CFL limitation). The modified equation for this method is

$$u_t + cu_x = -c\frac{(\Delta x)^2}{6}(1-\nu^2)u_{xxx} - \frac{c(\Delta x)^3}{8}\nu(1-\nu^2)u_{xxxx} + \ldots$$

The modified equation is the actual equation that the particular differencing method applied solves after substitution of the Taylor series expansion for each numerical discretization in the original equation (eg. $u_{j+1} = u_j + [\Delta x]u_x + O\,[\Delta x^2]u_{xx}$). In this case, the MacCormack method results in an odd-derivative of u with respect to x as the highest order residual (right-hand-side of the above equation). What this implies is that the method is dispersive. In other words, it will have oscillations in its solution. If the highest order was that of an even derivative of u with respect to x, the scheme

would be dissipative. For this scenario, the results would be damped, thereby removing the possibility of spurious oscillations. However, sharp discontinuities would also be damped out such that the bandwidth required to capture a shock is much larger. Thus, the shock becomes diffused over several node points as opposed to a sharp profile as desired.

The MacCormack method is a two-step predictor-corrector scheme. The predictor step utilizes a forward-difference for u_x, while the corrector step uses a backward-difference. For hyperbolic problems with flow speeds greater than Mach one, the predictor step is inherently unstable. This is seen through a characteristic analysis. Information is coming from upstream and not downstream as the forward-differencing implies. Normally this would cause the scheme to go unstable, but the presence of the stable backward-differencing step prevents this. The linear wave equation is discretized in the following manner. The first equation is the predictor step while the second one is the backward-differenced corrector step.

$$u_j^{\overline{n+1}} = u_j^n - c\frac{\Delta t}{\Delta x}(u_{j+1}^n - u_j^n)$$
$$u_j^{n+1} = \frac{1}{2}[u_j^n + u_j^{\overline{n+1}} - c\frac{\Delta t}{\Delta x}(u_j^{\overline{n+1}} - u_{j-1}^{\overline{n+1}})]$$

Anderson et al. (1984) suggests that the differencing can be constantly reversed throughout the computations. However, the advantages of this are minimal and so not warranted at this time.

It was earlier noted that the MacCormack method has a CFL of one. This

limitation is due to the fact that the scheme is explicit as opposed to being implicit. Implicit codes do not have this restriction and are capable, theoretically, of achieving unconditional stability (no CFL limitation). For an explicit scheme, surpassing the CFL condition would almost certainly invite instability. It is in this area that another feature of the filter becomes beneficial.

The CFL number is a relative measure of the amount of spatial and temporal coarseness acceptable for the problem. It is not absolute in that surpassing it means that the problem will definitely go unstable. For the linear wave equation, the following coefficient is readily visible:

$$CFL = c\frac{\Delta t}{\Delta x}$$

For a constant wave speed c, the incremental time step is limited by the grid coarseness. To satisfy the CFL condition, one would either have to decrease the time step or increase the grid spacing. The former results in longer computational times to convergence and, hence, is undesirable. The latter results in a loss of resolution as Δx increases. Recall that the error associated with a numerical solution is usually on the order of some power of Δx. Thus, a compromise must be found.

To better understand the CFL condition one must go to a characteristic analysis. One should envision a rectangular two-dimensional grid with the horizontal axis depicting the x-direction and the vertical axis depicting the t-direction (Figure 2). Imagine a diagonal line as the representation of a disturbance in the x-t plane. If the

x-t Space Characteristic

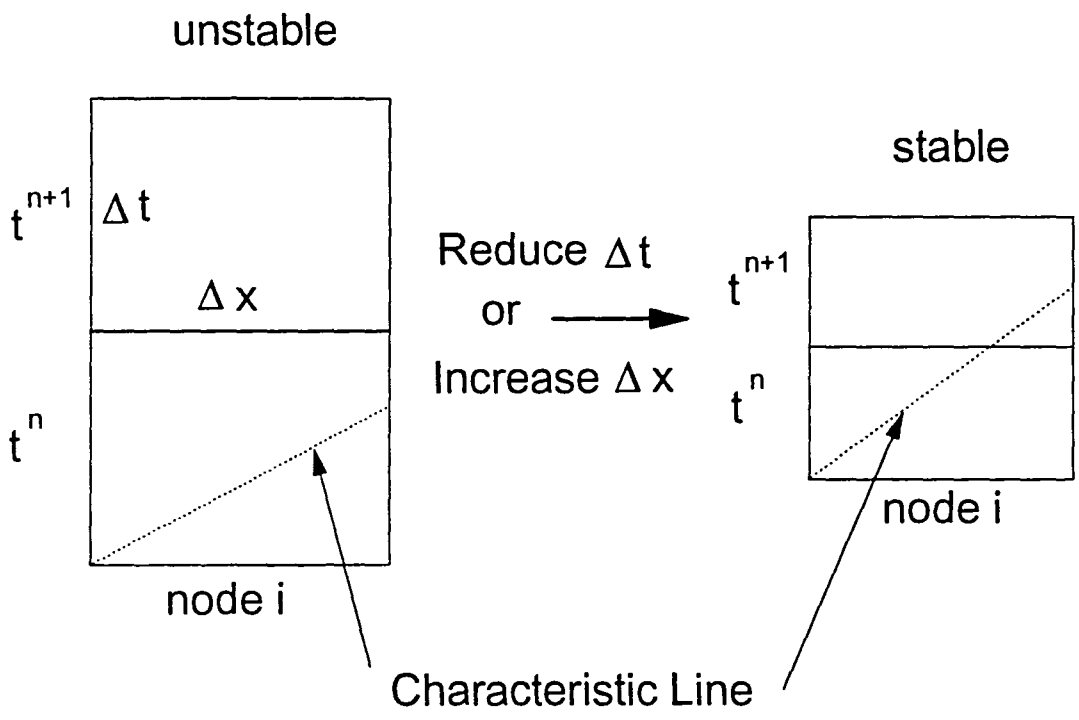


Figure 2

line (the characteristic of the problem) emanating from the t^n cell enters the t^{n+1} cell then information from the lower time level is being passed on to the next time level successfully and the scheme is stable. However, if the characteristic does not enter the t^{n+1} cell, then information is not being transferred to the next level and the scheme is unstable. Thus, either Δx must be increased or Δt must be decreased such that the above CFL criterion is met.

The CFL number dictates the largest time step that one can use for a given grid coarseness. However, it is believed that the Roe-averaging done by the Engquist filter allows larger time steps than that dictated by the CFL limit. This is because information normally found only in node i will now also be in the i-1 and i+1 nodes, thereby somewhat artificially increasing Δx to meet the CFL restriction. In this sense, one can accelerate the convergence in the earlier part of the simulation by using larger time steps with the filter. Overall, the amount of computational time to convergence should decrease with use of the filter due to the use of larger time steps. Thus, by increasing the stability limits, the filter will aid an explicit scheme but not necessarily an implicit one which theoretically has no stability limit.

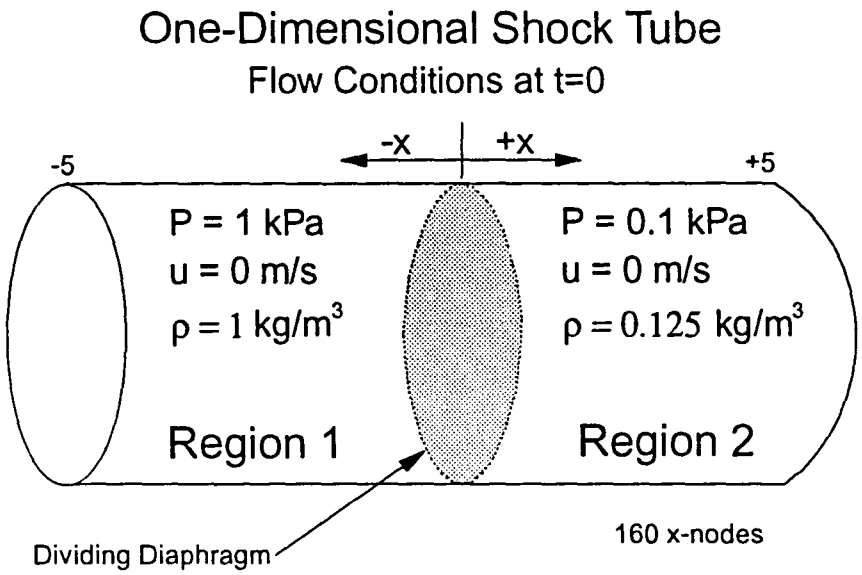
Chapter V

Test Conditions

One-Dimensional Problem:

The filter is first tested by applying it to a one-dimensional shock tube problem more widely known as the Riemann problem (Figure 3). Here a gas is separated into two regions (note that there is no flow variable gradient in the radial direction hence this is a one-dimensional problem). In one region the gas is at a higher pressure and density than in the other. Furthermore, the gas is motionless (i.e. $u = 0$). At time 0 sec, the dividing membrane is instantaneously removed and the gas is allowed to flow. What happens is that a contact discontinuity, a shock, and an expansion wave propagate in opposite directions through the tube (Figure 3). Across the contact discontinuity there is a density jump while the pressure and velocity remain constant. Across the shock, there is an abrupt pressure, density, and velocity jump. As for the expansion wave, there is a smooth density, pressure, and velocity gradient.

For the shock tube problem, the compressible Euler equations is the system investigated. The primary solver is the highly dispersive MacCormack algorithm. For boundary conditions, the tube walls are treated as impermeable slip walls. An


One-Dimensional Shock Tube
Flow Conditions at t=0
-x
+x
-5
+5
P = 1 kPa
u = 0 m/s
ρ = 1 kg/m³
P = 0.1 kPa
u = 0 m/s
ρ = 0.125 kg/m³
Region 1
Region 2
Dividing Diaphragm
160 x-nodes

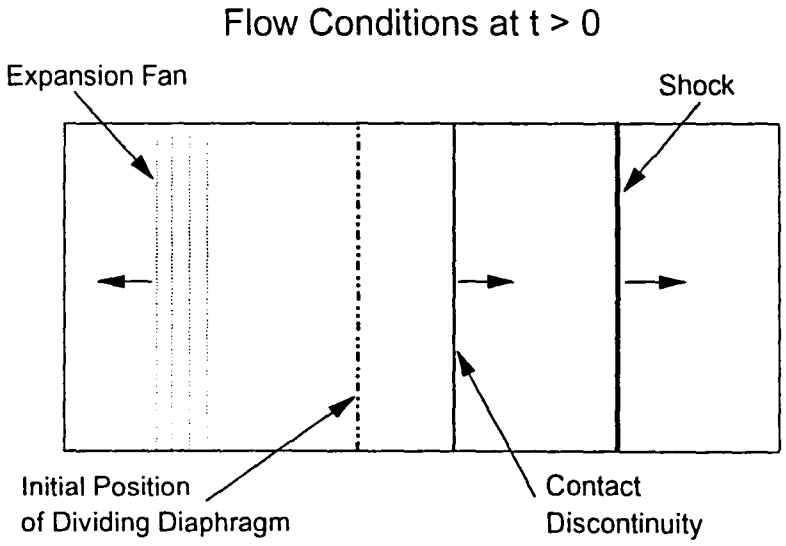

Flow Conditions at t > 0
Expansion Fan
Shock
Initial Position
of Dividing Diaphragm
Contact
Discontinuity

Figure 3

infinitely long shock tube is assumed so that no reflections occur at the tube ends. Each wave front has a velocity (relative to the laboratory). This is a transient problem, and so comparisons of time to convergence between non-filtered and filtered simulations are disregarded. Accuracy is the only question and is investigated by comparison of the numerical results for velocity, pressure, and density to analytical solutions.

The shock tube problem is a simple but challenging initial test case for the Engquist filter. The presence of a contact discontinuity tests the ability of the filter to handle monotonically changing variables. Additionally, the presence of the shock with oscillations tests the ability of the filter to maintain stability against perturbations as well as to discern steep gradients. Furthermore, since this is only a one-dimensional problem, the dimensional-splitting of the filter is presently avoided.

Two-Dimensional Problem:

The first of two two-dimensional problems chosen to test the filter incorporates a converging duct with shock reflection at an impermeable wall and utilizes the MacCormack scheme as the solver. Mach 3.0 compressible inviscid flow (ideal gas) arrives at the duct and hits a 10° compression ramp at the top of the duct forming an oblique shock. The shock travels downwards and reflects off an impermeable lower

wall to once again hit the angled ramp before leaving the duct. In total, the shock reflects twice off the top wall and once off the bottom wall.

The boundary conditions are: 1) at the top wall, an impermeable wall with a slip condition, 2) at the bottom wall, an impermeable wall with a slip condition, 3) at the inlet, the flow parameters are specified since the flow is supersonic, 4) at the exit, a linear extrapolation since the exiting flow is still supersonic. The boundary conditions at the impermeable wall are treated as being reflective such that the fluxes across the wall are zero. The numerical solutions are then compared to the results obtained via oblique shock tables.

The numerous shock reflections and their oblique angles with respect to the primary flow direction test the ability of the filter to accomplish several tasks. One is to see if the filter is able to resolve the shocks over a narrow bandwidth. Another is to see if it does so accurately. The results of the filter as the initial oblique shock is made more and more normal is also investigated. Furthermore, since this problem is being solved to steady-state, the question of whether or not the filter accelerates convergence is determined. Figure 4 shows the geometry and inlet conditions for this problem.

To test the filter in its application to viscous problems, a second two-dimensional problem is proposed. This problem is that of an oblique shock impinging on a laminar boundary layer on an adiabatic flat plate. This is chosen because of the many experimental data that are both trusted and readily available. The code chosen

Convergent Duct Geometry

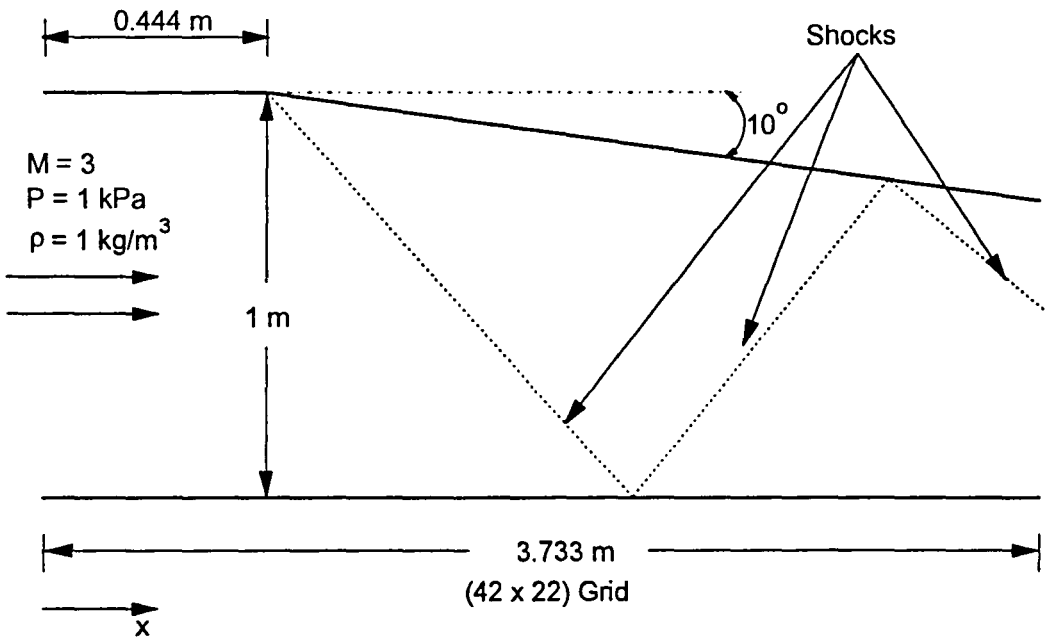


Figure 4

Geometry of Shock-Boundary Layer Interaction

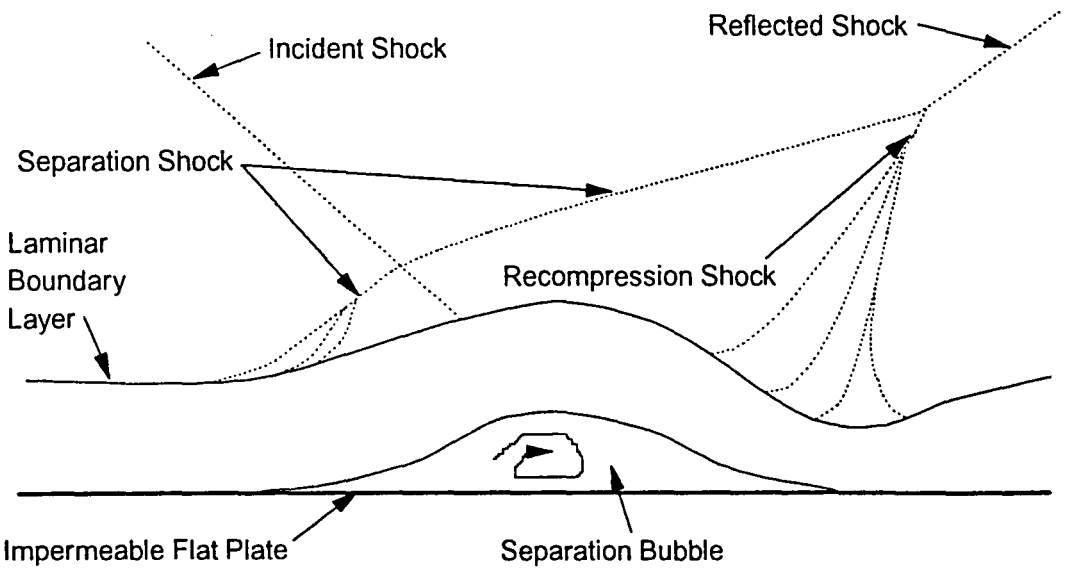


Figure 5

for this problem is the same code to be used in the three-dimensional problem, the PARC3D code. However, since PARC3D is designed for three-dimensional problems, the shock-boundary layer problem is modelled as being three-dimensional. This is done by setting the boundary conditions at the sides of the computational mesh to be an axis of symmetry. The two-dimensional skin-friction calculations are then taken to be the spanwise average of the plate. The incoming shock is set by specifying the farfield boundary condition (on the surface above the plate) to be the Rankine-Hugoniot jump conditions taken from oblique shock tables. The shock reflected from the plate is verified to leave through the exit boundary (perpendicular to the incoming flow) and not through the farfield surface. The incoming flow is set to freestream values and the no-slip adiabatic plate is seen by the flow after six node points. At the seventh node point, a boundary layer forms. The shock is made to impinge at the proper location (as dictated by the reference paper) by varying the vertical height of the farfield surface.

Two shock-boundary layer interactions are simulated. One is with no separation and the other with separation. Figure 5 shows an expanded view of the shock impinging on the laminar boundary layer causing separation. The incident shock causes a flow reversal in the boundary layer due to the adverse pressure gradient it creates such that the boundary layer separates. The point of separation is characterized by the skin friction becoming negative. The bulge induces the separation shock which interacts and is deflected by the incident shock. On the rear face of the bulge, the

sudden expansion causes an expansion fan to form and the flow is bent towards the plate once again. The flow then sees another compression region from which the recompression shock forms. At this location, the boundary layer reattaches and high momentum fluid nears the surface of the plate. The larger momentum results in a thinning of the boundary layer leading to high heating. At the reattachment point, the skin friction value becomes positive once again. The compression shock later merges with the separation shock to form the reflected shock that is familiar to inviscid compressible flow.

The above problem is chosen primarily to test the filter and the PARC3D code against more controlled experimental data. This is because the data for the three-dimensional problem to be discussed next may contain some measurement errors and would not be an absolute indicator of the abilities of the code and the filter. In any case, the shock-boundary layer problem with all of its multiple shock interactions will aid in validating PARC3D and the Engquist filter.

Three-Dimensional Problem:

NASA Dryden has instrumented the Pegasus launch vehicle. Specifically, surface heat flux data from the first two flights of the Pegasus and surface pressure data from the second flight are available. This presents an excellent opportunity for

testing the PARC3D code as well as the Engquist filter under three-dimensional conditions.

The Pegasus vehicle is a three-stage launch vehicle (Figure 6) designed to carry small payloads (approximately 4000 N or less) into low earth orbit. It is 15.24 meters long, has a 1.22 meter diameter cylindrical fuselage with a delta wing (6.71-meter wingspan) and a dry weight of approximately 186,824 N. It is a novel approach to satellite launchings. Earlier systems incorporated rockets launched from the ground. The Pegasus on the other hand is carried under the wing of an aircraft (for the first two flights it was carried by a B-52) and launched like a missile at an elevation of approximately 12,192 meters and an initial Mach number of 0.8. A primary advantage to this configuration is that the lower pressures and densities at the launch level lead to lower magnitudes of aerodynamic heating and drag. Thus, the vehicle does not have to carry as much fuel and is not as heavy as its ground-launched counterpart. The smaller frame leads to the vehicle being more economical and readily accessible for launchings. A primary drawback is that its smaller frame can only accommodate small payloads.

An interesting aspect of the Pegasus design was its lack of any wind-tunnel testing. Instead, the designers relied heavily on computational tools. CFD Codes such as MISL3, Missile DATCOM, SWINT, UPS, TURF, and MADM (Mendenhall et al., 1990) were used to predict the structural and thermal stresses that the vehicle was to experience. Thus far, the successful flights of Pegasus point to CFD as a viable tool

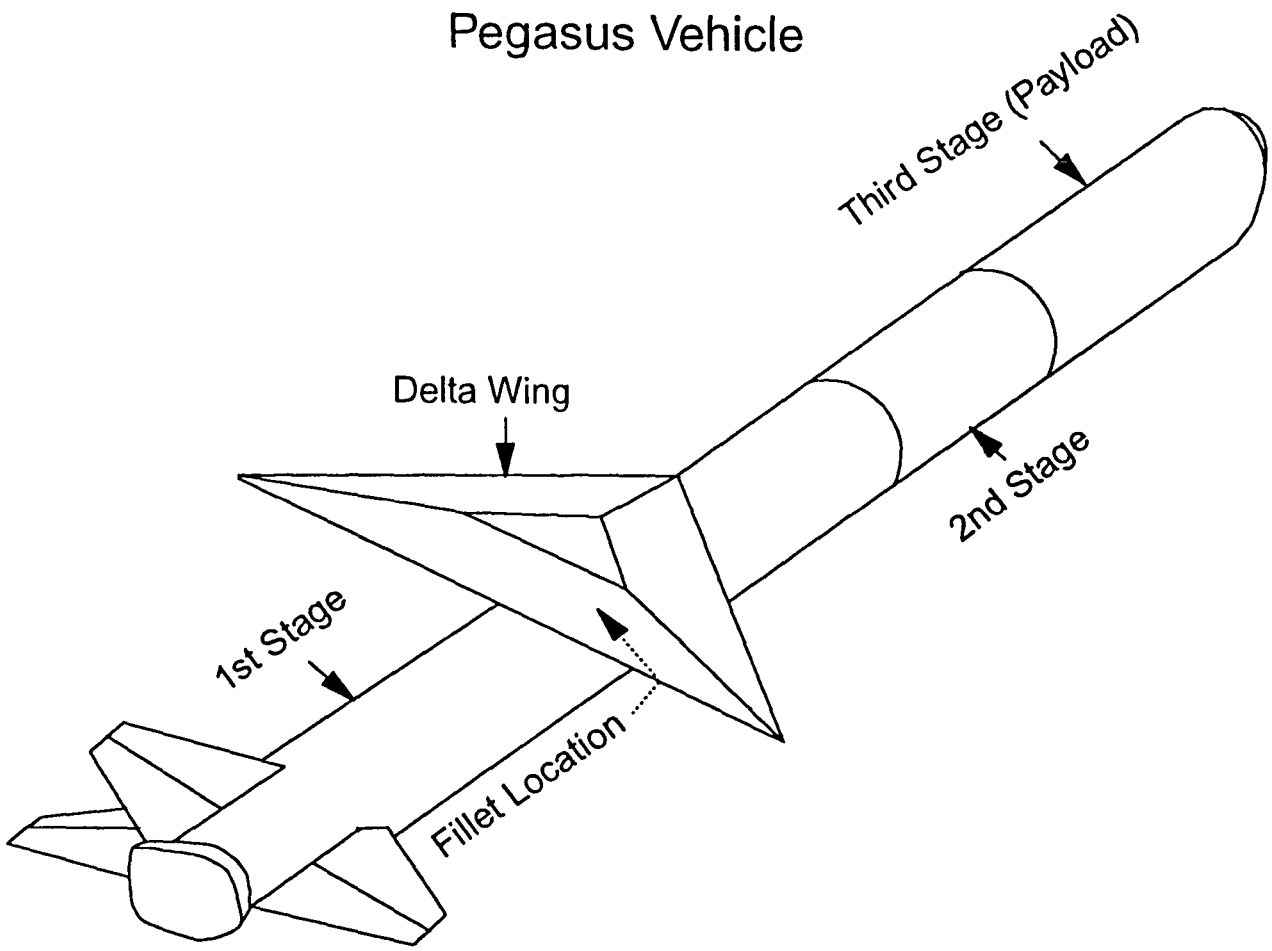

Pegasus Vehicle
Third Stage (Payload)
Delta Wing
2nd Stage
1st Stage
Fillet Location

Figure 6

in future vehicle design.

The first step in the numerical simulation is to generate the Pegasus grid and choose a computational code. For the former, the grid that was used in the initial vehicle design was obtained from Nielsen Engineering And Research (NEAR). This grid consists of two sections (Figure 7). One includes the area from the nose of the vehicle to its wing trailing edge. The other begins from the wing trailing edge to the rest of the vehicle. Since the flowfield is inherently supersonic and the area of interest (the fillet, more will be said about this later) is ahead of the wing trailing edge, it was decided that to save computational time, only the front grid would be used. This grid consists of 389,436 node points (92x83x51). Kuhn (1991) refers to the computational grid used here as the fine mesh. Kuhn finds that the heat fluxes are dependent on grid density in the axial direction (along the body) and that use of a coarser subset of the fine mesh effectively removes this dependence. However, to be more conservative, the fine mesh is used in these simulations.

As for the CFD code, PARC3D is chosen due to its robustness and ease of use. The flight simulation poses a considerable amount of difficulty. Some primary considerations are ablation of the Pegasus surface, dissociation in the flow, complex flow patterns near the junction of the wing and cylindrical body (the fillet), and the unsteady nature of the flight. All of these, if addressed, would result in such long computational times that the simulations would not be feasible. Thus, a few assumptions had to be made. Since the ablation rate is unknown it was decided that

Complete Pegasus Computational Grid

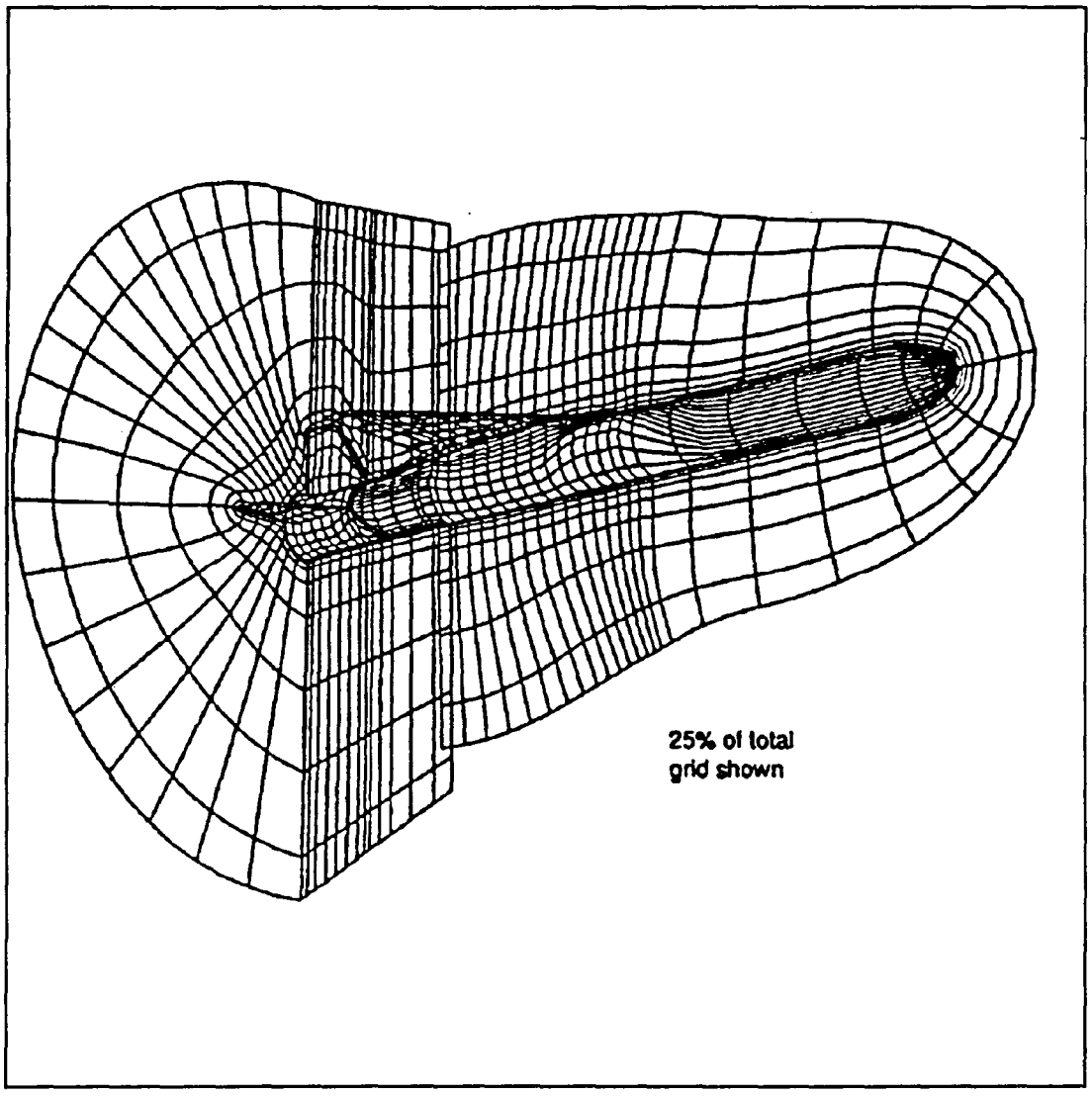


Figure 7

the effect of ablation on the flowfield be ignored for the time being. However, the vehicle skin temperature is kept constant (at the ablation melting temperature) to account for the Pegasus surface thermal decomposition. Furthermore, since airflow is quite adaptive as supported by very small characteristic times, the problem is assumed to be quasi-steady. As for the flow regime, it is assumed to be turbulent.

The location of interest on the Pegasus vehicle is the fillet area (Figure 8). It is there that Noffz et al. (1991) placed High-temperature Reusable Surface Insulation (HRSI) plugs made from shuttle tile material. A thermocouple is placed on each of the plug surfaces wetted by the gas flow. The fillet is instrumented because it was thought that the shock emanating from the wing would intersect it and hence cause high localized heating detrimental to the structural integrity of the vehicle. The surface heat fluxes are obtained in a roundabout manner because the ablating surface could not be instrumented without affecting the flowfield. The surface temperatures that are measured are input into the Lockheed Thermal Analyzer (LTA) program. A one-dimensional model of the plug is assumed due to the small width of the plug as compared to its depth (length). Furthermore, the back wall of the plug is assumed to be adiabatic. This assumption is justified by the interior temperature readings taken from the embedded wall thermocouples. The results have an accuracy of ±22.6% and are quite satisfactory considering the difficult experimental conditions. The large value is mostly due to uncertainties in the thermal properties of the shuttle tile material and not measurement error.

Magnified View of Fillet Area

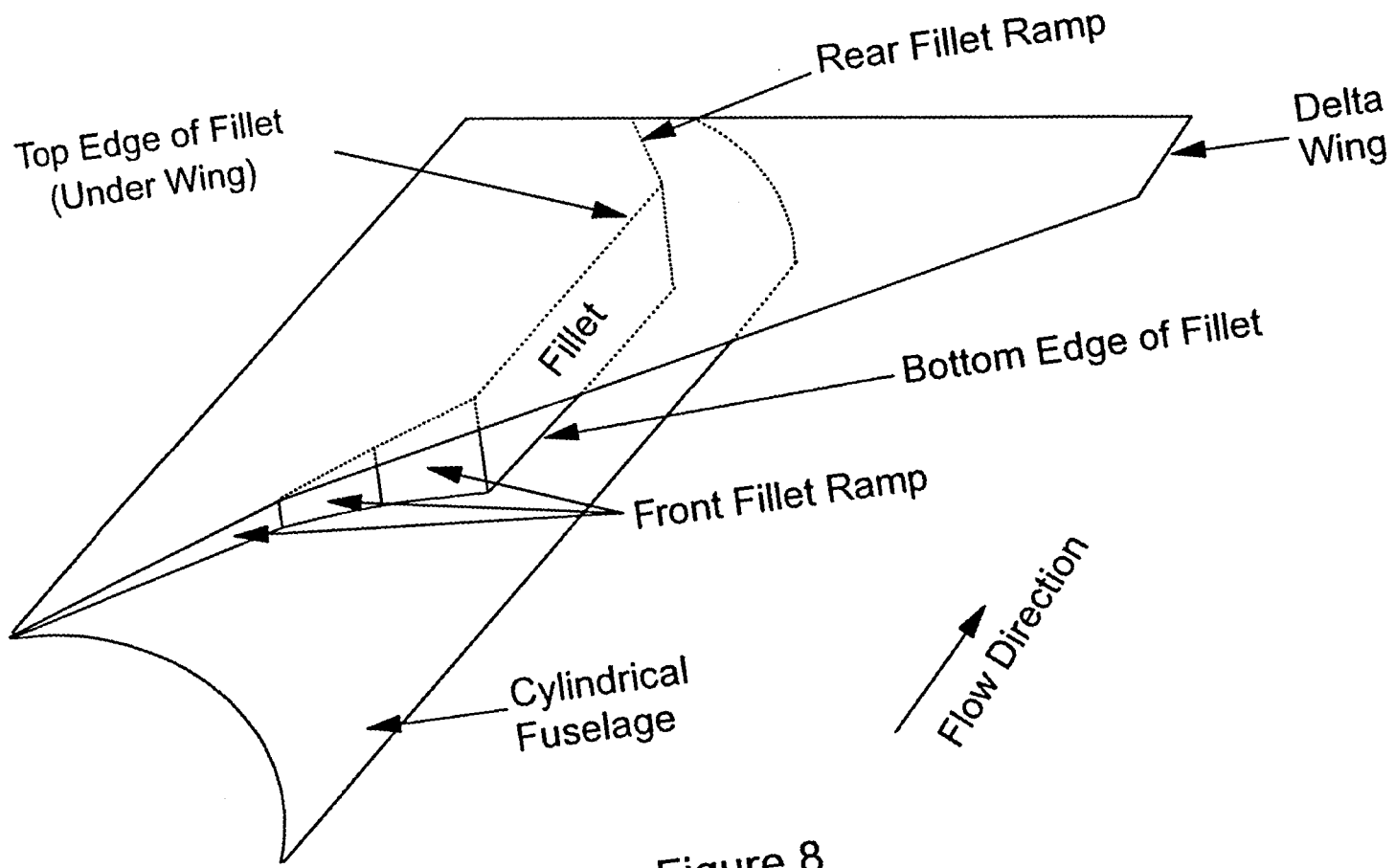


Figure 8

On the second flight of Pegasus, NASA instrumented a few of the HRSI plugs with pressure ports. These pressure readings enable a more direct comparison of CFD results to actual flight data because of the lack of an intermediate step as opposed to the heat flux calculations. The only problem is that at the higher altitudes, the ambient pressure became too low for the devices to measure and so the pressure readings became erroneous. Thus, the only usable pressure data come from the first two flight conditions (M = 3.52 and M = 5.0) of the second vehicle launch, thereby making pressure data sparse.

Five flight positions of the Pegasus launch vehicle are simulated. Two are from the first flight while the other three are from the second flight. The three-dimensional simulations are for:

1)	Flight 001	Mach 3.52	$\alpha = 7.35°$
		Re = 1,559,679	Pref = 3596.27 N/m^2
2)	Flight 001	Mach 6.67	$\alpha = 0.00°$
		Re = 26,447	Pref = 81.68 N/m^2
3)	Flight 002	Mach 3.52	$\alpha = 2.65°$
		Re = 1,595,600	Pref = 3774.38 N/m^2
4)	Flight 002	Mach 5.00	$\alpha = 0.50°$
		Re = 404,400	Pref = 946.59 N/m^2
5)	Flight 002	Mach 6.67	$\alpha = 0.00°$
		Re = 71,410	Pref = 190.37 N/m^2

The ambient pressure and temperature are taken to be the reference quantities and the Reynolds number is based on the diameter of the cylindrical fuselage. The simulations are all started with the flow variables initialized to ambient conditions.

In total there are four sets of data for each flight condition (Figures 18-22). One ("Art Disp") is for a simulation done by Fricker et al. (1992) which utilized a large amount of artificial dissipation. Another ("Fltrd") is for that which uses the nonlinear Engquist filter. The third ("Low Disp") is for the simulation without the Engquist filter but at a lower artificial dissipation value. The last ("Flight") is the experimental data which is taken to be the reference point.

Chapter VI

Results and Discussion

Riemann Problem:

The geometry and initial conditions for this problem are shown in Figure 3. The CFL number is set to 0.7 and the problem is allowed to run from t = 0 sec to approximately t = 0.00233 sec. The MacCormack scheme is truly dispersive as can be seen in Figure 9. Also visible is the fact that the filter is successful in removing the oscillations.

The one aspect of the filter that is highly touted is its ability to resolve discontinuities over a narrower bandwidth than without it. This is clearly the case in the plots in which the shock (the discontinuity at x = 1.3 cm) is captured over four node points with the filter as opposed to seven node points for the simulation without the filter. The filtered solution does well throughout all of the varying discontinuities in comparison to the exact solution. There was some concern on how the filter would do when it encountered a smoothly varying region. The figures, however, show that the filtered solution actually follows the monotonic nature of the exact solution across the expansion fan (farthest left front, located wholly in the -x region). Also, as earlier

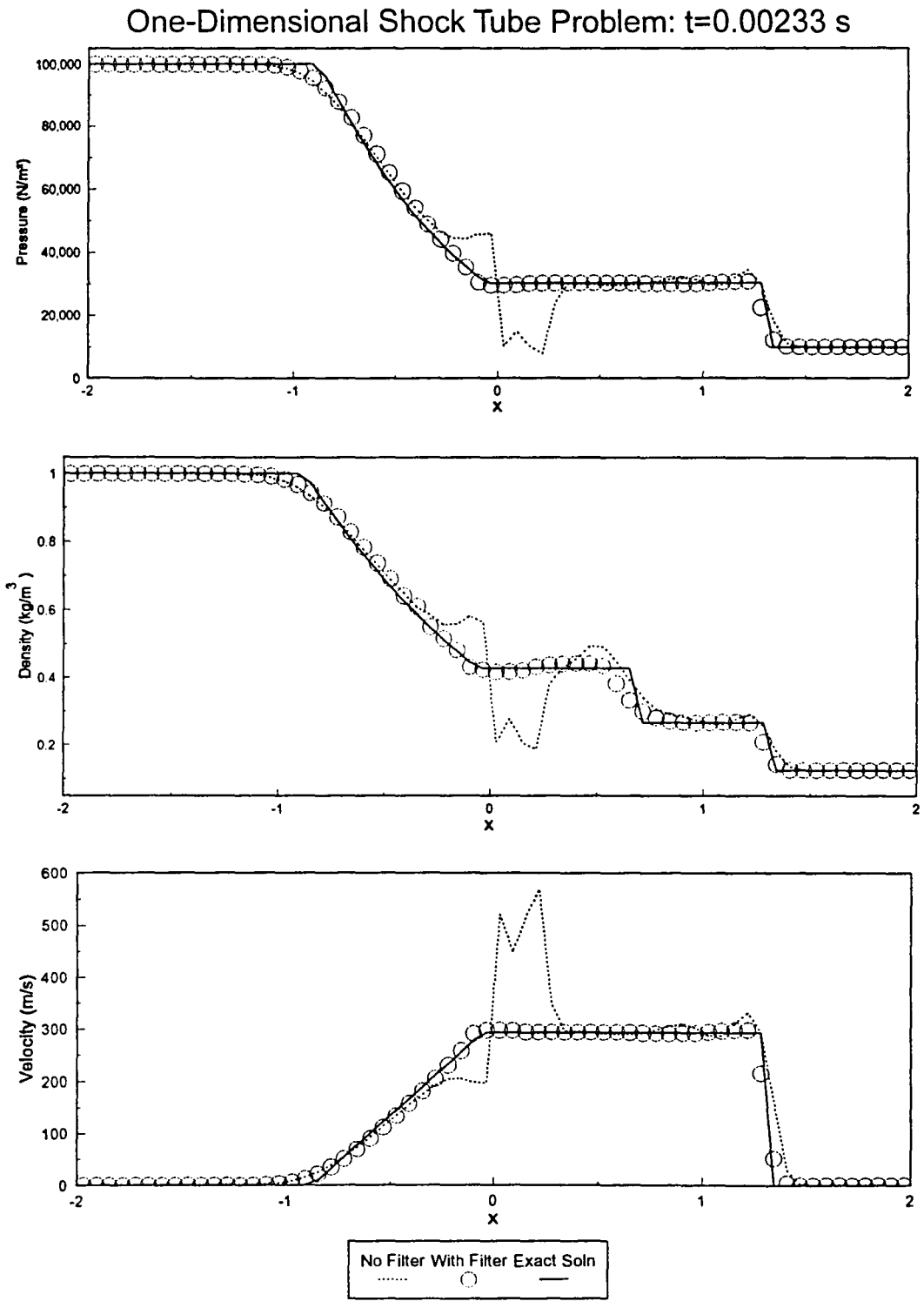

One-Dimensional Shock Tube Problem: t=0.00233 s
Pressure (N/m²)
100,000
80,000
60,000
40,000
20,000
0
-2
-1
0
1
2
x
Density (kg/m³)
1
0.8
0.6
0.4
0.2
Velocity (m/s)
600
500
400
300
200
100
No Filter With Filter Exact Soln

Figure 9

noted in the discussion of the Engquist theory, the filter is not completely TVD, but that the oscillations are of small amplitudes. This is shown by a slight waviness in the filtered solutions near the vicinity of a discontinuity (Figure 9).

Although convergence times could not be examined in this transient problem, the ability of the filter to maintain stability when the CFL number is overspecified can be looked into since the MacCormack scheme is an explicit one. For this case the theoretical CFL limit is one. The method, however, remained stable up to CFL = 1.2 without the aid of the filter. The results show oscillations even greater than those depicted in Figure 9. When the filter is applied, the method remains stable up to a CFL value of 1.65. Even more impressive is that the solution still accurately matches the exact solution. Thus, the filter is indeed able to increase the stability limit for this case. In theory, the one-dimensional MacCormack scheme with the filter would result in faster convergence times than the method without the filter.

Converging Duct:

The geometry and initial conditions for this problem are shown in Figure 4. The results of the simulation are shown in Figure 10, with the exact solution derived from oblique shock tables. The non-dimensional density is arrived at by dividing the dimensional value by the inlet value which in this case is 1 kg/m^3. A total of 1000

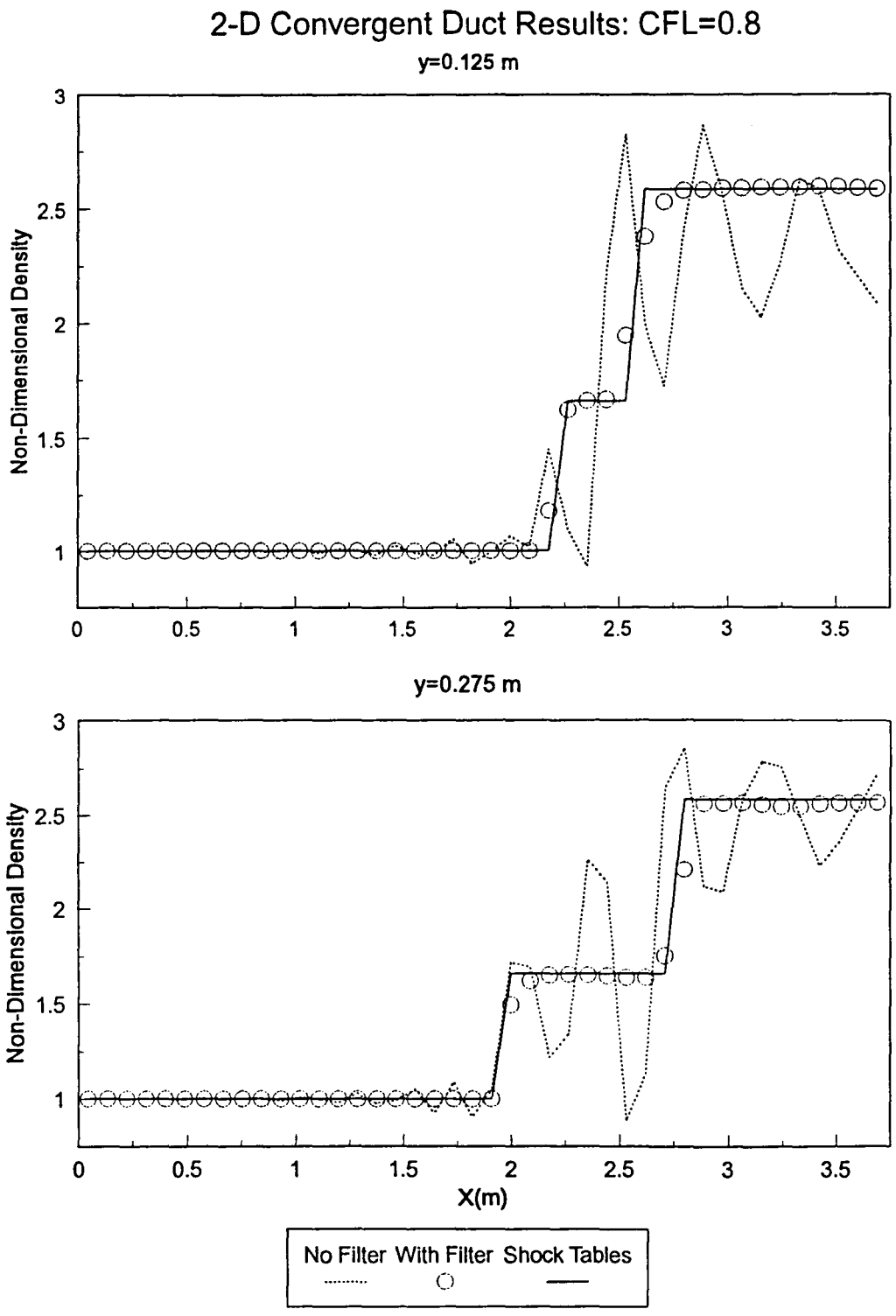

2-D Convergent Duct Results: CFL=0.8
y=0.125 m
Non-Dimensional Density
3
2.5
2
1.5
1
0
0.5
1
1.5
2
2.5
3
3.5
y=0.275 m
Non-Dimensional Density
X(m)
No Filter
With Filter
Shock Tables

Figure 10

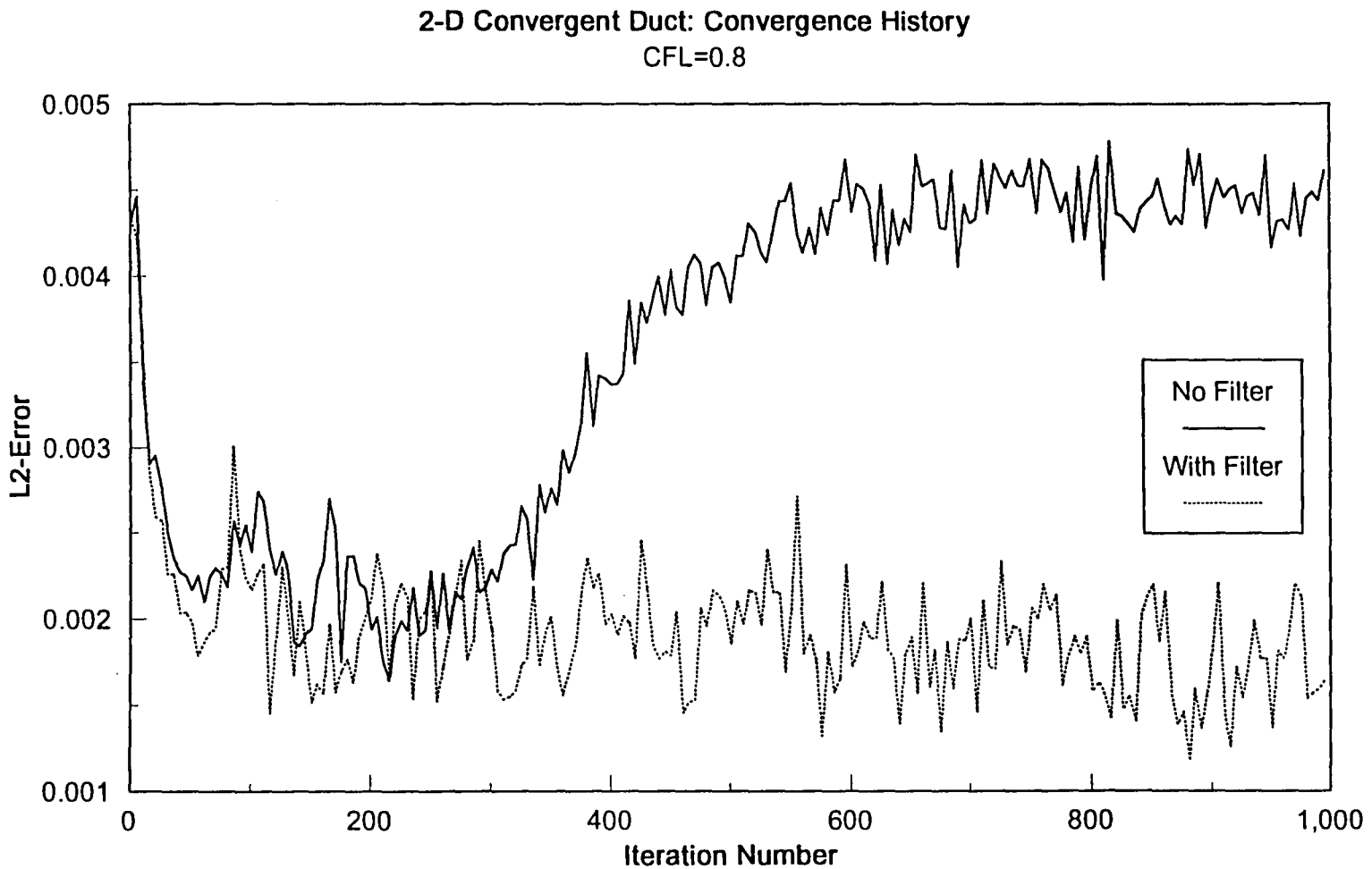

2-D Convergent Duct: Convergence History
CFL=0.8
0.005
0.004
0.003
0.002
0.001
L2-Error
0
200
400
600
800
1,000
Iteration Number
No Filter
With Filter

Figure 11

MacCormack time steps with a CFL number of 0.8 are allowed for each the filtered and non-filtered cases. Convergence is viewed via the convergence history of the L2-error (Figure 11). The L2-error is a relative measure of the change between consecutive iterations. Note that the MacCormack scheme remains stable for the nonfiltered case as indicated by the L2-error plateauing with small oscillations. The large L2-error is attributed to the large amount of dispersiveness in the solution as seen in Figure 10. The dispersiveness causes a node point to see greatly varying values for successive time steps and hence a large L2-error. The filter suppresses these variations and so shows lower L2-error. Thus, the "converged" nonfiltered solution cannot be trusted since at the next time step it will change drastically unlike the filtered case.

The total computational time shows a discrepancy as expected. For the no filter case the total CPU usage is 11.10 sec while for the filtered case it is 18.54 sec. That is an increase of 67% which is attributable to the calculation of eigenvectors. This percentage could have been larger if the filter was applied in both the x- and y-directions. However, in comparing the resulting solutions obtained from filtering in both directions as opposed to only along the x-direction, it is found that the accuracy is not greatly improved. Thus, filtering is done only in the primary flow direction.

Figure 10 displays the ability of the filter to capture discontinuities over a narrower bandwidth. For this grid, the spacing in the y-direction is uniform at each particular x-position. Thus, the spacing decreases as x increases since the duct converges. The position of the y-line shown in Figure 10 is referenced to its initial y-

position at the inlet of the duct. Note how dispersive the MacCormack scheme is near the discontinuities. The perturbations are so large that the flow discontinuities are no longer discernible. It appears that the unfiltered solution is about to become unstable. The filter, on the other hand, is able to suppress these large perturbations and also result in a solution that compares very well with the exact solution. Furthermore, the discontinuities are captured over a minimal number of node points (approximately four nodes). Hence, once again this aspect of the filter is supported.

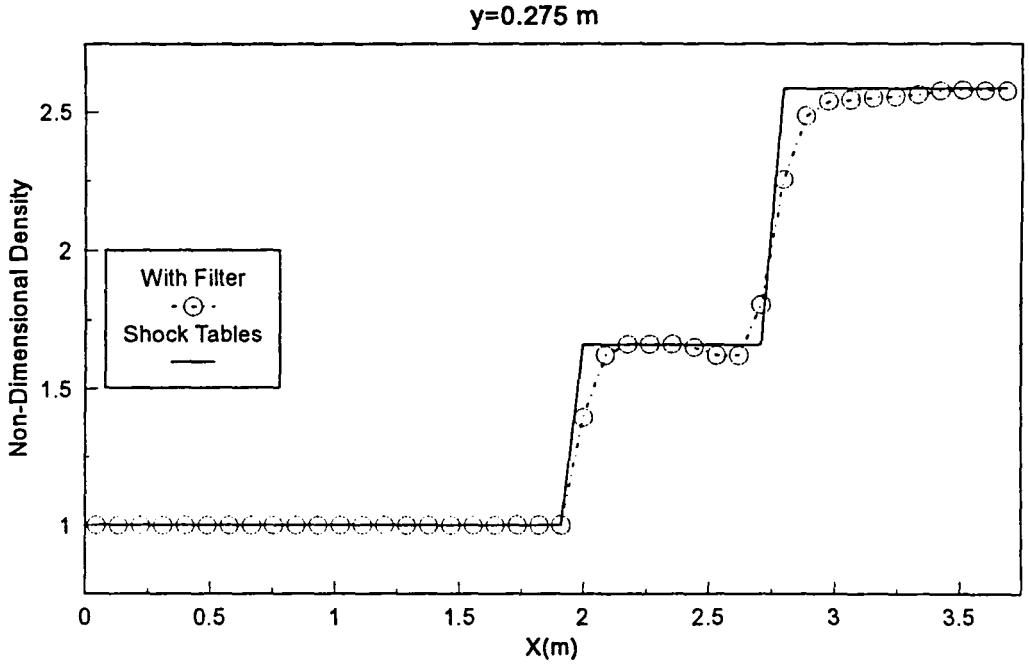
Since this problem is solved to steady-state, it is a good case to see whether the filter is able to converge to the solution faster. As stated earlier, the total CPU usage for 1000 time steps for the filtered simulation is 67% greater than the nonfiltered case. However, looking at a plot of the convergence history in Figure 11 shows that the filtered solution is converged at about 200 iterations while the nonfiltered case converges at 500 iterations. Taking this into account, the filter converges in 3.71 CPU seconds while the nonfiltered case converges in 5.55 CPU seconds. That is an acceleration of approximately 33%. Furthermore, the filtered case results in a more accurate solution.

Another aspect of the simulation is the ability of the filter to increase the stability regime of the explicit MacCormack scheme. Theoretically the scheme is stable up to a CFL of one. However, the large fluctuations only allow the scheme to be stable up to CFL = 0.9. The filter though is able to suppress the unstabilizing nature of the perturbations and allows the filtered scheme to be stable up to CFL =

1.25. Furthermore, the resulting solution is quite accurate as shown in Figure 12. The filter, though, has to battle the much larger perturbations which are evident in the convergence history (Figure 12). Thus, to accelerate convergence, one could initially use a large CFL number and decrease this value later in the simulation to obtain better accuracy.

Results for the filtered solutions for increasing deflection angles (the angle of the top ramp in Figure 4) are plotted in Figure 13. Here, the dimensional pressure is nondimensionalized by the inlet pressure. As can be seen, as the deflection angle increases from 5° to 10°, the shock is resolved more compactly. This is especially noticeable in the last shock jump. For 5° the shock is resolved over 9 node points, for 7° over 8 nodes, and for 10° over 5 node points. Thus, as the shock is more aligned toward one coordinate direction (the closer to normal it is) the better the ability of the filter to capture it.

2-D Convergent Duct: CFL=1.25



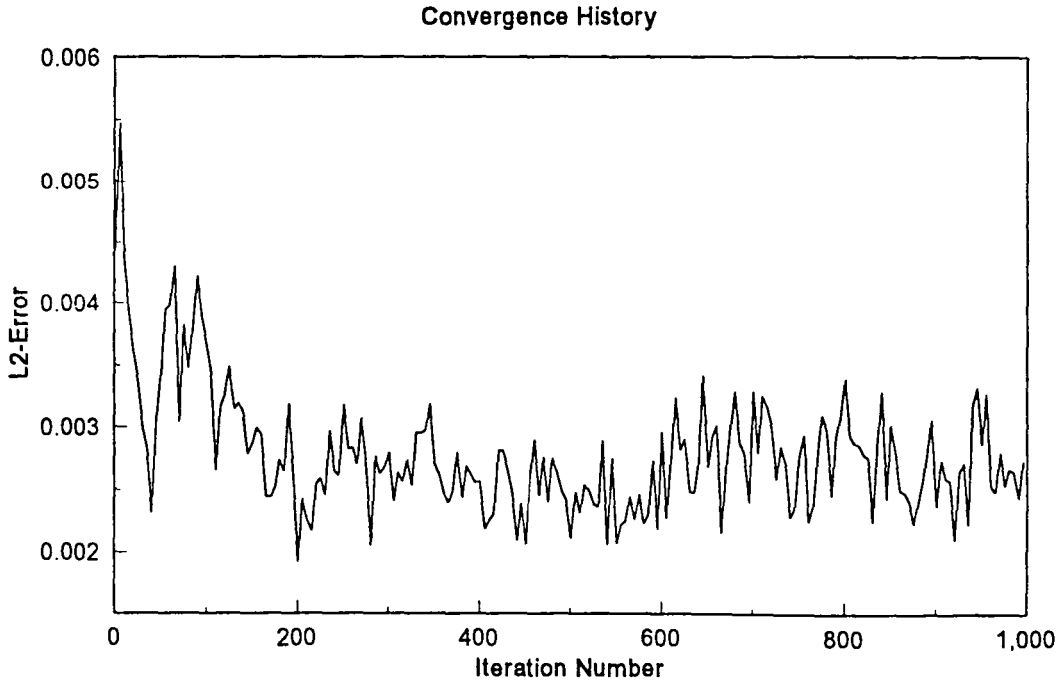


Figure 12

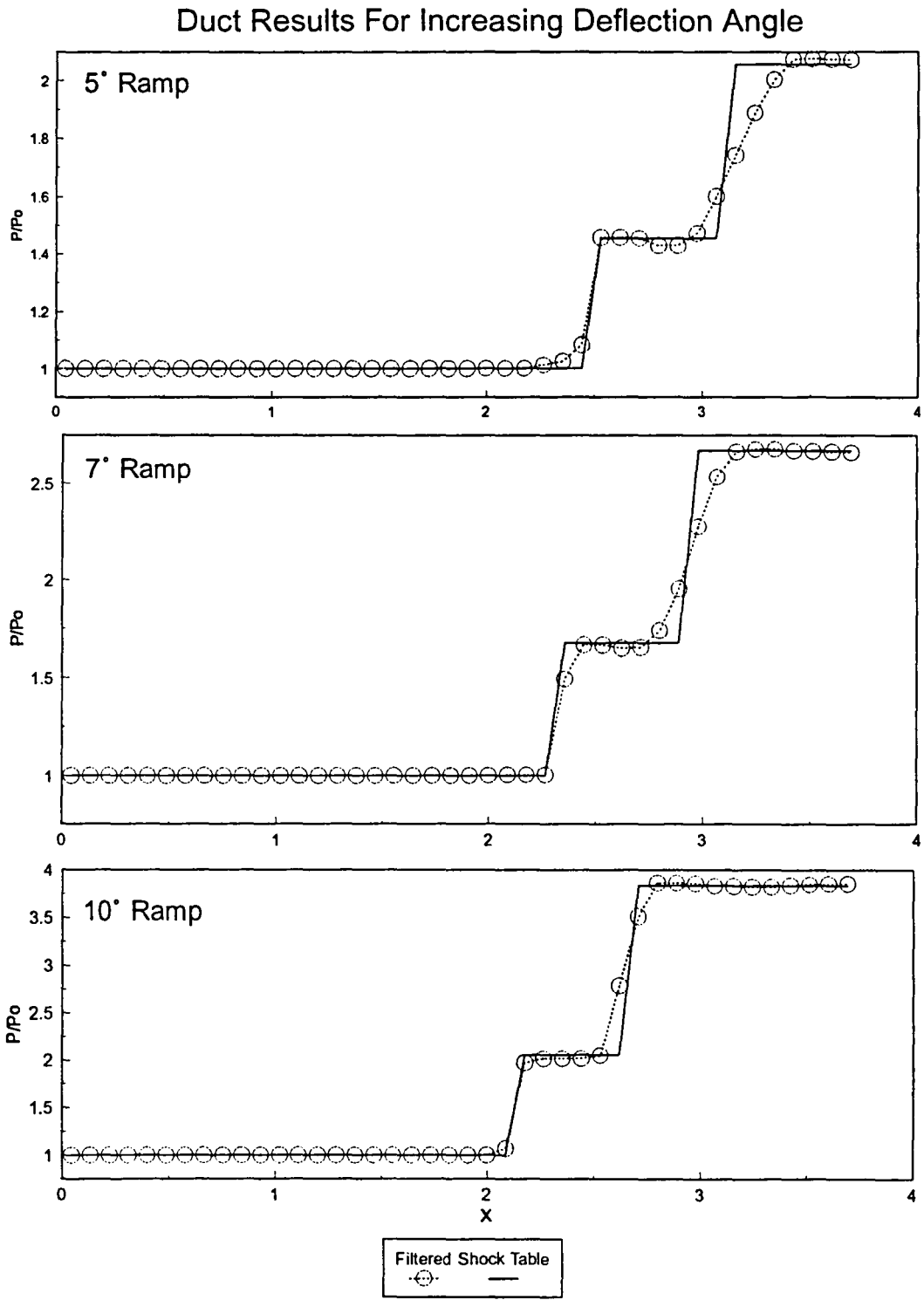

Duct Results For Increasing Deflection Angle
5° Ramp
7° Ramp
10° Ramp
P/Po
X
Filtered Shock Table

Figure 13

Shock-Boundary Layer Interaction:

The results of this problem are shown in Figures 14-15. The experimental data are taken from Hakkinen et al. (1959). The incoming flow is an ideal gas at Mach 2. The shock strength is varied by altering the pressure jump between the inlet and exit (behind the reflected shock). For $P_f/P_i = 1.2$, the shock is not strong enough to retard the flow in the boundary layer and hence does not cause separation. For $P_f/P_i = 1.4$, the shock causes separation and hence negative skin friction values. Note that the experimental data do not include negative measurements of skin friction since their apparatus was unable to measure it. Thus, the primary point to look at is the ability of the code to capture the separation and reattachment points. The geometry and flow conditions are described in Hakkinen et al. (1959) and also in MacCormack (1971).

As described earlier, this two-dimensional problem is solved as a pseudo-three-dimensional problem to be able to use the PARC3D code. The dimensions of the grid are 95x10x83. The 10 lateral node points make the two-dimensional problem a three-dimensional problem. The grid is also divided following MacCormack (1971), into two regions--a viscous boundary layer and an inviscid region. The boundary layer region extends from the surface to a height of 0.127 cm and the inviscid region to a height dependent on the shock height selected so that the incident shock would hit the boundary layer at the required x location. For the unseparated case, the height is taken to be 2.973 cm while for the separated case the height is 3.166 cm. A total of three

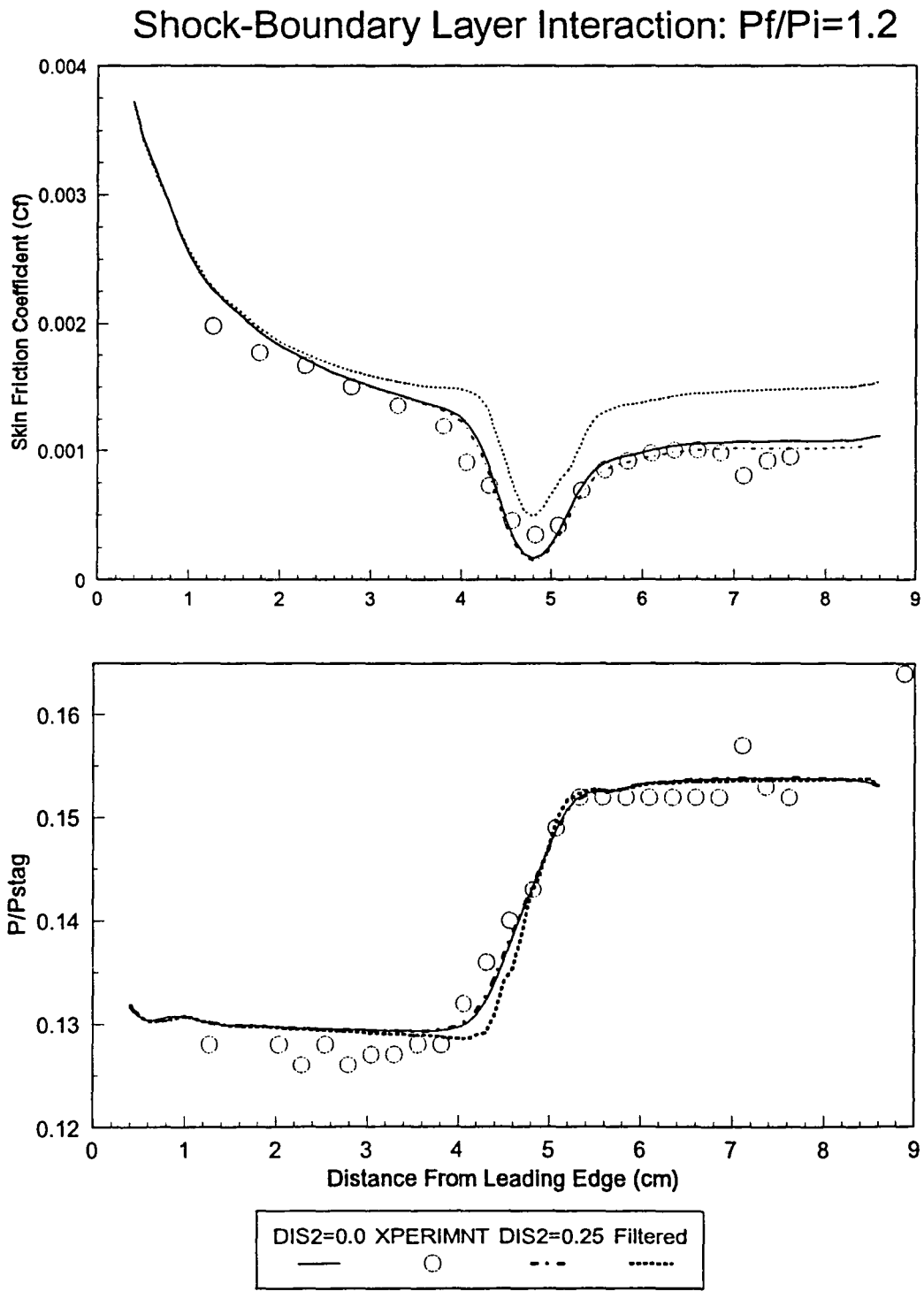

Shock-Boundary Layer Interaction: Pf/Pi=1.2
Skin Friction Coefficient (Cf)
0.004
0.003
0.002
0.001
0
0 1 2 3 4 5 6 7 8 9
P/Pstag
0.16
0.15
0.14
0.13
0.12
0 1 2 3 4 5 6 7 8 9
Distance From Leading Edge (cm)
DIS2=0.0 XPERIMNT DIS2=0.25 Filtered

Figure 14

simulations are done. One is with the PARC3D artificial dissipation parameter (DIS2) set to zero, one with it at 0.25, and one with DIS2=0 and the filter.

PARC3D is used so that it could be tested against proven experimental data unlike those taken during the Pegasus flights. The primary calculations are pressure and skin friction. To determine convergence, the L2-error history is looked at as well as the temperature gradient variation at one particular grid location (65,2,1). The PARC3D L2-error plotted in Figures 25-26 of Appendix B is a relative measure of convergence and hence should not be completely relied upon to determine convergence. Thus, it was decided that the temperature gradient at a particular location might be a more proper indicator.

Convergence for this problem varies as expected between the filtered and nonfiltered cases. The additional task of computing eigenvalues results in the filter being more computationally expensive. It would have been even more expensive if the filter had been dimensionally split and applied in all three coordinate directions. However, after an initial investigation, it is found, as in the convergent duct problem, that applying the filter in the x-direction is sufficient for this problem. In all, the filtered simulation takes about 7.55 CPU sec per iteration while the nonfiltered cases take 6.09 CPU sec per iteration. Thus, the filter results in a 24% penalty in computational time per iteration. However, the convergence histories of the simulations (Figures 25-26 in Appendix B) show that the filtered solutions converge in fewer iterations than the nonfiltered cases. Taking this into account, it is seen in

Table 1 that the filter actually results in an 18.7% (average) decrease in CPU time to convergence. Thus, even though PARC3D is an implicit code (hence the ability of the filter to increase its theoretically infinite stability limit is meaningless), the filter still manages to decrease the required CPU time.

In the convergence histories (Figures 25-26 in Appendix B), one notices that the L2-error tends to oscillate first while the temperature gradient attains a more steady value quite quickly. However, correlation of convergence as indicated by the L2-error and that by the temperature gradient is quite good.

Simulation	Iterations To Convergence	Iterations To Convergence	Convergence CPU Time	Convergence CPU Time
	$P_f/P_i = 1.2$	$P_f/P_i = 1.4$	$P_f/P_i = 1.2$	$P_f/P_i = 1.4$
DIS2=0.25 No Filter	380	400	2314.2 sec	2436 sec
DIS2=0 No Filter	350	400	2131.5 sec	2436 sec
DIS2=0 Filtered	250	250	1887.5 sec	1887.5 sec

Table 1

Unseparated Case

Figure 14 shows the results for this particular case. Note that the skin friction does not go negative and hence the boundary layer remains attached throughout the problem. The property of the filter to steepen gradients is very evident in the pressure plots. In the plate surface pressure plot, one can easily see that the pressure jump is captured over a narrower bandwidth than the nonfiltered cases. Also, since pressure variations are not as scattered in the boundary layer (the geometry is of a flat plate) except for near the shock impingement, the filtered and nonfiltered pressure values are basically identical away from this point. For all cases, the numerical solutions yield values that are slightly higher than those measured but the general shape of the experimental pressure variation is captured.

For the skin friction plot, the discrepancy between the nonfiltered cases and the filtered case is quite evident. The filter steepens the velocity gradient in the boundary layer so much that the resulting skin friction values are higher than those measured. It still follows the general shape of the experimental results but not as well as the nonfiltered cases. The nonfiltered cases actually predict the skin friction values rather well throughout the plate surface but underpredict the reattachment location. Note that, although only slight, one can see that the simulation for DIS2=0.25 shows a little more diffusion than its DIS2=0.0 counterpart. For this problem, the artificial dissipation parameter was able to be set to zero primarily because the grid is not very skewed. This will not be the case for a more skewed grid as will be seen in the Pegasus

simulations. Nevertheless, PARC3D displays its capability for handling this problem.

Separated Case

Figure 15 shows the pressure and skin friction plot for the separated case. Note that the pressure measured shows a slight plateau in the separated region. This plateau, however, is not captured in the filtered numerical solution which yields a smaller separated region, as shown by the skin friction plot. This is the result of not having enough grid points near the separated area and by the filter trying to resolve the discontinuity in as compact a space as possible. As for the nonfiltered case, the results show a slight inflection point near the pressure plateau due to the fact that the calculated separated region is wider than that in the filtered simulation. Thus, it has a few more node points over which to resolve the plateau. It still manages, however, to diffuse the results with the DIS2=0.25 case showing the most diffusion.

The filtered results overpredict the skin friction and manage to strengthen the boundary layer thereby decreasing the length of the separated region. For better accuracy, one would need only to look at the nonfiltered cases where the skin friction contour is basically captured. Thus, for this case, it appears that the filter tries to capture the shock within a bandwidth so narrow that it limits its effects to a smaller region than what is actually experienced due to the natural diffusion of the flow. However, PARC3D is still able to capture the general trends.

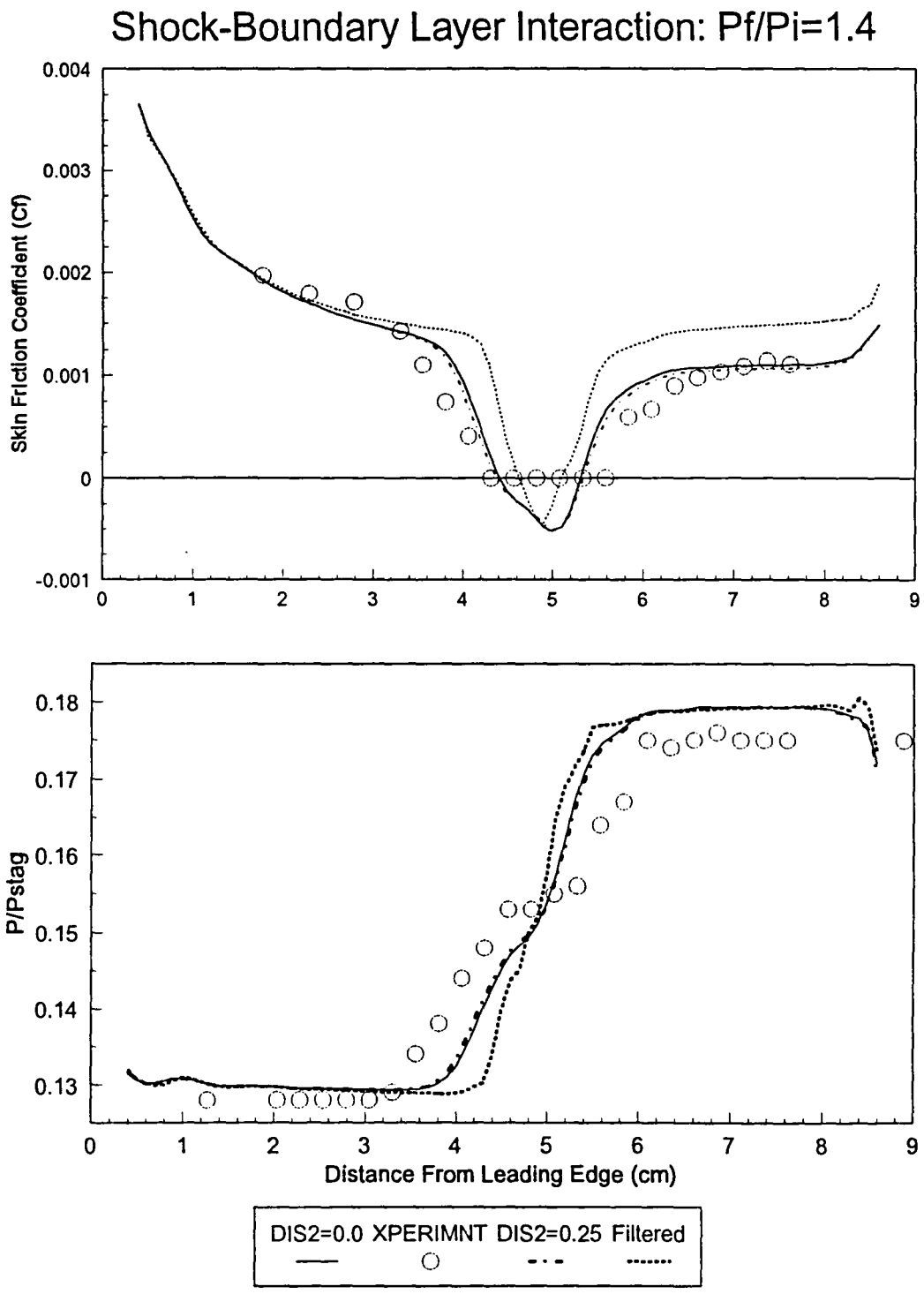

Shock-Boundary Layer Interaction: Pf/Pi=1.4
Skin Friction Coefficient (Cf)
0.004
0.003
0.002
0.001
0
-0.001
0
1
2
3
4
5
6
7
8
9
P/Pstag
0.18
0.17
0.16
0.15
0.14
0.13
Distance From Leading Edge (cm)
DIS2=0.0 XPERIMNT DIS2=0.25 Filtered

Figure 15

In looking at the plots for the unseparated and separated cases, it appears that the artificial dissipation parameter does not affect the results very much. However, this is only so for weaker shocks. For stronger shock strengths, the effects of the artificial dissipation parameter are felt tremendously. Figures 16a and 16b (with Figure 16c and 16d showing the shock impingement area in greater detail) show the results for a shock with the same flow conditions as above but with $P_f/P_i = 2.4$. Note the large difference when the artificial dissipation is varied. Thus, care must be taken when choosing a value for this parameter.

For the above simulations, the grid is refined in both the x and y directions. In the x-direction, the shock impingement region (between 3 and 7 cm) is given twice as many nodes as in the previous cases. Thus, in this region Δx is 0.05 cm, while outside of this region $\Delta x = 0.1$ cm. In the y-direction, the grid is algebraically stretch following Equation 5-216 in Anderson, et al. (1984) with the stretching parameter given the value of 1.0045. In total, the x-direction has 135 node points while the y-direction has 141 node points.

The results for all of the simulations show that the separation point is numerically determined to be substantially farther down the plate than that measured in the laboratory (Figure 16a). This is because PARC3D is unable to capture the pressure propagation far upstream of the shock impingement point that is displayed in the experimental results (Figure 16b). Instead, PARC3D localizes the pressure influence to a narrower region as to create the nonphysical situation of two (at one

instance, three for the DIS2=0.09 case) separated regions (Figure 16c). The large drop in pressure as calculated by PARC3D at 5.8 cm causes a low pressure region which "sucks" the separated layer above it to reattach before separating once again due to the large adverse pressure gradient that follows (Figure 16d). Note that the ability of the filter to suppress numerical oscillations leads to it being the only simulation to maintain one separated region (Figure 16c , k=2 line). Also note that the filter best approximates the location of reattachment and post-reattachment skin friction. Thus, for this case the filter shows the more accurate results.

The oscillations in the skin friction plot are attributable to the large gradients created in the narrower region of pressure influence. These extrema do not stay at one particular location but instead fluctuate from point to point within the separated region. Further investigations show the same results. One simulation, (DIS2=0.25 and no filter) in which the leading edge is further refined in the x-direction, shows similar results to the DIS2=0.25 case in Figures 16a-16d. Thus, leading edge grid coarseness is concluded not to cause these oscillations or the delayed initial separation point. Furthermore, since this problem is inherently transient in nature, it was decided to view the skin friction values over successive iterations. The results for this simulation show that while the locations of separation and reattachment are basically constant, the extrema in the separated region varied in both location and magnitude. However, a time average still results in positive skin friction values between the first separation and last reattachment points. Thus, due to their inherently fluctuating manner, the

oscillations are concluded to be nonphysical and the result of numerical oscillations.

For the $P_f/P_i = 2.4$ case, it was decided to investigate the use of the filter farther from the plate surface where the physical viscosity is large and should therefore damp out oscillations naturally (Figures 16a-16d, k=41 line). Thus, the filter is applied from k=41 (0.0941 cm above the plate) and greater. However, it is found that artificial viscosity (DIS2=0.09) is still required for the nonfiltered region (below k=41) to maintain numerical stability. The results show that the composite method is similar to the DIS2=0.09 case which displays more oscillations (Figure 16a-d) and similarly results in two separated regions. Furthermore, this simulation incurred a 23.5% CPU increase per iteration over the DIS2=0.09 case. Thus, this composite method is undesirable since it does not obtain better accuracy than the fully filtered case and results in approximately the same computational time penalty. An interesting note is that when the filter is applied from the wall and on (k=2), the artificial dissipation is able to be completely removed (set to zero).

The use of the filter for the two-dimensional shock-impingement-on-a-laminar-boundary-layer problem is dependent on the shock strength. Although it increases the CPU time per iteration, the filter requires fewer iterations to convergence. Thus, in total it reduces the amount of computational time required to convergence. As for accuracy, for the weaker shock strengths it is not improved and hence the filter is not needed. However, for the stronger shock strengths, the filter is able to improve accuracy through suppression of numerical oscillations and so is warranted.

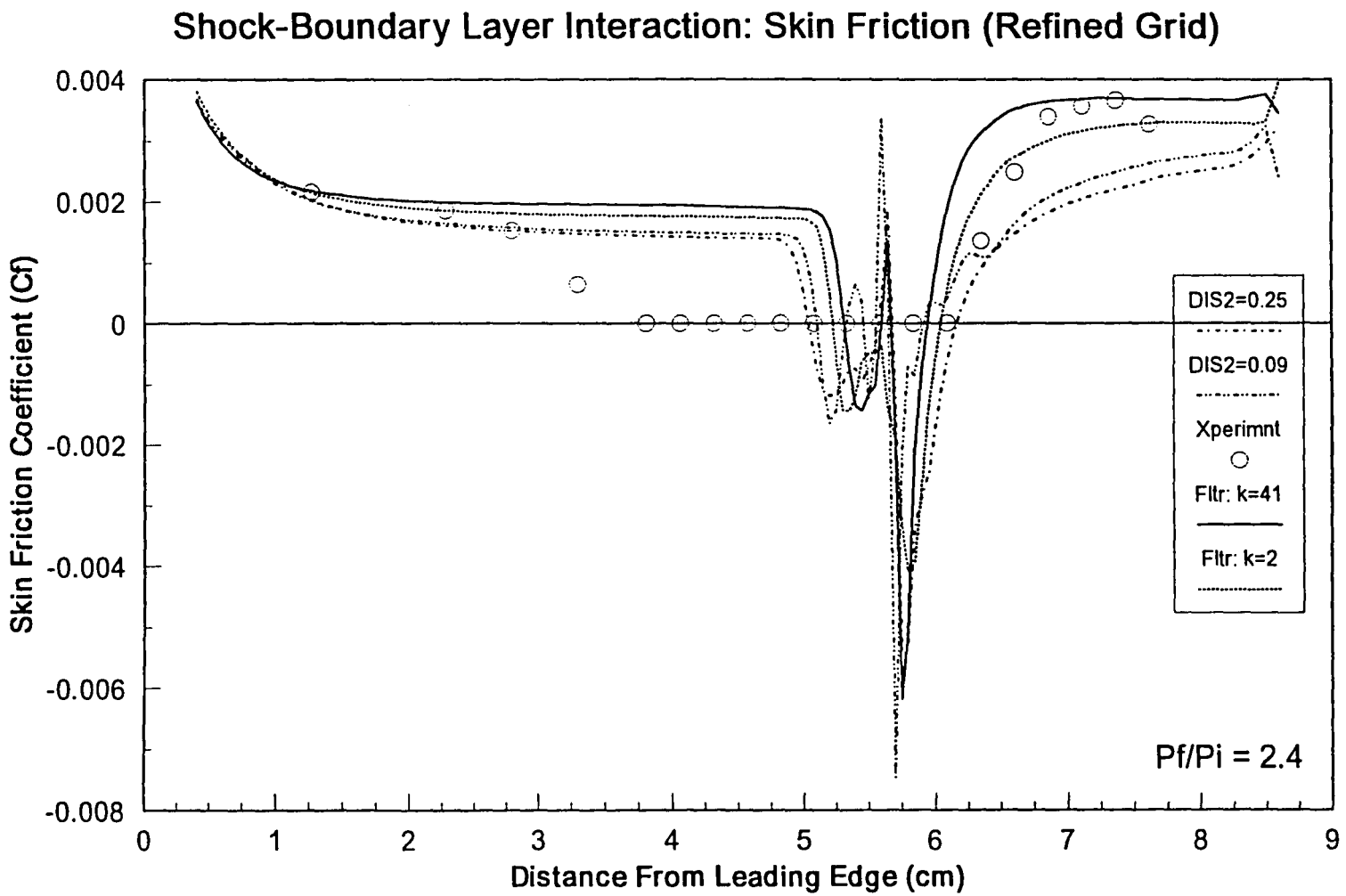

Shock-Boundary Layer Interaction: Skin Friction (Refined Grid)
Skin Friction Coefficient (Cf)
0.004
0.002
0
-0.002
-0.004
-0.006
-0.008
0
1
2
3
4
5
6
7
8
9
Distance From Leading Edge (cm)
DIS2=0.25
DIS2=0.09
Xperimnt
Fltr: k=41
Fltr: k=2
Pf/Pi = 2.4

Figure 16a

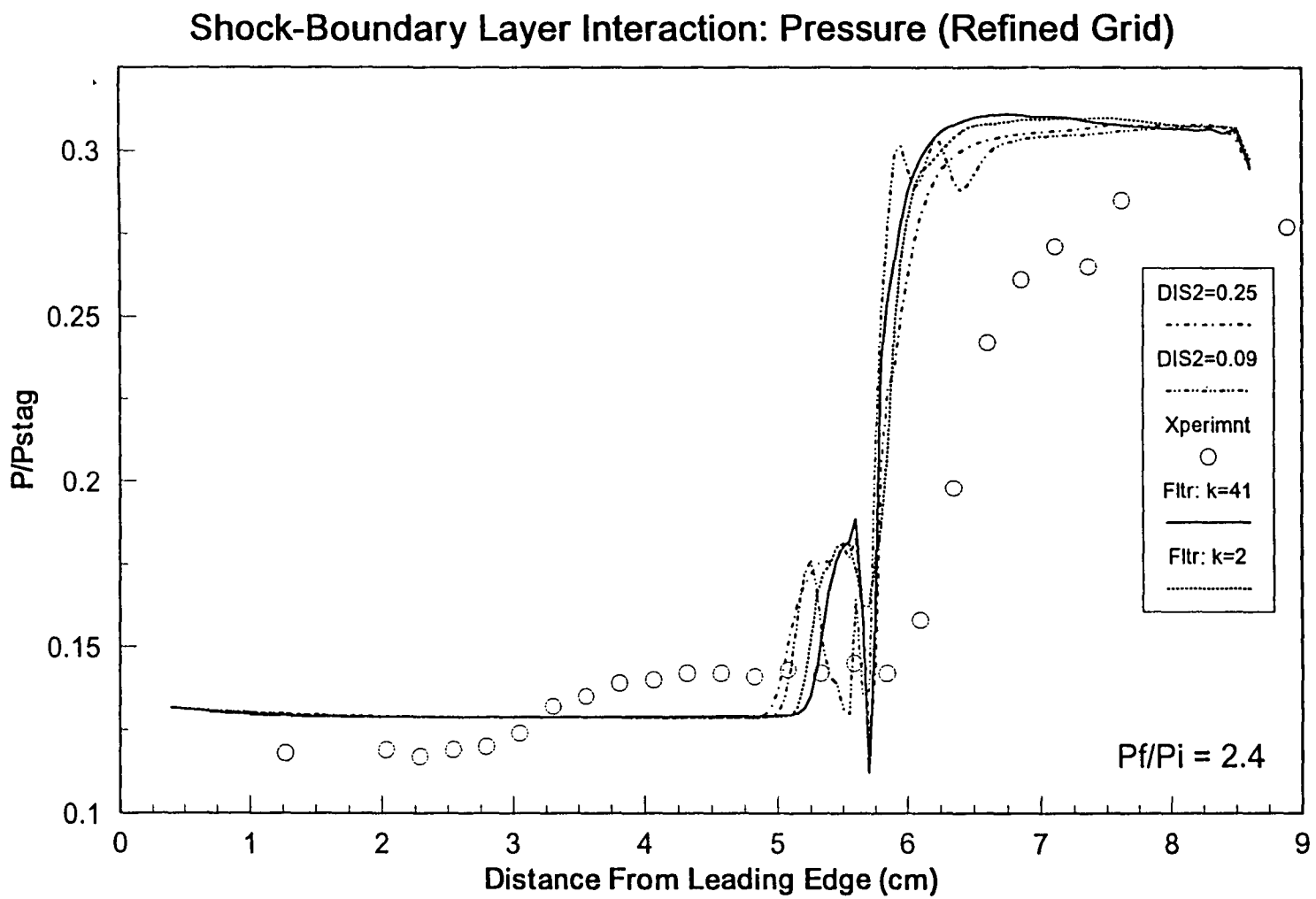

Shock-Boundary Layer Interaction: Pressure (Refined Grid)
P/Pstag
0.3
0.25
0.2
0.15
0.1
0
1
2
3
4
5
6
7
8
9
Distance From Leading Edge (cm)
DIS2=0.25
DIS2=0.09
Xperimnt
Fltr: k=41
Fltr: k=2
Pf/Pi = 2.4

Figure 16b

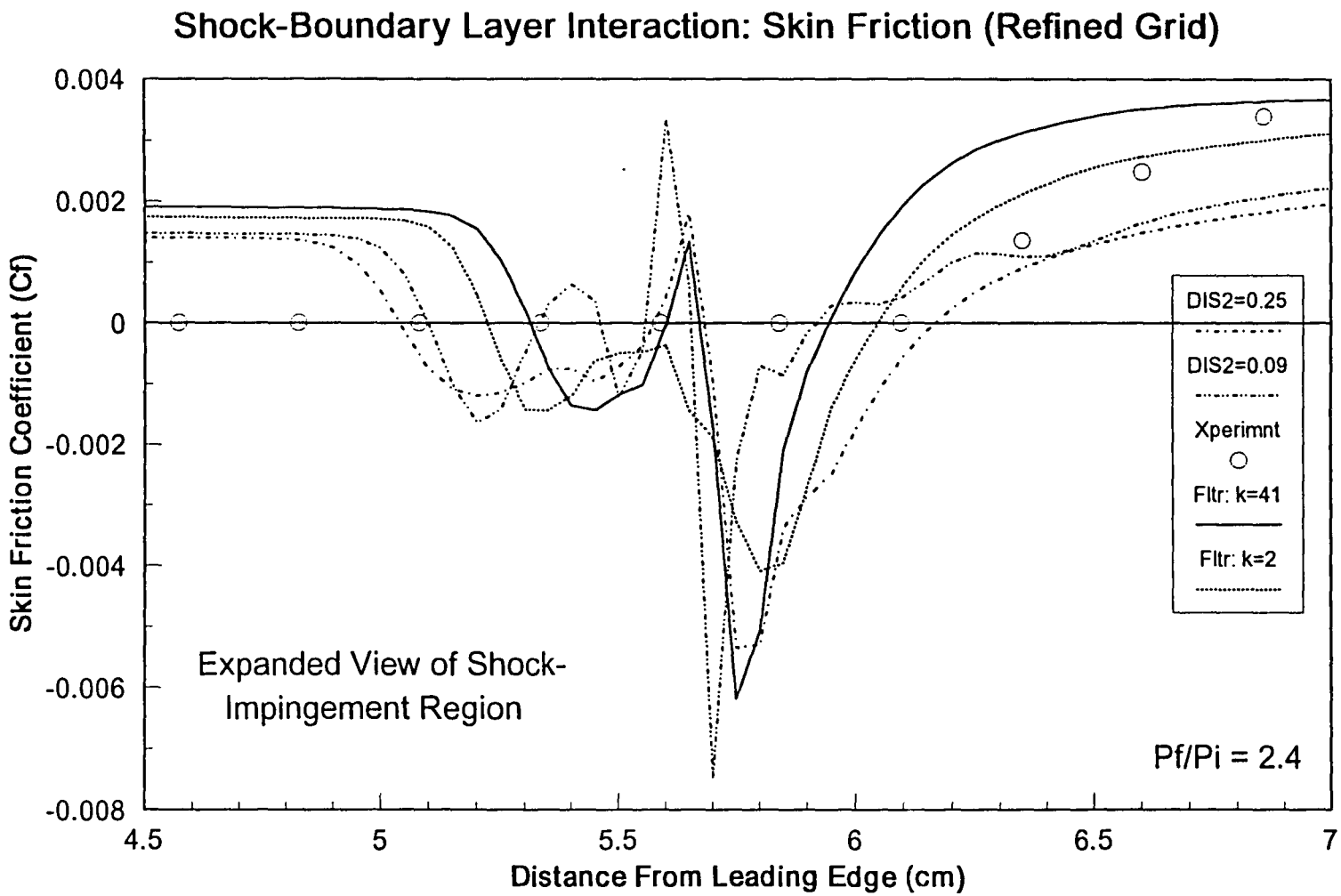

Shock-Boundary Layer Interaction: Skin Friction (Refined Grid)
Skin Friction Coefficient (Cf)
0.004
0.002
0
-0.002
-0.004
-0.006
-0.008
4.5
5
5.5
6
6.5
7
Distance From Leading Edge (cm)
DIS2=0.25
DIS2=0.09
Xperimnt
Fltr: k=41
Fltr: k=2
Expanded View of Shock-Impingement Region
Pf/Pi = 2.4

Figure 16c

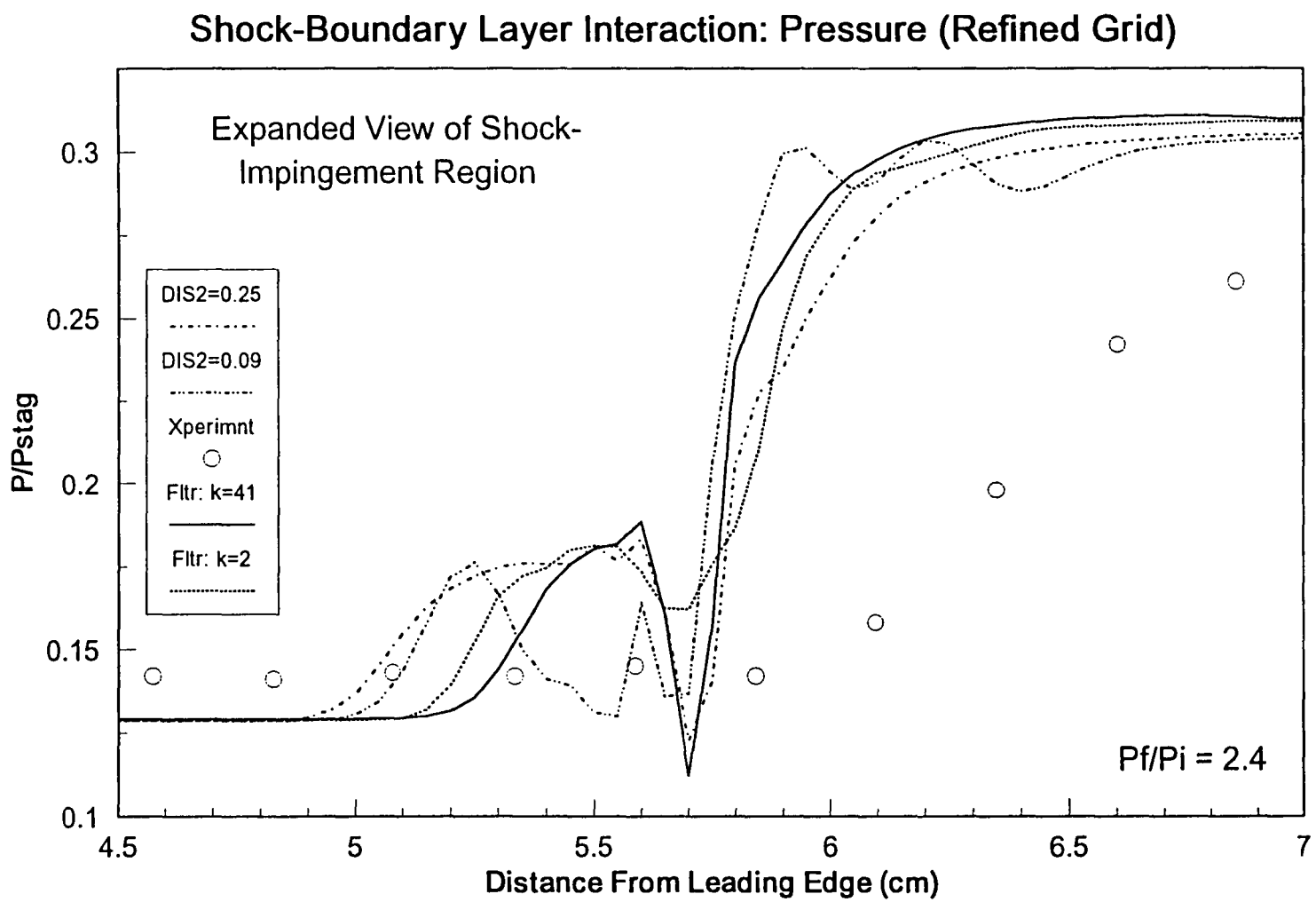

Shock-Boundary Layer Interaction: Pressure (Refined Grid)
Expanded View of Shock-Impingement Region
DIS2=0.25
DIS2=0.09
Xperimnt
Fltr: k=41
Fltr: k=2
Pf/Pi = 2.4
P/Pstag
0.3
0.25
0.2
0.15
0.1
4.5
5
5.5
6
6.5
7
Distance From Leading Edge (cm)

Figure 16d

Pegasus Flights: Heat Transfer

The surface heat fluxes are calculated at positions corresponding to the locations of the HRSI plugs. The fillet region and the locations of the HRSI plugs for the two flights are shown in Figure 8 and Figure 17, respectively. Note that the HRSI plugs recorded temperatures which were later converted to surface heat fluxes using the Lockheed Thermal Analyzer (LTA) program, see Noffz et al. (1991). Although these fluxes are not directly measured, they are taken as the reference points for the comparisons to be made later. According to Noffz et al. (1991), the heat fluxes have an uncertainty of ±22.6%.

The importance of computational efficiency is very evident in this problem. Due to the large scale of the Pegasus and the number of required grid points, the amount of CPU usage is on the order of hours as opposed to minutes for the problems discussed earlier. For the Pegasus simulations, the filtered simulation ("Fltrd") takes the longest time to convergence. On the IBM ES 9000 machine, the average CPU time used is 36.4 hours while the nonfiltered ("Low Disp") case converges in 26.9 CPU hours. As for the "Art Disp" case, Fricker et al. (1992) reports them to converge in about 50 CPU hours. The large discrepancy is attributed to the "Art Disp" simulations being run for more iterations than were necessary. These large amounts of CPU usage demonstrate why one would want to decrease them. It is noted in the previous section that the filter did improve the accuracy of the results enough to

HRSI Plug Location (Side View of Vehicle)

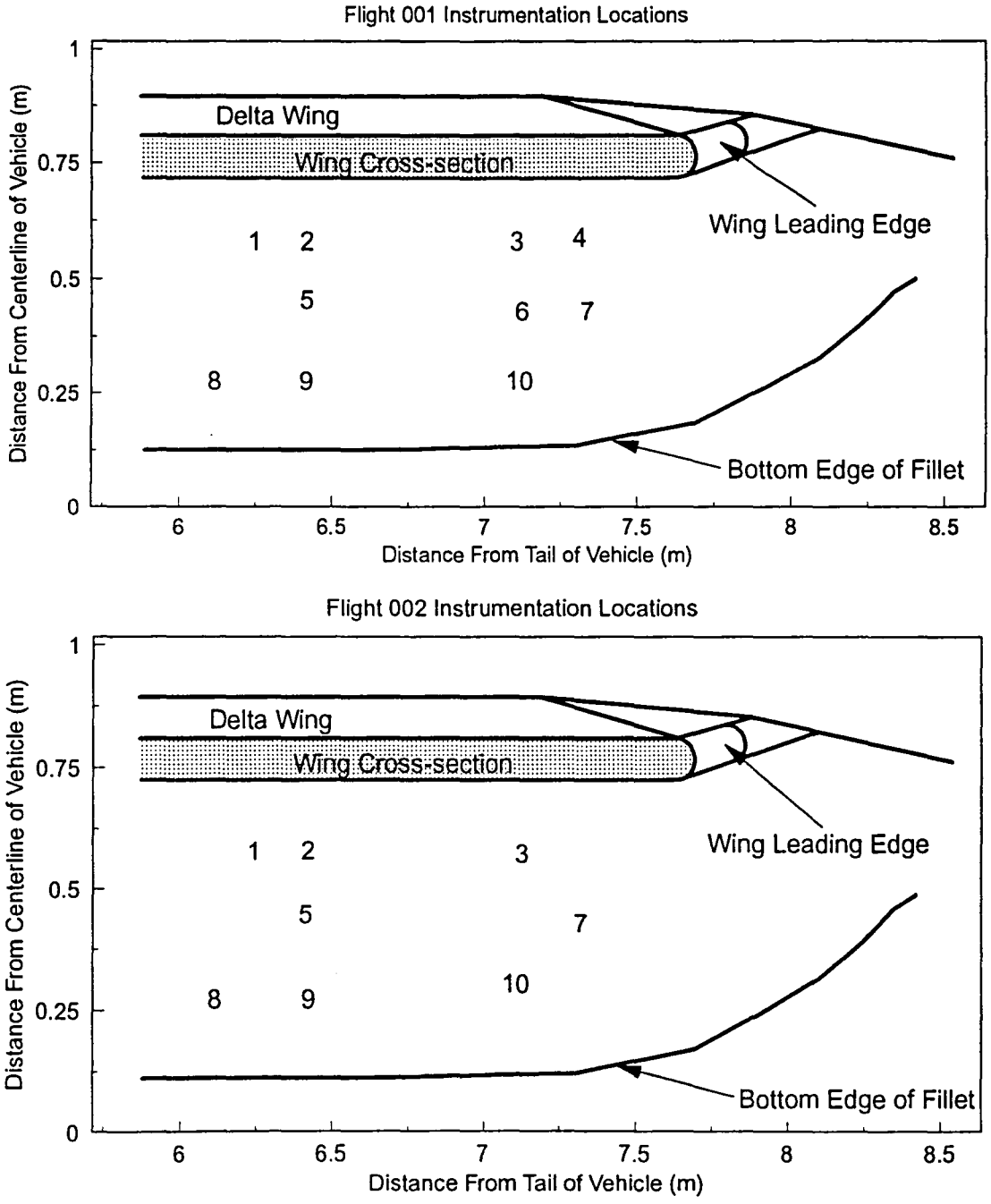


Figure 17

warrant the computational expense. Similarly, for this problem, it is found that the results did improve but not by much. Thus, the filter is an unnecessary addition to the PARC3D code for this problem.

One aspect to note in the upcoming discussion is the physical size of the Pegasus vehicle that is being simulated. Accurately calculating the skin friction on a 10-cm long plate is a feat in itself. However, to closely estimate the heat fluxes which are dependent on phenomena in the boundary layer (order of millimeters) while still solving the Navier-Stokes equations for the flowfield surrounding the vehicle (order of meters) would be quite an accomplishment.

Mach 3.52 $\alpha = 7.35°$ Flight 1

At an angle of attack, the shock tends to hug the underside of the delta wing. However, due to the presence of the front fillet ramp, a shock forms along the ramp such that a subsonic high pressure region exists behind it. This prevents the shock emanating from the forward portion of the wing (just above the ramp) from impinging on the fillet. Nevertheless, one expects, for a given flow direction, that the stagnation region and hence point of highest heating would be closer to the underside of the wing (plugs 1-4). The high pressure region created by the front fillet ramp is smoothly decreased as the flow runs over the fillet since the corner that connects the front ramp

to the fillet acts as a corner where an expansion fan forms. Across an expansion fan, the pressure, density, and temperature decrease as the flow accelerates over it. Towards the rear portion of the fillet, farther downstream from the effects of the expansion fan, the high pressure from the wing shock is felt. However, the shock is now farther away (due to the 45° sweep of the delta wing) from the fillet sidewall and hence does not impact on the wall. Its presence, though, is still felt by HRSI plug 5. The other plugs (6, 7, 8, 9, and 10) are farther from the stagnation corner and hence would be expected to see lower heating.

The above discussion is supported by the "Low Disp", "Fltrd", and "Flight" calculations cases (Figure 18). However, for the large dissipation case ("Art Disp") the heat fluxes are of relatively the same magnitude although there is a slight rise in the values at HRSI plugs 1, 2, 4, 5, and 6. Thus, it can be postulated that the artificial dissipation has diffused the overall shock strength instead of localizing it to one particular section of the fillet surface.

Mach 3.52 α = 2.65° Flight 2

The smaller angle of attack implies that the shock is not as close to the underside of the wing as in the Mach 3.52 Flight 1 case. Instead, what is seen is the effect of the abrupt change in curvature at the lower edge of the fillet region where the

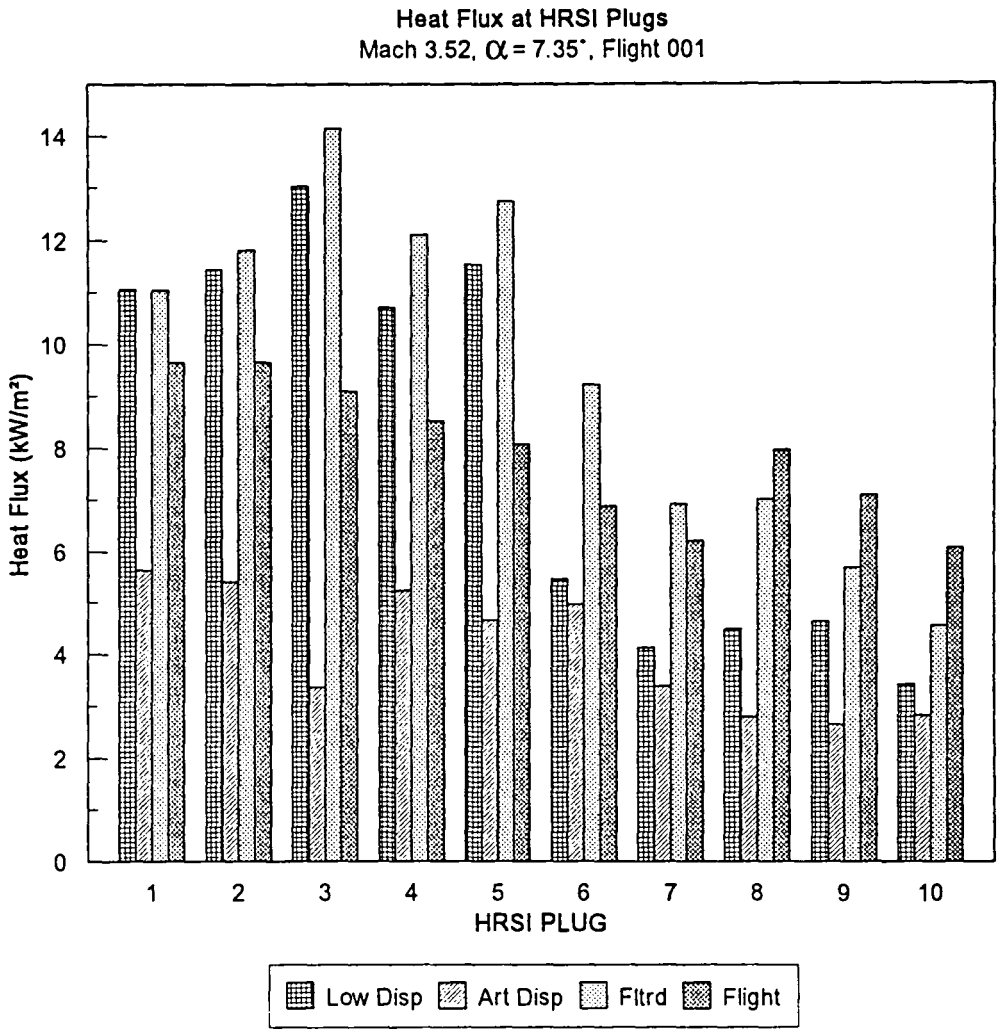

Heat Flux at HRSI Plugs
Mach 3.52, α = 7.35°, Flight 001
Heat Flux (kW/m²)
0
2
4
6
8
10
12
14
1
2
3
4
5
6
7
8
9
10
HRSI PLUG
Low Disp
Art Disp
Fltrd
Flight

Figure 18

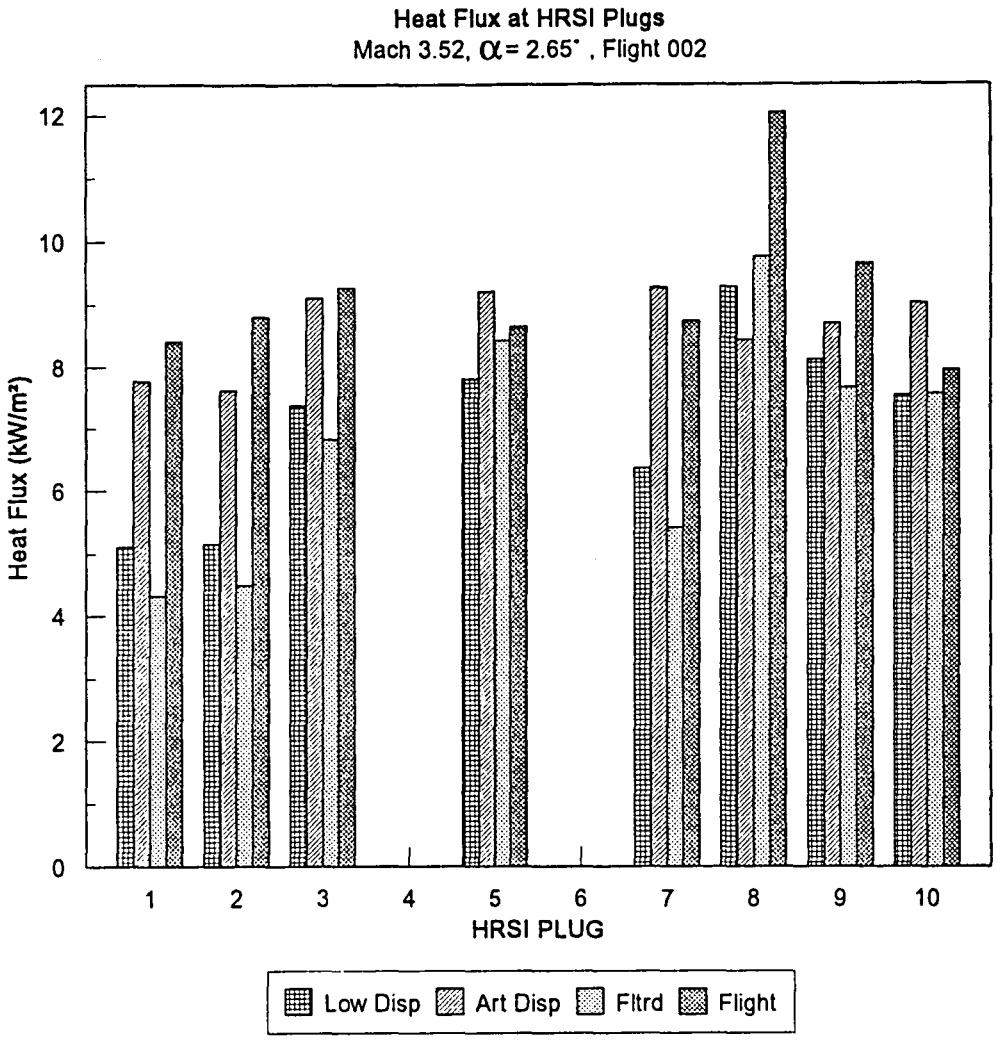

Heat Flux at HRSI Plugs
Mach 3.52, α= 2.65˚ , Flight 002
Heat Flux (kW/m²)
0
2
4
6
8
10
12
1
2
3
4
5
6
7
8
9
10
HRSI PLUG
Low Disp
Art Disp
Fltrd
Flight

Figure 19

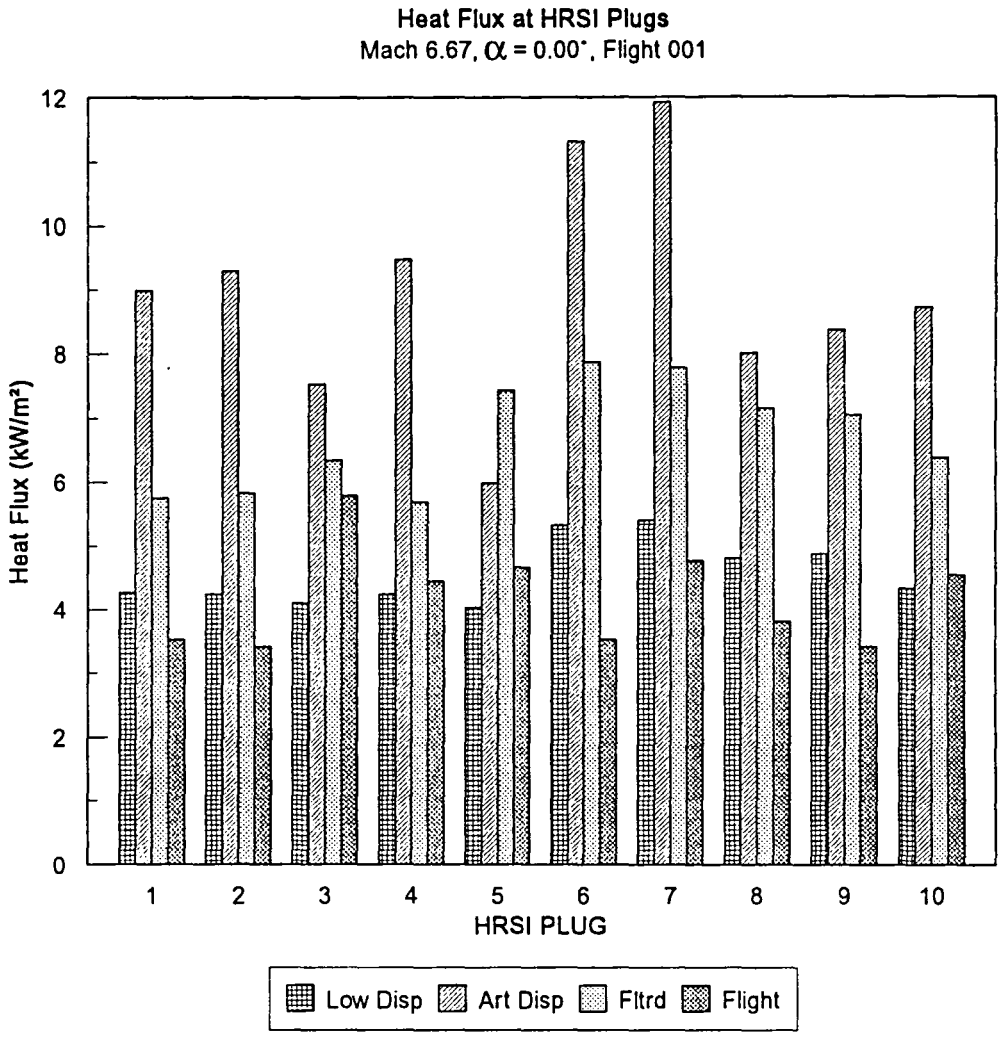

Heat Flux at HRSI Plugs
Mach 6.67, α = 0.00°, Flight 001
Heat Flux (kW/m²)
0
2
4
6
8
10
12
1
2
3
4
5
6
7
8
9
10
HRSI PLUG
Low Disp
Art Disp
Fltrd
Flight

Figure 20

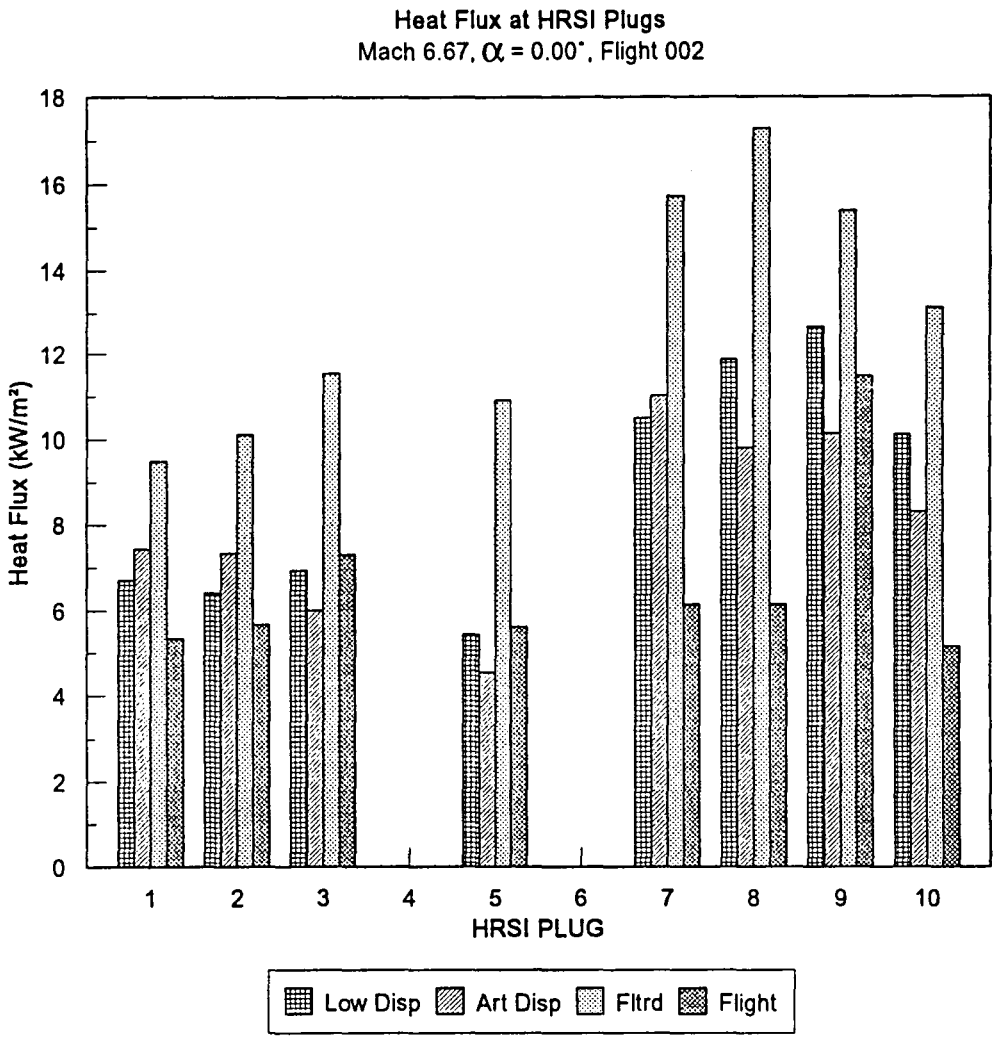

Heat Flux at HRSI Plugs
Mach 6.67, α = 0.00°, Flight 002
Heat Flux (kW/m²)
0
2
4
6
8
10
12
14
16
18
1
2
3
4
5
6
7
8
9
10
HRSI PLUG
Low Disp
Art Disp
Fltrd
Flight

Figure 21

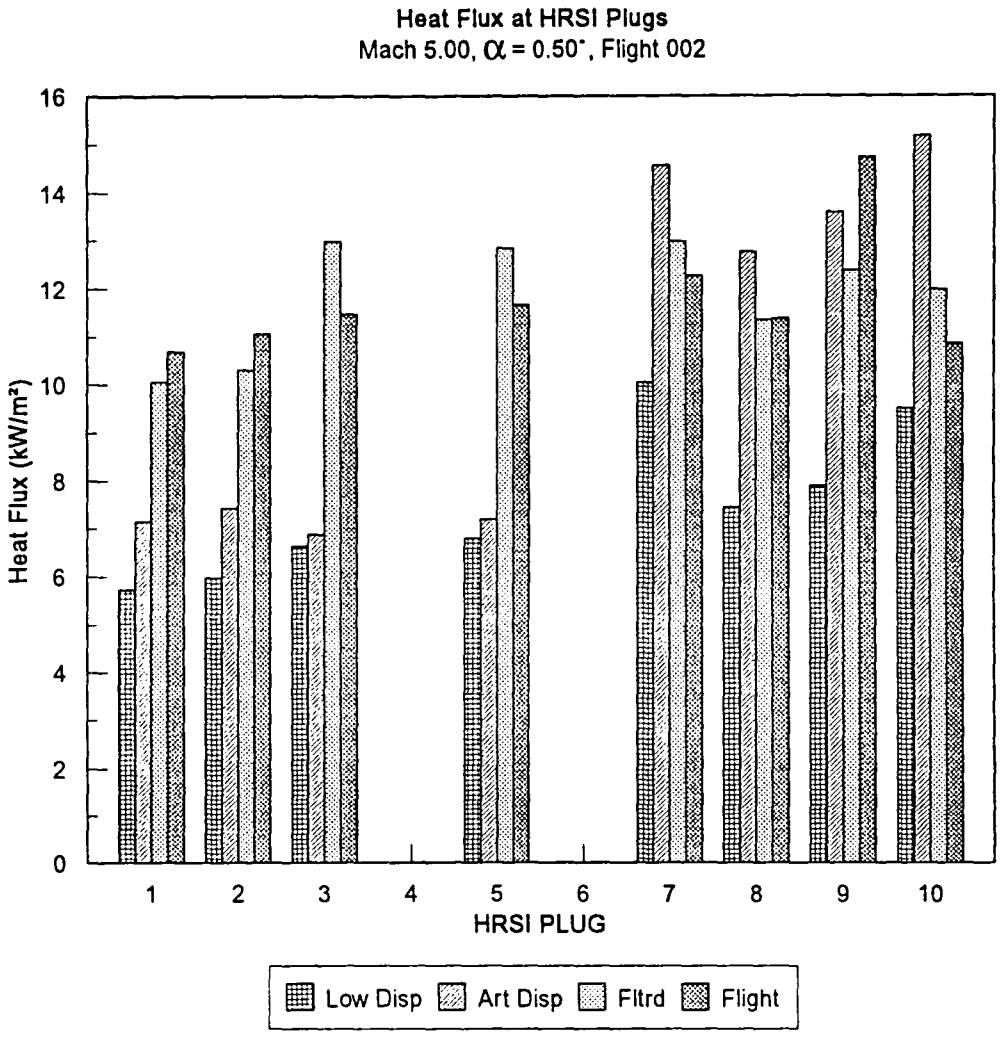

Heat Flux at HRSI Plugs
Mach 5.00, α = 0.50°, Flight 002
Heat Flux (kW/m²)
0
2
4
6
8
10
12
14
16
1
2
3
4
5
6
7
8
9
10
HRSI PLUG
Low Disp
Art Disp
Fltrd
Flight

Figure 22

fillet meets the cylindrical fuselage. This area becomes another compression corner. This is also the case near the expansion corner of the front ramp and fillet where the initial pressure is high in the compression region created by the front fillet ramp. Hence, it is expected that higher heating will be displayed at plugs 7 and 10. Furthermore, the smaller angle of attack results in the wing shock not being as close to the area of the fillet and wing root junction as before. Instead, it will be felt lower down, closer to the bottom edge of the fillet (plugs 8 and 9). As expected, the largest numerical magnitudes are found at HRSI plugs 7-10 (Figure 19). This is also generally supported by the flight data. Note that since the wing shock is not able to reach the upper front corner of the wing and fillet intersection, the lower pressure from the expansion fan from the front fillet ramp results in plugs 1 and 2 seeing the lowest amount of heating.

Figure 19 again shows the effect of the artificial dissipation. Instead of sharp contrasts between plugs, the calculated "Art Disp" heat fluxes are approximately the same magnitude. This implies that the artificial dissipation has diffused the shock strength over the entire fillet region. The lower dissipation cases ("Low Disp" and "Fltrd") do not exhibit this tendency.

Mach 6.67 $\alpha = 0°$ Flight 1

The higher Mach numbers occur later in the flight where ablation should be

occurring at a high rate. Figure 20 shows, as expected, that the calculated heat fluxes are larger than the flight data. This is because ablation lowers the amount of surface heating by absorbing some of the incoming energy and converting it into latent energy. It further lowers the surface heating by acting like a thermal shield and impeding incoming energy due to the positive mass flux from the surface.

At zero angle of attack and lower ambient pressure (81.68 N/m^2), the HRSI plugs are expected to see smaller values of heating compared to those at the lower elevation with the larger dynamic pressures (Figure 20). Furthermore, Figure 20 reveals that the front fillet ramp helps shield the fillet from the wing shock. It appears that a shock forms at the front fillet ramp. This shock merges with the nearby wing shock and the flow through the expansion corner, thereby decreasing its "heating strength". However, the plugs closer to the expansion corner should still see the higher heating which is somewhat visible in the values for plugs 3, 4, 6, and 7. Plugs 8-10 also see high heating due to their close proximity to the compression corner created by the bottom fillet edge and the cylindrical fuselage. As for the "Art Disp" case, it is seen that the diffusing effect results in the high pressure region created by the front fillet ramp and wing shock intersection being felt more strongly by nearby plugs 4, 6, and 7.

Mach 6.67 $\alpha = 0°$ Flight 2

Since the flight geometry is similar to that for the first flight, it is expected that the flowfield would also be similar. One particular difference is that the ambient pressure is 190.37 N/m^2 for this flight condition. Thus, the heat flux values in this situation are expected to be slightly higher than those on the first flight at the same Mach number (Figure 21). As in the first flight, HRSI plugs 7-10 all show high heating. Also, due to the low pressure region near the wing root and fillet intersection (via the expansion fan and the lack of angle of attack), plugs 1-3 and 5 show lower heating. The lower pressure in the corner may also be another factor in the wing shock being "sucked" toward the body thereby allowing its presence to be felt by the lower row of HRSI plugs.

Mach 5.0 $\alpha = 0.5°$ Flight 2

The relatively small angle of attack makes the shock geometry and flowfield similar to the Mach 6.67 cases of the two flights. However, the heating is stronger due to the larger ambient pressure (Figure 22). Note that the slight angle of attack has shifted the point of highest heating to the lower row of plugs (particularly plugs 9 and 10) due to the compression region created by the bottom fillet edge.

The neglect of ablation clearly has an effect as can be seen in the heat flux plots. At the lower Mach number cases, a consistent trend is not visible due to ablation being less of a factor (or maybe not yet present). At the higher Mach numbers where ablation is prominent, the measured heat fluxes are considerably lower than the calculated ones. Again, this is because the measured heat fluxes do not include the energy being used to ablate the surface or the effect of shielding of the surface by the ablation products.

The calculated heat fluxes agree rather well with flight data. It can be seen that the Engquist filter does sharpen gradients. An overall comparison of the "Low Disp" case to the "Fltrd" case reveals the latter results in higher heat fluxes due to the filter steepening the temperature gradients near the fillet surface (Figure 23). Comparisons of the "Fltrd" to the "Art Disp" cases result in the realization that the filter is successful in removing the diffusion caused by the large values of artificial dissipation.

In terms of accuracy, the "Fltrd" simulations result in only slightly lower error with respect to the flight data as compared to the highly dissipative cases ("Art Disp"). This is concluded from evaluations of the overall L2-norm with respect to flight data. The "Fltrd" solutions (0.178 L2-norm) result in more accurate heat flux predictions than its "Art Disp" counterpart (0.180 L2-norm). However, the most accurate of the three is the "Low Disp" case (overall L2-norm of 0.123). It is also the cheapest computationally since the "Fltrd" case results in 17% more CPU time per iteration than the "Low Disp" case.

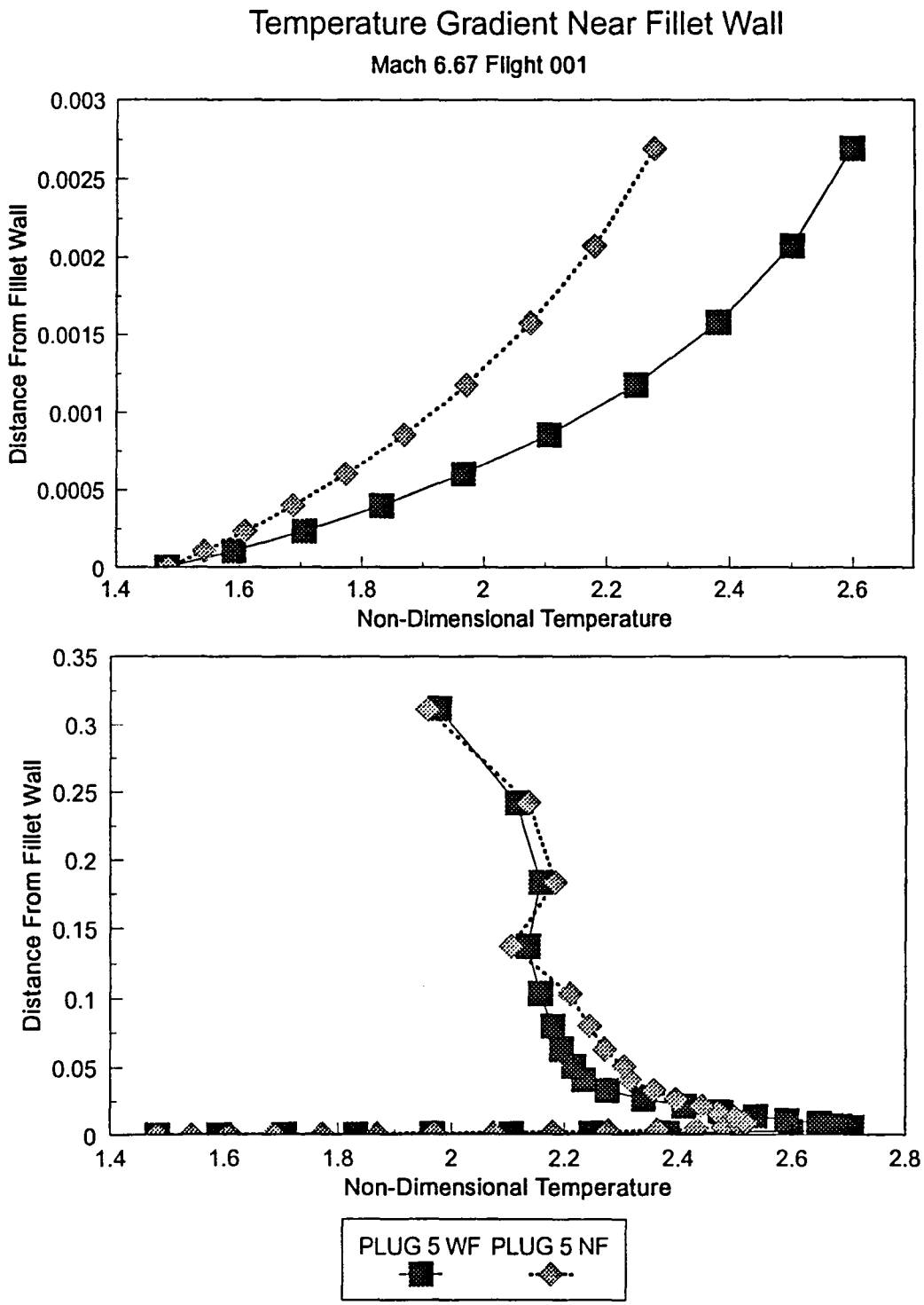
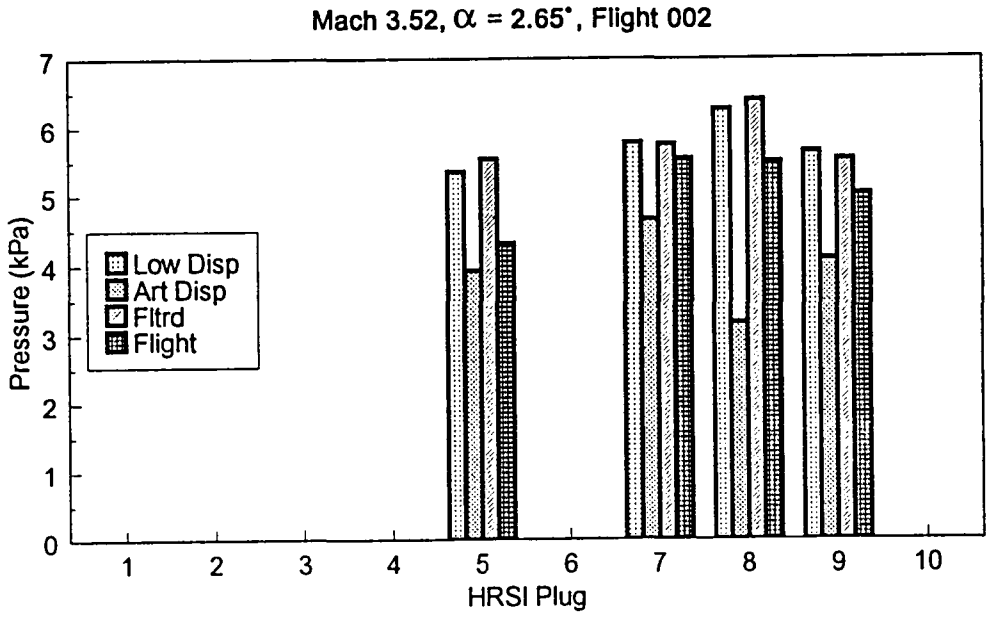

Temperature Gradient Near Fillet Wall
Mach 6.67 Flight 001
Distance From Fillet Wall
0.003
0.0025
0.002
0.0015
0.001
0.0005
0
1.4
1.6
1.8
2
2.2
2.4
2.6
Non-Dimensional Temperature
Distance From Fillet Wall
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
1.4
1.6
1.8
2
2.2
2.4
2.6
2.8
Non-Dimensional Temperature
PLUG 5 WF
PLUG 5 NF

Figure 23

Pegasus Flights: Pressure Coefficients

As already stated, the pressure data are sparse. However, it is still possible to get an indication of how well the code and the filter perform from the little data that are available. Looking at Figure 24 will give a brief overview of the results. Treating the flight data to be the reference point, the L2-error can be derived for each different simulation of the two flight conditions. For Mach 3.52, the accuracy follows the same general trend as in the heat flux cases. The "Art Disp" case shows the largest L2-error of 0.125. For this flow variable, the "Fltrd" solution improves the accuracy over the "Art Disp" case by posting an L2-error of 0.087. The best accuracy, however, still belongs to the "Low Disp" case with an L2-error of 0.077. This scenario is similarly found in the Mach 5.0 case where the L2-errors for the "Art Disp", "Fltrd", and "Low Disp" are 0.166, 0.116, and 0.110, respectively. Thus, although the filter is an improvement over the "Art Disp" case, its computational expense is not worth the cost since running a "Low Disp" case results in a cheaper and more accurate simulation.

Pressure at HRSI Plugs



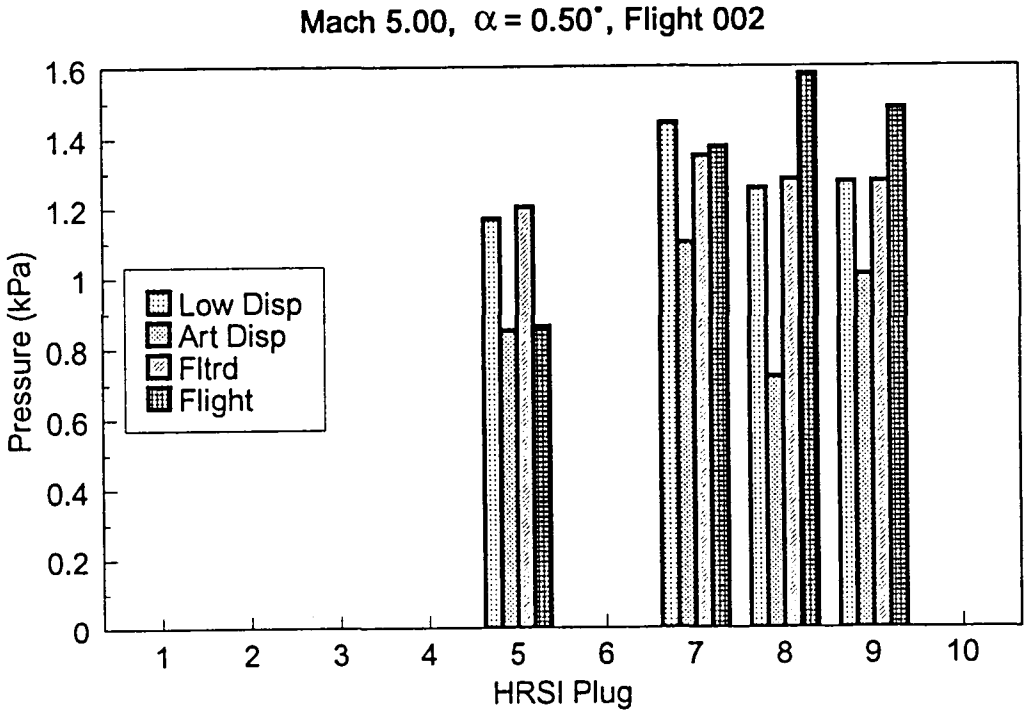


Figure 24

Chapter VII

Conclusions and Future Work

Improving accuracy and minimizing computational time led to utilization of the Engquist filter in various codes. It is found that for an explicit MacCormack scheme, the filter is successful in achieving the above goal with its results being almost identical to exact solutions. As for CPU time minimization, the ability of the filter to increase the stability threshold allows faster initial convergence and hence an overall CPU reduction.

The Engquist filter used in the implicit PARC3D code sharpens the gradients yielding more accurate heat flux estimations than simulations with large values of artificial dissipation. However, as expected, it results in longer computational times per iteration. It is found that for problems with large gradients and a moderately uniform grid (e.g. strong shock impingement problem), the filter is able to improve solution accuracy and CPU time to convergence. However, for nonuniform grids as found in the Pegasus simulations, it is found that without the filter and at lower dissipation values, the accuracy of the simulations increases while the required amount of computational time to convergence decreases. These suggest that PARC3D is a feasible tool for heat flux estimation on flight vehicles and that implementation of the Engquist filter could help in terms of accuracy and time-to-convergence for certain

problems. Furthermore, it must be noted that careful consideration must be given to how artificial viscosity is used.

The assumptions made in the numerical simulations need to be further investigated. For one, neglecting ablation must be looked into since it clearly affects the flight data at the higher Mach numbers. This requires a coupling of the analysis for the fluid side to the solid side (the vehicle thermal protection system). This leads to a very complicated situation due to the great disparity between characteristic times in both domains. Also, another thermodynamic model for the gas, one that accounts for real gas effects, must be investigated. This will allow a better look into the effects of dissociation. Still another area of improvement is the turbulence model used. PARC3D uses an algebraic one in order to minimize computational time. Today with the increasing abilities of computers, it is of interest to investigate the implementation of a two-equation turbulence model.

Appendix A

Engquist Filter Programs

SUBROUTINE ENGQUIST(Q,IL,JL,KK)

This is the one-dimensional form of the Engquist filter. The three by one matrix of conservative variables, **Q**, is coded in three-dimensional form for ease of application to future multidimensional problems. The first index is for the x-nodes with a maximum of **IL** nodes. The next two indices are for the y- and z-nodes with maximums **JL** and **KK**, respectively. For this subroutine, **IL=160**, **JL=3**, and **KK=1**. The fourth index of the **Q** matrix refers to a conservative variable: 1 is for density, 2 is for the x-direction mass flux, and 3 is for the total energy per unit mass. The matrix **E** is comprised of the right eigenvectors of the Jacobian matrix. **DPU** is the matrix of Δu. The first index is associated with the position in the **Q** matrix (i.e. 1 is for the difference in densities between two node points). The second index of **DPU** refers to the local node point at which the difference is taken. There are four of them, from $\Delta_+ u_{j-2}$ to $\Delta_+ u_{j+1}$. **ALPHA** is the matrix of incremental jump magnitudes over each eigenvector. Its first index refers to its associated eigenvector while the second index refers to the location of the jump: 1 for j-½ and 2 for j+½. The first part of the program is to determine whether or not an extremum in any of the conservative variables exists at node j. If one is found then **ISCILA** is given the value of one. The eigenvectors are calculated using the Roe-averaged values. The **DPU** matrix is then filled. The larger of the two values of **ALPHA** for each associated conservative variable is determined, divided by two, and if this value is less than the smaller **ALPHA** value, is denoted **D** else the smaller **ALPHA** value is given to **D**. **D** is the incremental change that is denoted as δ in the theory section of the Engquist filter. The adjacent node point to be corrected is given to **JCORR**. The incremental changes are then added to the respective node points through the corrected **DPU** matrix.

```
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      INTEGER S
      DIMENSION E(3,3),DPU(3,4),ALPHA(3,2),Q(160,3,1,3)

      DO 10 J=3, IL-2
       ISCILA = 0
       DO 5 K=1, 3
        DPU1 = Q(J+1,2,1,K)-Q(J,2,1,K)
        DMU1 = Q(J,2,1,K)-Q(J-1,2,1,K)
        IF (DPU1*DMU1 .LT. 0.D0) ISCILA = 1
5      CONTINUE
```

```
      IF (ISCILA .EQ. 1) THEN
       RHO1 = Q(J,2,1,1)
       RHOP = Q(J+1,2,1,1)
       R = (RHOP/RHO1)**0.5
       U = (R*Q(J+1,2,1,2)/Q(J+1,2,1,1)+Q(J,2,1,2)/Q(J,2,1,1))
     &      /(R+1.D0)
       QQ = U**2
       QJP = (Q(J+1,2,1,2)**2)/RHOP**2
       QJ = (Q(J,2,1,2)**2)/RHO1**2
       HP = 1.4*Q(J+1,2,1,3)/RHOP-0.2*QJP
       H = 1.4*Q(J,2,1,3)/RHO1-0.2*QJ
       H = (R*HP+H)/(R+1.D0)
       A = SQRT(0.4*(H-0.5*QQ))

       E(1,1) = 1.D0
       E(2,1) = U-A
       E(3,1) = H-U*A
       E(1,2) = 1.D0
       E(2,2) = U
       E(3,2) = 0.5*QQ
       E(1,3) = 1.D0
       E(2,3) = U+A
       E(3,3) = H+U*A

       DO 30 M=1,3
        DPU(M,1)= Q(J-1,2,1,M)-Q(J-2,2,1,M)
        DPU(M,2)= Q(J,2,1,M)-Q(J-1,2,1,M)
        DPU(M,3)= Q(J+1,2,1,M)-Q(J,2,1,M)
        DPU(M,4)= Q(J+2,2,1,M)-Q(J+1,2,1,M)
30     CONTINUE
       DO 40 I=2, 3

* CALCULATE THE ALPHA'S AT J-1/2 AND J+1/2

        C1 = 0.4D0*(DPU(3,I)+0.5D0*QQ*DPU(1,I)-U*DPU(2,I))/A/A
        C2 = (DPU(2,I)-U*DPU(1,I))/A
        ALPHA(1,I-1) = 0.5D0*(C1-C2)
        ALPHA(2,I-1) = DPU(1,I)-C1
        ALPHA(3,I-1) = 0.5D0*(C1+C2)
40     CONTINUE

       DO 50 M=1, 3
```

```
      IF ((ALPHA(M,1)*ALPHA(M,2)) .LT. 0.0) THEN
       IF (DABS(ALPHA(M,2)) .LT. DABS(ALPHA(M,1))) THEN
        DP = DABS(ALPHA(M,1))
        DM = DABS(ALPHA(M,2))
        JCORR = J-1
       ELSE
        DP = DABS(ALPHA(M,2))
        DM = DABS(ALPHA(M,1))
        JCORR = J+1
       ENDIF
       D = DM
       IF ((DP/2.0) .LT. DM) D=DP/2.0
       S=1
       IF (ALPHA(M,1) .LT. 0.0) S=-1
       DO 60 MM=1, 3
        IF (JCORR .EQ. J-1) THEN
         DPU(MM,3)=DPU(MM,3)+S*D*E(MM,M)
         DPU(MM,2)=DPU(MM,2)-2.D0*S*D*E(MM,M)
         DPU(MM,1)=DPU(MM,1)+S*D*E(MM,M)
        ELSE
         DPU(MM,2)=DPU(MM,2)-S*D*E(MM,M)
         DPU(MM,4)=DPU(MM,4)-S*D*E(MM,M)
         DPU(MM,3)=DPU(MM,3)+2.D0*S*D*E(MM,M)
         ENDIF
60     CONTINUE
      ENDIF
50   CONTINUE
     DO 70 I=J-1, J+2
      DO 70 M=1, 3
       Q(I,2,1,M) = Q(I-1,2,1,M)+DPU(M,I-J+2)
70   CONTINUE
    ENDIF
10  CONTINUE

    RETURN
    END
```

SUBROUTINE ENGQUIST(Q,IL,JL,KK,ETAXX,ETAXY,ETAYX,ETAYY)

This is similar to the one-dimensional program. The only difference is that the normals of the cell areas are included. **ETAXX** is the x-component of the normal of the x-face of the cell area. **ETAXY** is the y-component of the normal of the x-face of the cell area. **ETAYY** is the y-component of the normal of the y-face of the cell area. **ETAYX** is the x-component of the normal of the y-face of the cell area. Note that for the 2-D problem the filter is only applied in the x-direction. Also, the first two components of **Q** is like that of the 1-D form but the third is the mass flux in the y-direction while the fourth is the total energy per unit mass. The rest of the program is similar to the 1-D program. Note that for this 2-D version, **IL=42**, **JL=22**, and **KK=1**.

```
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      INTEGER S
      DIMENSION EE(4,4),DPU(4,4),ALPHA(4,2),Q(42,22,1,4)
      DIMENSION ETAXX(IL,JL),ETAXY(IL,JL),ETAYX(IL,JL),ETAYY(IL,JL)

C FILTER IN THE X-DIRECTION.

      DO 10 IN=2, JL-1
      DO 10 J=3, IL-2
       ISCILA = 0
       DO 5 K=1, 4
        DPU1 = Q(J+1,IN,1,K)-Q(J,IN,1,K)
        DMU1 = Q(J,IN,1,K)-Q(J-1,IN,1,K)
        IF (DPU1*DMU1 .LT. 0.D0) ISCILA = 1
5      CONTINUE

       IF (ISCILA .EQ. 1) THEN
        RHO1 = Q(J,IN,1,1)
        RHOP = Q(J+1,IN,1,1)
        R = (RHOP/RHO1)**0.5
        U = (R*Q(J+1,IN,1,2)/Q(J+1,IN,1,1)+Q(J,IN,1,2)/Q(J,IN,1,1))
     &      /(R+1.D0)
        V = (R*Q(J+1,IN,1,3)/Q(J+1,IN,1,1)+Q(J,IN,1,3)/Q(J,IN,1,1))
     &      /(R+1.D0)
        QQ = U**2+V**2
        QJP = (Q(J+1,IN,1,2)**2)/RHOP**2+(Q(J+1,IN,1,3)**2)/RHOP**2
       QJ = (Q(J,IN,1,2)**2)/RHO1**2+(Q(J,IN,1,3)**2)/RHO1**2
        HP = 1.4*Q(J+1,IN,1,4)/RHOP-0.2*QJP
        H = 1.4*Q(J,IN,1,4)/RHO1-0.2*QJ
```

```
      H = (R*HP+H)/(R+1.D0)
      A = SQRT(0.4*(H-0.5*QQ))

      ETAX = ETAXX(J,IN)
      ETAY = ETAXY(J,IN)
      UBAR = U*ETAX + V*ETAY
      VBAR = V*ETAX - U*ETAY

      EE(1,1) = 1.D0
      EE(2,1) = U-ETAX*A
      EE(3,1) = V-ETAY*A
      EE(4,1) = H-UBAR*A

      EE(1,2) = 1.D0
      EE(2,2) = U-ETAY*A
      EE(3,2) = V+ETAX*A
      EE(4,2) = 0.5D0*QQ+VBAR*A

      EE(1,3) = 1.D0
      EE(2,3) = U+ETAY*A
      EE(3,3) = V-ETAX*A
      EE(4,3) = 0.5D0*QQ-VBAR*A

      EE(1,4) = 1.D0
      EE(2,4) = U+ETAX*A
      EE(3,4) = V+ETAY*A
      EE(4,4) = H+UBAR*A

      DO 30 M=1,4
       DPU(M,1)= Q(J-1,IN,1,M)-Q(J-2,IN,1,M)
       DPU(M,2)= Q(J,IN,1,M)-Q(J-1,IN,1,M)
       DPU(M,3)= Q(J+1,IN,1,M)-Q(J,IN,1,M)
       DPU(M,4)= Q(J+2,IN,1,M)-Q(J+1,IN,1,M)
30    CONTINUE

      DO 40 I=2, 3

* CALCULATE THE ALPHA'S AT J-1/2 AND J+1/2

       X1 = (ETAX-ETAY)*A
       X2 = -A*(ETAX+ETAY)
       X3 = -2.D0*ETAX*A
```

```
      Y2=X2/X1
      Y1=A*(ETAY-ETAX)-X2*Y2
      Y3=-EE(2,1)*Y2-EE(3,1)
      Y4=-2.D0*ETAY*A+X3*Y2

      ZZ1=A*(A/0.4D0-(UBAR+VBAR))
      ZZ2=A*(A/0.4D0+(VBAR-UBAR))
      Z1=2.D0*UBAR*A-(ZZ1*(Y2*Y4/Y1+X3/X1)+Y4/Y1*ZZ2)

      Z2=(Y2*ZZ1+ZZ2)/Y1
      Z3=ZZ1*(Y2*Y2/Y1+1.D0/X1)+Y2/Y1*ZZ2
      Z4=ZZ1*(Y2*Y3/Y1-EE(2,1)/X1)+Y3/Y1*ZZ2-EE(4,1)

      ALPHA(4,I-1)=(DPU(4,I)+DPU(3,I)*Z2+DPU(2,I)*Z3
     &             +DPU(1,I)*Z4)/Z1

      ALPHA(3,I-1)=(DPU(3,I)+DPU(2,I)*Y2+DPU(1,I)*Y3
     &             +Y4*ALPHA(4,I-1))/Y1

      ALPHA(2,I-1)=(DPU(2,I)-EE(2,1)*DPU(1,I)+X2*ALPHA(3,I-1)
     &             +X3*ALPHA(4,I-1))/X1

      ALPHA(1,I-1)=DPU(1,I)-ALPHA(2,I-1)-ALPHA(3,I-1)
     &             -ALPHA(4,I-1)
40    CONTINUE

      DO 50 M=1, 4
      IF ((ALPHA(M,1)*ALPHA(M,2)) .LT. 0.0) THEN
       IF (DABS(ALPHA(M,2)) .LT. DABS(ALPHA(M,1))) THEN
        DP = DABS(ALPHA(M,1))
        DM = DABS(ALPHA(M,2))
        JCORR = J-1
       ELSE
        DP = DABS(ALPHA(M,2))
        DM = DABS(ALPHA(M,1))
        JCORR = J+1
       ENDIF
       D = DM
       IF ((DP/2.0) .LT. DM) D=DP/2.0
       S=1
       IF (ALPHA(M,1) .LT. 0.0) S=-1
       DO 60 MM=1, 4
```

```
        IF (JCORR .EQ. J-1) THEN
         DPU(MM,3)=DPU(MM,3)+S*D*EE(MM,M)
         DPU(MM,2)=DPU(MM,2)-2.D0*S*D*EE(MM,M)
         DPU(MM,1)=DPU(MM,1)+S*D*EE(MM,M)
        ELSE
         DPU(MM,2)=DPU(MM,2)-S*D*EE(MM,M)
         DPU(MM,4)=DPU(MM,4)-S*D*EE(MM,M)
         DPU(MM,3)=DPU(MM,3)+2.D0*S*D*EE(MM,M)

        ENDIF
60      CONTINUE
       ENDIF
50     CONTINUE
      DO 70 I=J-1, J+2
       DO 70 M=1, 4
        Q(I,IN,1,M) = Q(I-1,IN,1,M)+DPU(M,I-J+2)
70     CONTINUE
     ENDIF
10   CONTINUE
```

SUBROUTINE ENGQST

The three-dimensional form of the Engquist filter uses the cell volume normals. These are found in the variables **XX** (x-component of the x-face normal), **XY** (y-component of the x-face normal), and **XZ** (z-component of the x-face normal). The other faces are similarly described in the variables **YX, YY, YZ, ZX, ZY**, and **ZZ**. **GAMMA** is the constant ratio of specific heats (1.4). **GAMI=GAMMA-1**. The calculation of **ALPHA** is now done via a Gaussian elimination program called through the subroutines **FACTOR** and **SOLVE**, which both make use of **NPIVOT** and **IER**. Note that the matrices **AM** and **B** are just dummy matrices used in the Gaussian elimination program. The right eigenvectors are stored in the matrix **EE**. The **Q** matrix is composed of: 1) density, 2) x-direction mass flux, 3) y-direction mass flux, 4) z-direction mass flux, and 5) total energy per unit mass. The rest of the Engquist program is similar to its one-dimensional form with **IL**, **JL**, and **KK** replaced by **NX**, **NY**, and **NZ**, respectively. As done in the two-dimensional version, the filter is applied only in the x-direction.

```
      IMPLICIT REAL*8 (A-H,O-Z)
      PARAMETER (NX=82,NY=83,NZ=51,NM=83)
      REAL*8 EE(5,5),DPU(5,4),ALPHA(5,2),AM(5,5),B(5)
      INTEGER S,NPIVOT(5),IER
      COMMON/HRAT/ GAMMA,GAMI,GGM1,GSGM
      COMMON/VARS/ Q(NX,NY,NZ,6)
      COMMON/VAR3/XX(NX,NY,NZ),XY(NX,NY,NZ),XZ(NX,NY,NZ),
     *            YX(NX,NY,NZ),YY(NX,NY,NZ),YZ(NX,NY,NZ),
     *            ZX(NX,NY,NZ),ZY(NX,NY,NZ),ZZ(NX,NY,NZ)

      DO 10 IN=2, NY-1
      DO 10 K=2, NZ-1
      DO 10 J=3, NX-2
       ISCILA = 0
       DO 5 L=1, 5
        DPU1 = Q(J+1,IN,K,L)*Q(J+1,IN,K,6)-Q(J,IN,K,L)*Q(J,IN,K,6)
        DMU1 = Q(J,IN,K,L)*Q(J,IN,K,6)-Q(J-1,IN,K,L)*Q(J-1,IN,K,6)
        IF (DPU1*DMU1 .LT. 0.D0) ISCILA = 1
5      CONTINUE

       IF (ISCILA .EQ. 1) THEN
        BT = 1./SQRT(2.)
        R1 = XX(J,IN,K)
        R2 = XY(J,IN,K)
        R3 = XZ(J,IN,K)
```

```
      RR123 = 1./SQRT(R1**2+R2**2+R3**2)
      R1 = R1*RR123
      R2 = R2*RR123
      R3 = R3*RR123

      RHO1 = Q(J,IN,K,1)*Q(J,IN,K,6)
      RHOP = Q(J+1,IN,K,1)*Q(J+1,IN,K,6)
      RA = (RHOP/RHO1)**0.5
      RHO2=(RHOP*RHO1)**0.5
      U = (RA*Q(J+1,IN,K,2)/Q(J+1,IN,K,1)+Q(J,IN,K,2)
     &     /Q(J,IN,K,1))/(RA+1.D0)
      V = (RA*Q(J+1,IN,K,3)/Q(J+1,IN,K,1)+Q(J,IN,K,3)
     &     /Q(J,IN,K,1))/(RA+1.D0)
      W = (RA*Q(J+1,IN,K,4)/Q(J+1,IN,K,1)+Q(J,IN,K,4)
     &     /Q(J,IN,K,1))/(RA+1.D0)
      ET= (RA*Q(J+1,IN,K,5)*Q(J+1,IN,K,6)+
     &     Q(J,IN,K,5)*Q(J,IN,K,6))/(RA+1.D0)

      RR=1./RHO2
      R=(RHO1*RHOP)**0.5
      UVW=.5D0*(U**2+V**2+W**2)
      PP=GAMI*(ET-RHO2*UVW)
      AA=GAMMA*PP*RR
      AA=SQRT(ABS(AA))
      CSR=1./AA
      BTRO=BT*R
      AA2=AA/GAMI
      C1=BTRO*CSR
      C2=R*R1
      C3=R*R2
      C4=R*R3
      C8=BTRO*R1
      C9=BTRO*R2
      C10=BTRO*R3
      C11=BTRO*AA2
      C12=U*C8 + V*C9 + W*C10
      C13=UVW*C1 + C11

      EE(1,1)  = R1
      EE(1,2)  = R2
      EE(1,3)  = R3
      EE(1,4)  = C1
```

```
      EE(1,5)  = C1
      EE(2,1)  = U*R1
      EE(2,2)  = U*R2 - C4
      EE(2,3)  = U*R3 + C3
      EE(2,4)  = C1*U + C8
      EE(2,5)  = C1*U - C8
      EE(3,1)  = V*R1 + C4
      EE(3,2)  = V*R2
      EE(3,3)  = V*R3 - C2
      EE(3,4)  = C1*V + C9
      EE(3,5)  = C1*V - C9
      EE(4,1)  = W*R1 - C3
      EE(4,2)  = W*R2 + C2
      EE(4,3)  = W*R3
      EE(4,4)  = C1*W + C10
      EE(4,5)  = C1*W - C10
      EE(5,1)  = UVW*R1 + V*C4 - W*C3
      EE(5,2)  = UVW*R2 + W*C2 - U*C4
      EE(5,3)  = UVW*R3 + U*C3 - V*C2
      EE(5,4)  = C13 + C12
      EE(5,5)  = C13 - C12

      DO 30 M=1,5
       DPU(M,1)= Q(J-1,IN,K,M)*Q(J-1,IN,K,6) -
     &           Q(J-2,IN,K,M)*Q(J-2,IN,K,6)
       DPU(M,2)= Q(J,IN,K,M)*Q(J,IN,K,6) -
     &           Q(J-1,IN,K,M)*Q(J-1,IN,K,6)
       DPU(M,3)= Q(J+1,IN,K,M)*Q(J+1,IN,K,6) -
     &           Q(J,IN,K,M)*Q(J,IN,K,6)
       DPU(M,4)= Q(J+2,IN,K,M)*Q(J+2,IN,K,6) -
     &           Q(J+1,IN,K,M)*Q(J+1,IN,K,6)
30     CONTINUE

      DO 45 I=2, 3
       DO 42 II=1,5
        DO 41 JJ=1,5
         AM(II,JJ)=EE(II,JJ)
41      CONTINUE
        B(II)=DPU(II,I)
42      CONTINUE
       CALL FACTOR(AM,NPIVOT,IER)
       IF (IER .EQ. 1) THEN
```

```
      WRITE(21,*) 'X',J,IN,K
      WRITE(21,*) 'IER = 1, THEREFORE THE MATRIX IS SINGULAR'
      GOTO 45
     ENDIF
     CALL SOLVE(AM,B,NPIVOT)
     DO 43 II=1, 5
      ALPHA(II,I-1) = B(II)
43    CONTINUE
45   CONTINUE

      DO 50 M=1, 5
       IF ((ALPHA(M,1)*ALPHA(M,2)) .LT. 0.0) THEN
        IF (DABS(ALPHA(M,2)) .LT. DABS(ALPHA(M,1))) THEN
         DP = DABS(ALPHA(M,1))
         DM = DABS(ALPHA(M,2))
         JCORR = J-1
        ELSE
         DP = DABS(ALPHA(M,2))
         DM = DABS(ALPHA(M,1))
         JCORR = J+1
        ENDIF
        D = DM
        IF ((DP/2.0) .LT. DM) D=DP/2.0
        S=1
        IF (ALPHA(M,1) .LT. 0.0) S=-1
        DO 60 MM=1, 5
         IF (JCORR .EQ. J-1) THEN
          DPU(MM,3)=DPU(MM,3)+S*D*EE(MM,M)
          DPU(MM,2)=DPU(MM,2)-2.D0*S*D*EE(MM,M)
          DPU(MM,1)=DPU(MM,1)+S*D*EE(MM,M)
         ELSE
          DPU(MM,2)=DPU(MM,2)-S*D*EE(MM,M)
          DPU(MM,4)=DPU(MM,4)-S*D*EE(MM,M)
          DPU(MM,3)=DPU(MM,3)+2.D0*S*D*EE(MM,M)
         ENDIF
60       CONTINUE
       ENDIF
50     CONTINUE

      DO 70 I=J-1, J+2
       DO 70 M=1, 5
        Q(I,IN,K,M)=(Q(I-1,IN,K,M)*Q(I-1,IN,K,6)+DPU(M,I-J+2))/
```

```
     &              Q(I,IN,K,6)
70     CONTINUE
      ENDIF
10    CONTINUE
     RETURN
     END


C THIS IS PART OF THE GAUSSIAN PROGRAM.
C THIS SUBROUTINE DOES THE LU DECOMPOSITION OF MATRIX A

     SUBROUTINE FACTOR(AM,NPIVOT,IER)
     INTEGER NPIVOT(5),IER
     REAL*8 CMAX,CK,AM(5,5),TEMP

C IER = 0 IF THE INVERSE OF MATRIX A EXISTS, OTHERWISE IER = 1.
     IER = 0
     NPIVOT(5)=5
     DO 100 K=1, 4

C FIND MAXIMUM VALUE IN EACH COLUMN
       NPIVOT(K)=K
       CMAX = DABS(AM(K,K))
       M = K
       DO 10 I=K, 4
         CK = DABS(AM(I+1,K))
         IF (CK .GT. CMAX) THEN
           CMAX = CK
           M = I + 1
         ENDIF
10      CONTINUE

C CHECK WHETHER OR NOT MATRIX IS SINGULAR.  IF IT IS, THERE
C IS NO INVERSE FOR THE MATRIX, THE SOLUTION DOES NOT EXIST.

       IF (CMAX .EQ. 0) THEN
         IER = 1
         GOTO 105
       ENDIF

C INTERCHANGE ROW M AND ROW K IF NECESSARY
```

```
      IF (M .EQ. K) THEN
        GOTO 25
      ELSE

C CHANGE THE VECTOR NPIVOT TO RECORD THE PIVOT CHANGES

        NPIVOT(K) = M
        DO 20 J=K, 5
          TEMP = AM(K,J)
          AM(K,J) = AM(M,J)
          AM(M,J) = TEMP
20       CONTINUE
      ENDIF

25    DO 40 I=K+1, 5

C FIND THE MULTIPLIERS OF THE ROWS AND STORE IT IN MATRIX A

      AM(I,K) = AM(I,K)/AM(K,K)

C MULTIPLY THE KTH ROW BY A(I,K) AND SUBTRACT THE RESULT

C FROM THE ITH ROW
      DO 30 J=K+1, 5
        AM(I,J) = AM(I,J) - AM(I,K)*AM(K,J)
30     CONTINUE
40    CONTINUE
100   CONTINUE
105   RETURN
110   END

C THIS SUBROUTINE SOLVES LU*X = B IN TWO STEPS:
C 1) FORWARD ELIMINATION
C 2) BACKWARD SUBSTITUTION

     SUBROUTINE SOLVE(AM,B,NPIVOT)
     INTEGER NPIVOT(5)
     REAL*8 AM(5,5),B(5),TEMP,SUM

C BEGIN THE FORWARD ELIMINATION BY SOLVING L*Z=P*B

     DO 20 K=1, 4
```

```
C CHECK IF THERE IS A ROW INTERCHANGE
      IF (NPIVOT(K) .EQ. K) THEN
        GOTO 5
      ELSE
        KK = NPIVOT(K)
        TEMP = B(K)
        B(K) = B(KK)
        B(KK) = TEMP
      ENDIF

C CALCULATE THE NEW B-VECTOR (AKA Z-VECTOR OF L*Z = B)

5       DO 10 I = K+1, 5
          B(I) = B(I) - AM(I,K)*B(K)
10      CONTINUE
20    CONTINUE

C BEGIN BACKWARD SUBSTITUTION; CALCULATE THE X-VECTOR IN
U*X = Z
     B(5) = B(5)/AM(5,5)
     DO 30 I = 4, 1, -1
       SUM = 0.D0
       DO 25 J=I+1, 5
         SUM = AM(I,J)*B(J) + SUM
25      CONTINUE
       B(I) = (B(I) - SUM)/AM(I,I)
30    CONTINUE
     RETURN
40    END
```

Appendix B

Shock-Boundary Layer Interaction Convergence Histories

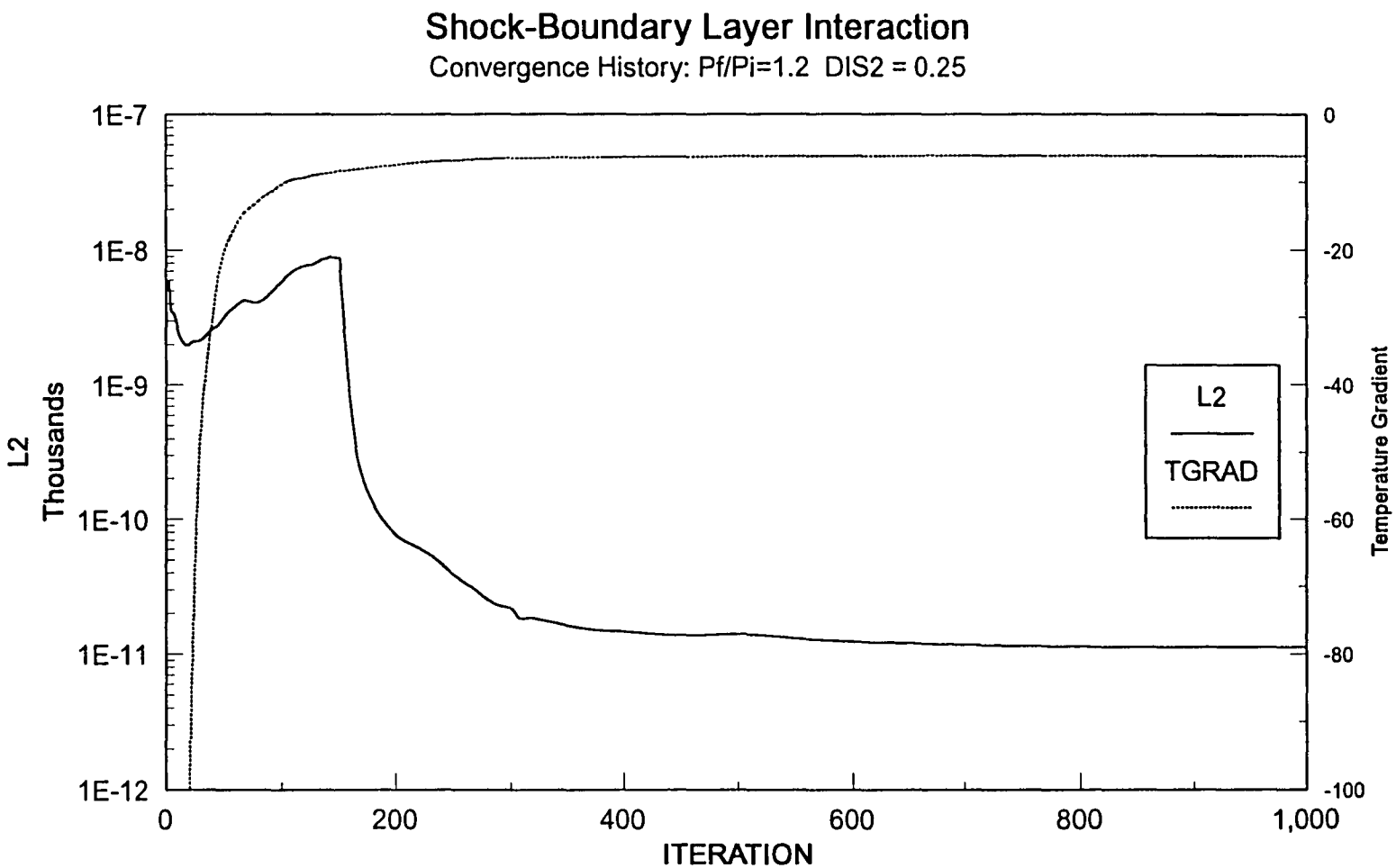

Shock-Boundary Layer Interaction
Convergence History: Pf/Pi=1.2 DIS2 = 0.25
L2
Thousands
1E-7
1E-8
1E-9
1E-10
1E-11
1E-12
0
200
400
600
800
1,000
ITERATION
0
-20
-40
-60
-80
-100
Temperature Gradient
L2
TGRAD

Figure 25a

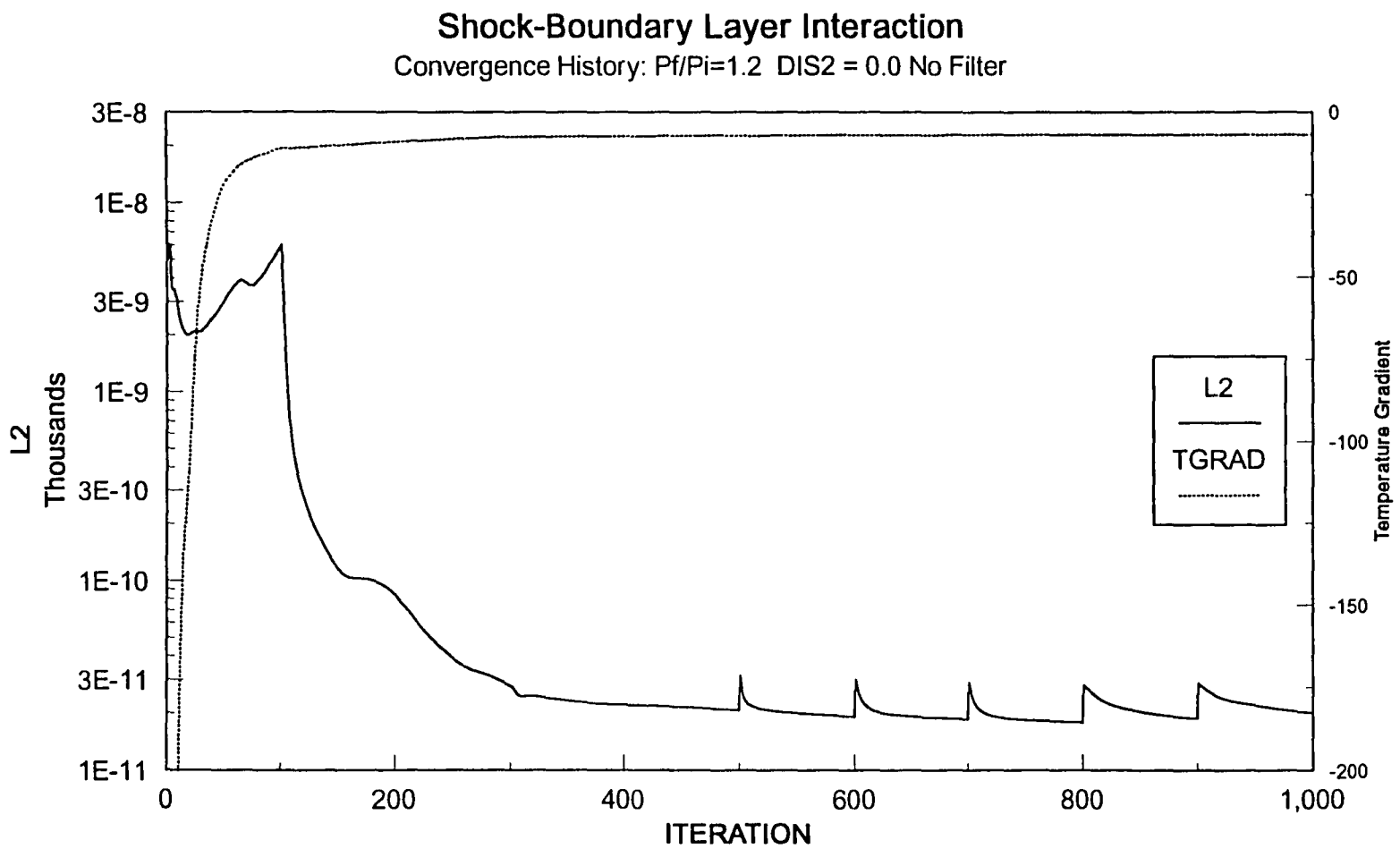

Shock-Boundary Layer Interaction
Convergence History: Pf/Pi=1.2 DIS2 = 0.0 No Filter
L2
Thousands
3E-8
1E-8
3E-9
1E-9
3E-10
1E-10
3E-11
1E-11
0
200
400
600
800
1,000
ITERATION
0
-50
-100
-150
-200
Temperature Gradient
L2
TGRAD

Figure 25b

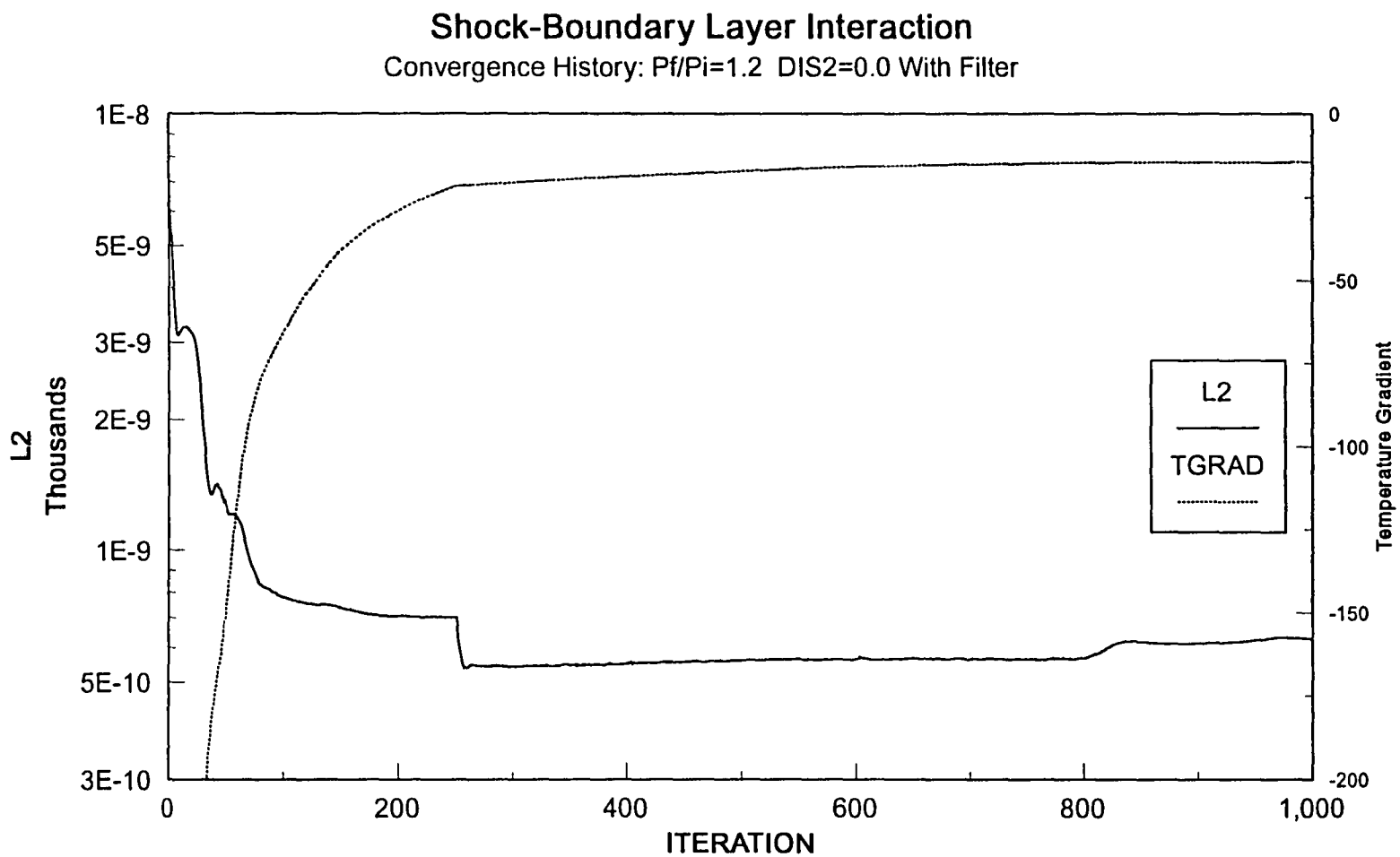

Shock-Boundary Layer Interaction
Convergence History: Pf/Pi=1.2 DIS2=0.0 With Filter
L2
Thousands
1E-8
5E-9
3E-9
2E-9
1E-9
5E-10
3E-10
0
200
400
600
800
1,000
ITERATION
0
-50
-100
-150
-200
Temperature Gradient
L2
TGRAD

Figure 25c

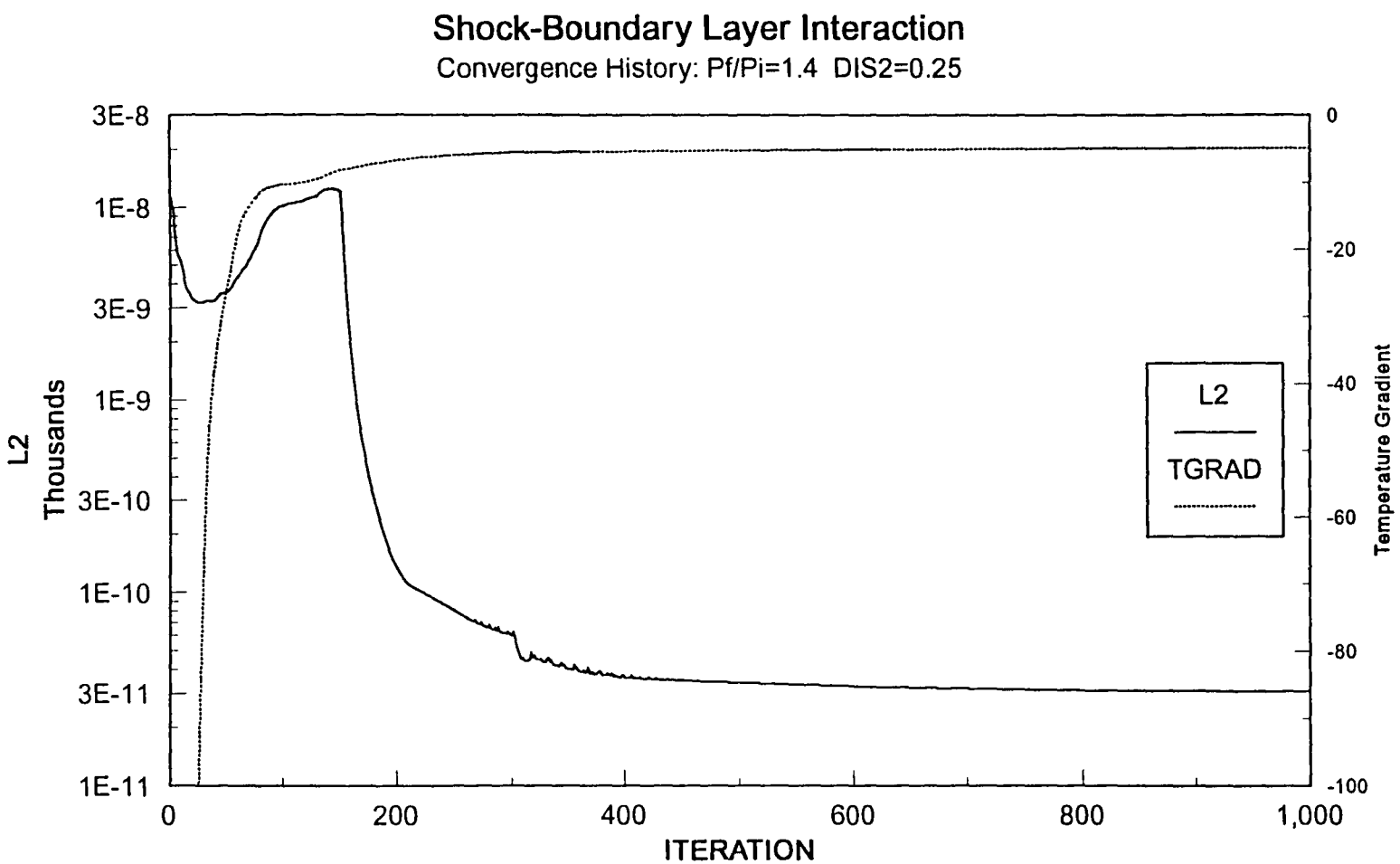

Shock-Boundary Layer Interaction
Convergence History: Pf/Pi=1.4 DIS2=0.25
L2
Thousands
3E-8
1E-8
3E-9
1E-9
3E-10
1E-10
3E-11
1E-11
0
200
400
600
800
1,000
ITERATION
0
-20
-40
-60
-80
-100
Temperature Gradient
L2
TGRAD

Figure 26a

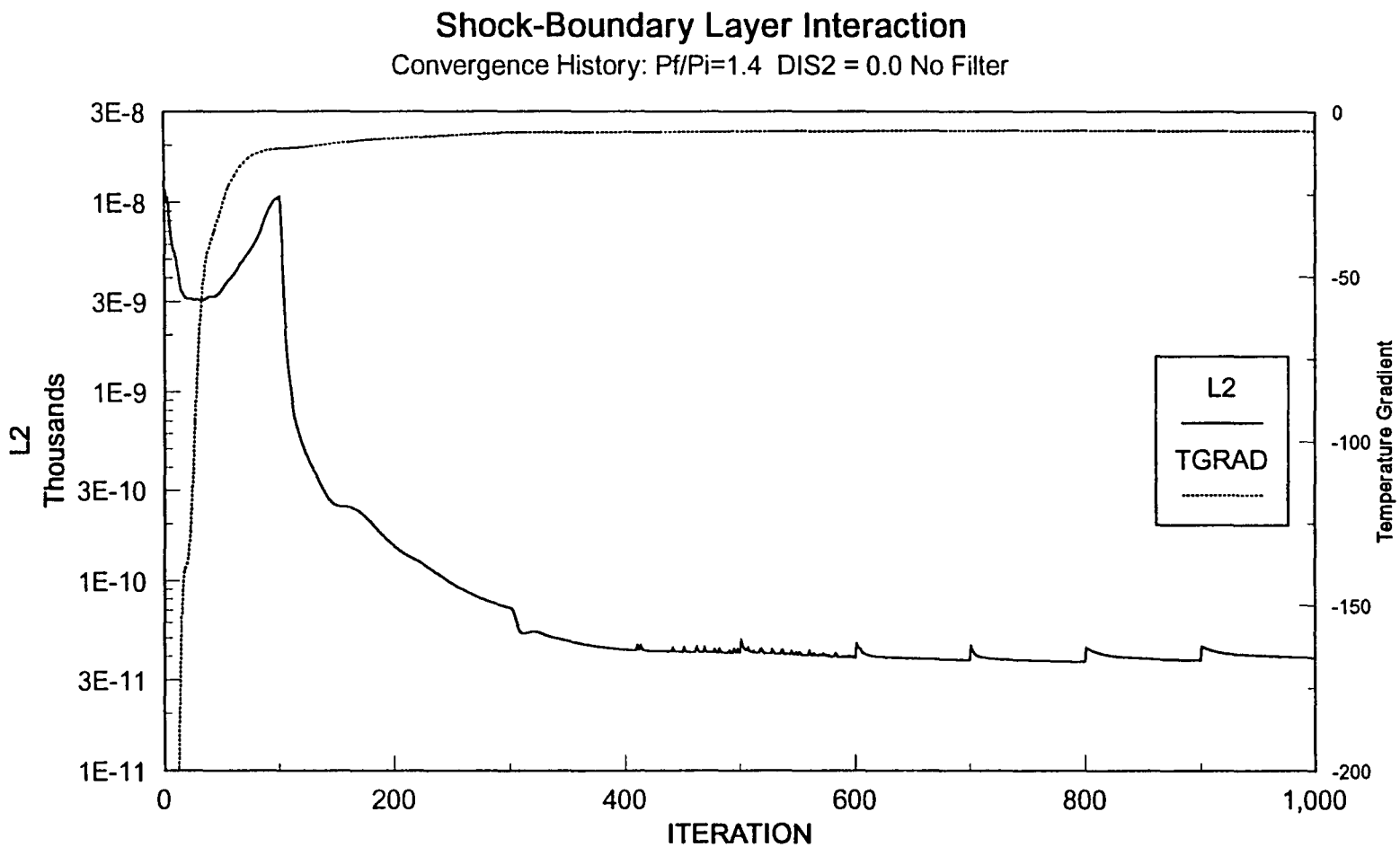

Shock-Boundary Layer Interaction
Convergence History: Pf/Pi=1.4 DIS2 = 0.0 No Filter
L2
Thousands
3E-8
1E-8
3E-9
1E-9
3E-10
1E-10
3E-11
1E-11
0
200
400
600
800
1,000
ITERATION
0
-50
-100
-150
-200
Temperature Gradient
L2
TGRAD

Figure 26b

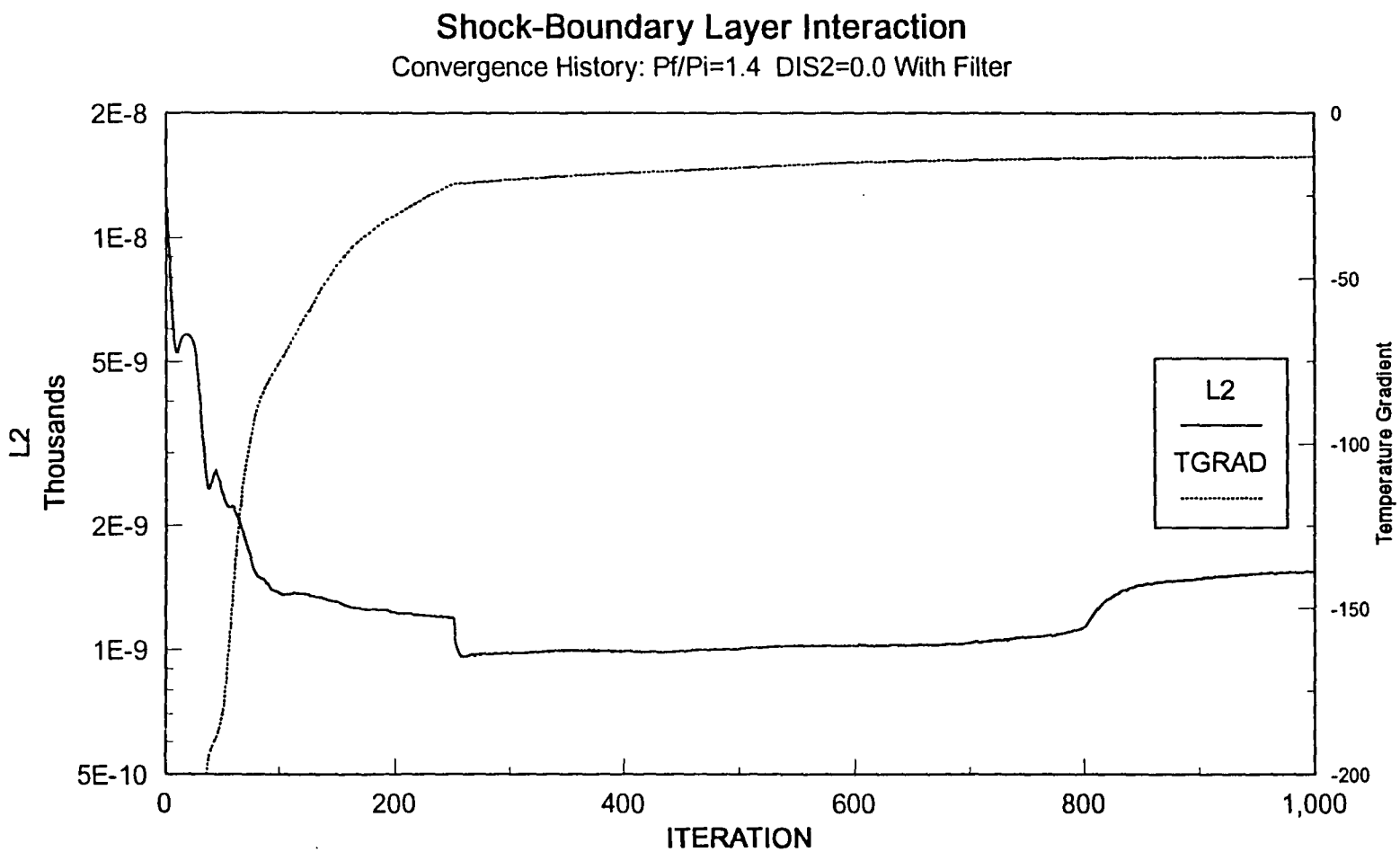

Shock-Boundary Layer Interaction
Convergence History: Pf/Pi=1.4 DIS2=0.0 With Filter
2E-8
1E-8
5E-9
2E-9
1E-9
5E-10
L2
Thousands
0
200
400
600
800
1,000
ITERATION
0
-50
-100
-150
-200
Temperature Gradient
L2
TGRAD

Figure 26c

References

Engquist, B., Lotstedt, P., and Sjogreen, B., "Nonlinear Filters for Efficient Shock Computation", Mathematics of Computation, Vol. 52, No. 186, pp. 509-537, April 1989.

Roe, P.L., "Approximate Riemann Solvers, Parameter Vectors, and Difference Schemes", Journal of Computational Physics, Vol. 43, pp. 357-372, 1981.

Anderson, D.A., Tannehill, J.C., and Pletcher, R.H., **Computational Fluid Mechanics and Heat Transfer**, Hemisphere, New York, 1984.

Mendenhall, M.R., Lesieutre, D.J., Caruso, S.C., Dillenius, M.F.E., Kuhn, G.D., "Aerodynamic Design of Pegasus™: Concept to Flight with CFD", Missile Aerodynamics, AGARD-CP-493, 1990.

Kuhn, G.D., "Post-Flight Aerothermal Analysis of Pegasus™ Using CFD Techniques", NEAR TR 423, Nielsen Engineering and Research, Inc., June 1991.

Noffz, G.K., Curry, R.E., Haering, E.A. Jr., and Kolodziej, P., "Aerothermal Test Results From the First Flight of the Pegasus Air-Launched Space Booster", NASA Technical Memorandum 4330, October 1991.

Fricker, D., Mendoza, J., and Catton, I., "A Computational Fluid Dynamics Analysis of the Hypersonic Flights of the Pegasus Air-Launched Space Booster", NASA CR 186023, August 1992.

Hakkinen, R.J., Greber, I., Trilling, L., and Abarbanel, S.S., "The Interaction of an Oblique Shock Wave With a Laminar Boundary Layer", NASA TM 2-18-59W, 1959.

MacCormack, R.W., "Numerical Solution of the Interaction of a Shock Wave With a Laminar Boundary Layer", Lecture Notes in Physics: Proceedings of the Second International Conference on Numerical Methods in Fluid Dynamics, Vol. 8, pp. 151-163, Springer-Verlag, 1971.

Jameson, A., Schmidt, W., and Turkel, E., "Numerical Solutions of the Euler Equations by Finite Volume Methods Using Runge-Kutta Time Stepping Schemes", AIAA Paper No. 81-1259, AIAA 14th Fluid and Plasma Dynamics Conference, Palo Alto, California, 1981.

Cooper, G.K. and Sirbaugh, J.R., **PARC Code: Theory and Usage**, AEDC-TR-89-15, Arnold Engineering Development Center, April 1989.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (Leave blank)

2. REPORT DATE
September 1995

3. REPORT TYPE AND DATES COVERED
Contractor Report

4. TITLE AND SUBTITLE
A Computational Fluid Dynamics Simulation of the Hypersonic Flight of the Pegasus™ Vehicle Using an Artificial Viscosity Model and a Nonlinear Filtering Method

5. FUNDING NUMBERS
WU 466-01

6. AUTHOR(S)
John Cadiz Mendoza

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
University of California, Los Angeles
Los Angeles, California

8. PERFORMING ORGANIZATION REPORT NUMBER
H-2071

9. SPONSORING/MONOTORING AGENCY NAME(S) AND ADDRESS(ES)
National Aeronautics and Space Administration
Dryden Flight Research Center
P.O. Box 273
Edwards, California 93523-0273

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
NASA CR-186033

11. SUPPLEMENTARY NOTES
Dryden Technical Monitor: Robert Curry. The information presented in this report was offered as a thesis in partial fulfillment of the requirements for the degree of Master of Science in Mechanical Engineering, University of California, Los Angeles, Los Angeles, California. This work was conducted under contract number FDP/NASA/A NCC-2-374.

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Unclassified—Unlimited
Subject Category 02

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

The computational fluid dynamics code, PARC3D, is tested to see if its use of non-physical artificial dissipation affects the accuracy of its results. This is accomplished by simulating a shock-laminar boundary layer interaction and several hypersonic flight conditions of the Pegasus™ launch vehicle using full artificial dissipation, low artificial dissipation, and the Engquist filter. Before the filter is applied to the PARC3D code, it is validated in one-dimensional and two-dimensional form in a MacCormack scheme against the Riemann and convergent duct problem. For this explicit scheme, the filter shows great improvements in accuracy and computational time as opposed to the nonfiltered solutions. However, for the implicit PARC3D code it is found that the best estimate of the Pegasus experimental heat fluxes and surface pressures is the simulation utilizing low artificial dissipation and no filter. The filter does improve accuracy over the artificially dissipative case but at a computational expense greater than that achieved by the low artificial dissipation case which has no computational time penalty and shows better results. For the shock-boundary layer simulation, the filter does will in terms of accuracy for a strong impingement shock but not as well for weaker shock strengths. Furthermore, for the latter problem the filter reduces the required computational time to convergence by 18.7%.

14. SUBJECT TERMS
Computational fluid dynamics, Engquist filter, Heat flux, Hypersonic aerodynamics, Pegasus

15. NUMBER OF PAGES
123

16. PRICE CODE
A06

17. SECURITY CLASSIFICATION OF REPORT
Unclassified

18. SECURITY CLASSIFICATION OF THIS PAGE
Unclassified

19. SECURITY CLASSIFICATION OF ABSTRACT
Unclassified

20. LIMITATION OF ABSTRACT
Unlimited

NSN 7540-01-280-5500

Available from the NASA Center for AeroSpace Information, 800 Elkridge Landing Road, Linthicum Heights, MD 21090; (301)621-0390

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102

End of Document