CHAPTER 8.0 REFERENCES

1 MacCormack, R. W., "The Effect of Viscosity in Hypervelocity Impact Cratering". AIAA Paper No. 69-354, 1969.

2 Doorly, D. J., "A Study of the Effect of Wake-Passing on Turbine Blades". PhD Dissertation, OEUL Report 1515/84, Oxford England.

3 Ashworth, D. A., LaGraff, J. E., Schultz, D. L., and Grindrod, K. J., "Unsteady Aerodynamic and Heat Transfer Processes in a Transonic Turbine Stage". ASME Journal of Turbomachinery, 87-GT-197, 1987.

4 Doorly, D. J., "Modelling the Unsteady Flow in a Turbine Rotor Passage". ASME Journal of Turbomachinery, Volume 107, pp. 1022-1030, October 1985.

5 Johnson, A. B., Rigby, M., Oldfield, M. L. G., Ainsworth, R. and Oliver, M., "Surface Heat Transfer Fluctuations on a Turbine Rotor Blade due to Upstream Shock Wave Passing". ASME Journal of Turbomachinery, Volume 111, 88-GT-172, pp. 105-113, 1989.

6 Rigby, M. J., Johnson, A. B., Oldfield, M. L. G., and Jones, T. V., "Temperature Scaling of Turbine Blade Heat Transfer with and without Shock Wave Passing". AIAA Paper ISABE 89-7070, 1989.

7 Johnson, A. B., Oldfield, M. L. G., Rigby, M. J., and Giles, M. B., "Nozzle Guide Vane Shock Wave Propagation and Bifurcation in a Transonic Turbine Rotor". ASME Paper No. 90-GT-370, 1990.

8 Hilditch, M. A., and Ainsworth, R. W., "Unsteady Heat Transfer Measurements on a Rotating Gas Turbine". ASME Paper No. 90-GT-175, 1990.

9 Guenette, G. R., Epstein, A. H., Giles, M. B., Haimes, R. and Norton, R. J. G., "Fully Scaled Transonic Turbine Rotor Heat Transfer Measurements", ASME Paper No. 88-GT-171, 1988.

10 Dunn, M. G., Seymour, P. J., Woodward, S. H., George, W. K., and Chupp, R. E., "Phase-Resolved Heat Flux Measurments on the Blade of a Full-Scale Rotating Turbine". ASME Paper No. 88-GT-173, 1988.

11 Dunn, M. G., "Phase and Time-Resolved Measurements of Unsteady Heat Transfer and Pressure in a Full-Stage Rotating Turbine". ASME Journal of Turbomachinery, Vol. 112, pp. 531-538, July 1990.

12 Collie, J. C., Moses, H. L., Schetz, J. A., and Gregory, B. A., "Recent Advances in Simulating Unsteady Flow Phenomena Brought About by Passage of Shock Waves in a Linear Turbine Cascade". ASME Journal of Turbomachinery, Volume 115 (Oct 1993): pp. 687-698.

13 Doughty, R. L., "Effects of Multiple Incident Shock Waves on the Flow in a Transonic Turbine Cascade". PhD Dissertation, Virginia Tech, Blacksburg Virginia, USA, 1994.

14 Nix, A. C., "Effects of Shock Wave Passing on Turbine Blade Heat Transfer in a Transonic Turbine Cascade". Master's Thesis, Virginia Tech, Blacksburg, Virginia, USA, 1996.

15 Diller, T. E., "Advances in Heat Flux Measurements", *Advances in Heat Transfer*, Volume 23, Academic Press New York, pp. 279-368, 1994.

16 Peabody, H. L., "Evaluation of a Heat Flux Microsensor in a Transonic Turbine Cascade". Master's Thesis, Virginia Tech, Blacksburg, Virginia, USA, 1997.

17 Anderson, D. A., Tannehill, J. C., Pletcher, R. H., *Computational Fluid Mechanics and Heat Transfer*, McGraw-Hill, 1984, pp. 480-481.

18 Hirsch, C., *Numerical Computation of Internal and External Flows: Volume 2*, Wiley Publishers, 1992, p. 224-300.

19 Cline, M. C., "VNAP2: A Computer Program for Computation of 2-D, Unsteady, Compressible Turbulent Flows". Los Alamos Laboratory Document LA-8872, Aug 1981.

20 Giles, M. B., "Stator/Rotor Interaction in a Transonic Turbine". AIAA Journal of Propulsion (88-3093), 1988.

21 Giles, M. B., "Stator/Rotor Interaction in a Transonic Turbine". AIAA Journal of Propulsion (88-3093), 1988.

22 Anderson, D. A., Tannehill, J. C., Pletcher, R. H., *Computational Fluid Mechanics and Heat Transfer*, McGraw-Hill, 1984, pp. 190-193.

23 Huber, P. W., Fitton, C. E., Jr., and Delpino, F., "Experimental Investigation of Moving Pressure Disturbances and Shock Waves and Correlation with One-Dimensional Unsteady-Flow Theory", NACA Technical Note, No. 1878, May 1949.

24 Anderson, J. D., *Modern Compressible Flow*, McGraw-Hill Publishing, 1990, Appendix A., Table 1, pp. 623-626.

25 Moss, R. W., Ainsworth, R. W., Sheldrake, C. D. and Miller, R., "The Unsteady Pressure Field Over A Turbine Blade Surface: Visualisation And Interpretation Of Experimental Data", ASME Paper 97-GT-474, Orlando FLA, June1997.

26 Tannehill, J. C., Holst, T. L., and Rakich, J. V., "Numerical Computation of Two-Dimensional Viscous Blunt Body Flows with an Impinging Shock", AIAA Paper 75-154, Pasadena, CA, 1975.

27 Hirsch, C., *Numerical Computation of Internal and External Flows: Volume 2*, Wiley Publishers, 1992, pp. 240-241.

28 Roache, P. J., *Computational Fluid Dynamics*, Hermosa Publishers, pp. 262-272, 1972.

29 Anderson, J. D. , *Computational Fluid Dynamics*, McGraw-Hill, pp. 455-457, 1995.

30 Anderson, J. D., *Modern Compressible Flow*, McGraw-Hill Publishing, 206-214, 1990.

31 Anderson, J. D., *Modern Compressible Flow*, McGraw-Hill Publishing, 236-238, 1990.

32 Incropera, F. P., *Fundamentals of Heat Transfer*, Wiley Publishers, 1981, pp.204-205.

33 Kays, W. M., and Crawford M. E., *Convection Heat and Mass Transfer*, McGraw-Hill, 1993, pp.88-104 and 69-75.

34 Greenburg, M. D., *Advanced Engineering Mathematics*, Prentice Hall Publishers, pg. 464, 1988.

35 Holmberg, D. G., and Diller, T. E., "High-Frequency Heat Flux Sensor Calibration and Modelling", Presented at the 1994 ASME Winter Annual Meeting in Chicago in Ill, 1994.

36 Holmberg, D. L., Mukkamala, Y. S., and Diller, T. E., "Shock Tunnel Evaluation of Heat Flux Sensors", AIAA Paper No. 94-0730, Presented at 32nd Aerosciences Meeting and Exhibit, Reno, NV, January 1994.

37 MacCormack, R. W., "The Effect of Viscosity in Hypervelocity Impact Cratering". AIAA Paper No. 69-354, 1969.

38 Abhari, Guenette, Epstein, and Giles, "Comparison of Time-Resolved Turbine Rotor Blade Heat Transfer Measurements and Numerical Calculations". Journal of Turbomachinery, Volume 114 (Oct 1992): pp. 818-827.

39 Hung, C. M., and MacCormack, R. W., "Numerical Solutions of Supersonic and Hypersonic Laminar Compressible Corner Flows". AIAA Journal, Volume 14, No. 4, pp. 475-480, 1976.

40 MacCormack, R. W., and Baldwin, B. S., “A Numerical Method for Solving the Navier-Stokes Equations with Applications to Shock-Boundary Interactions”. AIAA Paper No. 75-1, 1975.

APPENDIX A: NONDIMENSIONALIZATION OF THE GOVERNING EQUATIONS

In this appendix, the complete 3-D Navier-Stokes equations will be described. The components of stress-stress and heat flux are defined. In addition, the simplifying assumptions (previously mentioned in Chapter 3) that reduce the spatial order from 3-D to "quasi" 2-D will be described. The manner in which these equations are non-dimensionalized will be shown. The final form of these equations are used to produce all of the results contained in this document.

A.1 The Complete Navier-Stokes Equations

The complete 3-D Navier-Stokes equations describe all physical flow scenarios. However, they are extremely difficult to solve. The computer storage necessary to do so can be massive. It is in the best interest of the engineer to reduce this system of equations to something more manageable. If done correctly, the resulting system of equations can contain some of the relevant flow physics of interest while omitting others that effect the problem minimally. The starting point for the simplification of the governing equations begins with the compressible 3-D Navier-Stokes equations in Cartesian coordinates without body forces or external heat addition. This systems of equations can be written in vector form

$$\frac{\partial \vec{Q}}{\partial t} + \frac{\partial \vec{E}}{\partial x} + \frac{\partial \vec{F}}{\partial y} + \frac{\partial \vec{G}}{\partial z} = 0 \qquad [A.1]$$

where $\vec{Q}, \vec{E}, \vec{F}$,and $\vec{G}$ are vectors given by

$$\vec{Q} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho E \end{bmatrix}$$

$$\vec{E} = \begin{bmatrix} \rho u \\ \rho u^2 + P - \tau_{xx} \\ \rho uv - \tau_{xy} \\ \rho uw - \tau_{xz} \\ (\rho E + P)u - u\tau_{xx} - v\tau_{xy} - w\tau_{xz} + q_x \end{bmatrix}$$

$$\vec{F} = \begin{bmatrix} \rho v \\ \rho uv - \tau_{xy} \\ \rho v^2 + P - \tau_{yy} \\ \rho vw - \tau_{yz} \\ (\rho E + P)v - u\tau_{xy} - v\tau_{yy} - w\tau_{yz} + q_y \end{bmatrix}$$

$$\vec{G} = \begin{bmatrix} \rho w \\ \rho uw - \tau_{xz} \\ \rho vw - \tau_{yz} \\ \rho w^2 - \tau_{zz} \\ (\rho E + P)w - u\tau_{xz} - v\tau_{yz} - w\tau_{zz} + q_z \end{bmatrix} \qquad [A.2]$$

The first row of the vector equation A.1 corresponds to the continuity equation. Similarly, the second, third and fourth rows correspond to the x-, y-, and z-momentum equations. The fifth row is the energy equation. The components of the shear-stress and heat flux vector are given by

$$\tau_{xx} = \frac{2}{3}\mu\left(2\frac{\partial u}{\partial x} - \frac{\partial v}{\partial y} - \frac{\partial w}{\partial z}\right)$$

$$\tau_{yy} = \frac{2}{3}\mu\left(2\frac{\partial v}{\partial y} - \frac{\partial u}{\partial x} - \frac{\partial w}{\partial z}\right)$$

$$\tau_{zz} = \frac{2}{3}\mu\left(2\frac{\partial w}{\partial z} - \frac{\partial u}{\partial x} - \frac{\partial v}{\partial y}\right)$$

$$\tau_{xy} = \mu\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right) = \tau_{yx}$$

$$\tau_{xz} = \mu\left(\frac{\partial w}{\partial x} + \frac{\partial u}{\partial z}\right) = \tau_{zx}$$

$$\tau_{yz} = \mu\left(\frac{\partial v}{\partial z} + \frac{\partial w}{\partial y}\right) = \tau_{zy}$$

$$q_x = -k\frac{\partial T}{\partial x}$$

$$q_y = -k\frac{\partial T}{\partial y} \qquad \text{[A.3]}$$

$$q_z = -k\frac{\partial T}{\partial z}$$

The combination of equations A.1, A.2, and A.3 make up the 3-D compressible Navier-Stokes equations. Simplifying assumptions are made to this three-dimensional system of equations to reduce the complexity of the governing equations. They will be discussed in the next section.

A.2 Simplifying Assumptions

Assumption 1 states that the z-direction (w-velocity) terms are neglected. This removes the entire z-momentum equation from the system of equations. Any relation with a z- component or a w-velocity is replaced with zero. This assumption reduces the spatial order of the Navier-Stokes equations from 3-D to 2-D.

Assumption 2 states that only the fluxes in the y-direction are considered important. This is a critical assumption because the influences of the flow at adjacent

blade locations do not influence the location chosen for study. The boundary layer thickness and the freestream velocity is not a function of position. To satisfy y-momentum, the pressure gradient in the y-direction is zero. It is important to note that the equations still allow a flow with both x- and y- components of velocity to be investigated.

After these assumptions have been made, the equations take on the following vector form:

$$\frac{\partial \vec{Q}}{\partial t} + \frac{\partial \vec{F}}{\partial y} = 0 \qquad \text{[A.4]}$$

where $\vec{Q}$ and $\vec{F}$ are vectors given by

$$\vec{Q} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho E \end{bmatrix}$$

$$\vec{F} = \begin{bmatrix} \rho v \\ \rho uv - \tau_{xy} \\ \rho v^2 + P - \tau_{yy} \\ (\rho E + P)v - u\tau_{xy} - v\tau_{yy} + q_y \end{bmatrix} \qquad \text{[A.5]}$$

After the previously discussed assumptions, the remaining components of the shear-stress and heat flux vector are

$$\tau_{yy} = \frac{4}{3}\mu\left(\frac{\partial v}{\partial y}\right)$$

$$\tau_{xy} = \mu\left(\frac{\partial u}{\partial y}\right) = \tau_{yx} \qquad \text{[A.6]}$$

$$q_y = -k\frac{\partial T}{\partial y}$$

Equations [A.4], [A.5], and [A.6] define the "quasi" 2-D compressible Navier-Stokes equations. This system of equations define the simplified numerical simulation that is being evaluated in this document.

A.3 Nondimensionalization Of The "Quasi" 2-D Navier-Stokes Equations

The reduction of the Navier-Stokes equations have discussed in the previous sections. The system of equations can be written in the following form:

$$\frac{\partial}{\partial t}\begin{bmatrix}\rho\\ \rho u\\ \rho v\\ \rho E\end{bmatrix}+\frac{\partial}{\partial y}\begin{bmatrix}\rho v\\ \rho uv-\tau_{xy}\\ \rho v^2+P-\tau_{yy}\\ (\rho E+P)v-u\tau_{xy}-v\tau_{yy}+q_y\end{bmatrix}=0 \qquad [A.7]$$

Numerous procedures for nondimensionalizing the governing equations are possible. The procedure chosen for this body of research is

$$y^*=\frac{y}{L} \qquad t^*=\frac{a_\infty t}{L} \qquad \mu^*=\frac{\mu}{\mu_\infty} \qquad k^*=\frac{k}{k_\infty}$$

$$u^*=\frac{u}{a_\infty} \qquad v^*=\frac{v}{a_\infty} \qquad T^*=\frac{T}{T_\infty}$$

$$\rho^*=\frac{\rho}{\rho_\infty} \qquad P^*=\frac{P}{\rho_\infty a_\infty^2} \qquad e^*=\frac{e}{a_\infty^2}$$

where the nondimensional variables are denoted by the asterisk (*), freestream conditions are denoted by ∞, and L is the reference length corresponding to the length of the computational window. The following parameter, which is defined as the Reynolds number, shows up as a result of this nondimensionalization procedure:

$$\mathrm{Re}_L^*=\frac{\rho_\infty a_\infty L}{\mu_\infty}$$

The final nondimensionalized system of equations becomes

$$\frac{\partial \vec{Q}^*}{\partial t^*}+\frac{\partial \vec{F}^*}{\partial y^*}=0 \qquad [A.8]$$

where the nondimensionalized vectors $\vec{Q}^*$ and $\vec{F}^*$ defined as

$$\vec{Q}^* = \begin{bmatrix} \rho^* \\ \rho^* u^* \\ \rho^* v^* \\ \rho^* E^* \end{bmatrix}$$

$$\vec{F}^* = \begin{bmatrix} \rho^* v^* \\ \rho^* u^* v^* - \dfrac{\mu^*}{\mathrm{Re}_L^*}\left(\dfrac{\partial u^*}{\partial y^*}\right) \\ \rho^* v^{*2} + P^* - \dfrac{4\mu^*}{3\mathrm{Re}_L^*}\left(\dfrac{\partial v^*}{\partial y^*}\right) \\ (\rho^* E^* + P^*)v^* - u^* \dfrac{\mu^*}{\mathrm{Re}_L^*}\left(\dfrac{\partial u^*}{\partial y^*}\right) - v^* \dfrac{4\mu^*}{3\mathrm{Re}_L^*}\left(\dfrac{\partial v^*}{\partial y^*}\right) - \dfrac{1}{(\gamma-1)\mathrm{Re}_L^* \mathrm{Pr}}\left(k^* \dfrac{\partial T^*}{\partial y^*}\right) \end{bmatrix} \quad \text{[A.9]}$$

Equations A.8 and A.9 represent the "quasi" 2-D compressible Navier-Stokes equations. The solutions generated in this document are obtained from this nondimensionalized system of equations.

APPENDIX B: ARTIFICIAL VISCOSITY SMOOTHING

Spurious oscillations tend to appear in a computational region containing a discontinuity. Smoothing routines (sometimes referred to as "artificial" viscous dissipation) are often needed to minimize these oscillations. In this section, the source of these oscillations and the "artificial" viscosity smoothing routine used to minimize them will be discussed briefly. As a result, a specific "artificial" viscosity smoothing chosen will be used throughout the document.

B.1 Establish the Order of Truncation Error

As stated in Chapter 3, the "quasi" 2-D Navier-Stokes equations takes on the form

$$\frac{\partial \vec{Q}}{\partial t} + \frac{\partial \vec{F}}{\partial y} = 0 \qquad \text{[B.1]}$$

To develop a finite difference expression for this equation, the following Taylor's series expansion is developed:

$$F_{j+1} = F_j + \Delta y \frac{\partial F}{\partial y} + \frac{\Delta^2 y}{2!}\frac{\partial^2 F}{\partial y^2} + \frac{\Delta^3 y}{3!}\frac{\partial^3 F}{\partial y^3} + \frac{\Delta^4 y}{4!}\frac{\partial^4 F}{\partial y^4} + \ldots$$

from this equation, the forward difference representation can be written as

$$\left.\frac{\partial F}{\partial y}\right|_{fwd} = \frac{F_{j+1} - F_j}{\Delta y} + \Delta y\left(\frac{1}{2!}\frac{\partial^2 F}{\partial y^2} + \frac{\Delta y}{3!}\frac{\partial^3 F}{\partial y^3} + \frac{\Delta^2 y}{4!}\frac{\partial^4 F}{\partial y^4} + \ldots\right) \qquad \text{[B.2]}$$

In a similarly manner,

$$\left.\frac{\partial F}{\partial y}\right|_{bckwd} = \frac{F_1 - F_{j-1}}{\Delta y} + \Delta y\left(-\frac{1}{2!}\frac{\partial^2 F}{\partial y^2} + \frac{\Delta y}{3!}\frac{\partial^3 F}{\partial y^3} - \frac{\Delta^2 y}{4!}\frac{\partial^4 F}{\partial y^4} + \ldots\right) \qquad \text{[B.3\}}$$

$$\left.\frac{\partial Q}{\partial t}\right|_{fwd} = \frac{Q_{j+1} - Q_j}{\Delta t} + \Delta t\left(\frac{1}{2!}\frac{\partial^2 Q}{\partial t^2} + \frac{\Delta t}{3!}\frac{\partial^3 Q}{\partial t^3} + \frac{\Delta^2 t}{4!}\frac{\partial^4 Q}{\partial t^4} + \ldots\right) \qquad \text{[B.4]}$$

Recalling the discussion of MacCormack's method, a forward predictor/backward corrector technique is used to solve equation B.1. The average of this simultaneous technique can have an implied of

$$\frac{1}{2}\left\{\frac{\partial \vec{Q}}{\partial t} + \frac{\partial \vec{Q}}{\partial t}\right\} + \frac{1}{2}\left\{\left.\frac{\partial \vec{F}}{\partial y}\right|_{fwd} + \left.\frac{\partial \vec{F}}{\partial y}\right|_{bckwd}\right\} = \textbf{\textit{truncation error}} \qquad \text{[B.5]}$$

where

$$\begin{aligned}\textbf{\textit{truncation error}} = &-\frac{\Delta t}{2}\left(\frac{\Delta t}{2!}\frac{\partial^2 Q}{\partial t^2} + \frac{\Delta^2 t}{3!}\frac{\partial^3 Q}{\partial t^3} + \frac{\Delta^3 t}{4!}\frac{\partial^4 Q}{\partial t^4} + \ldots\right) \\ &-\frac{\Delta y}{2}\left(+\frac{1}{2!}\frac{\partial^2 F}{\partial y^2} + \frac{\Delta y}{3!}\frac{\partial^3 F}{\partial y^3} + \frac{\Delta^2 y}{4!}\frac{\partial^4 F}{\partial y^4} + \ldots\right) \\ &-\frac{\Delta y}{2}\left(-\frac{1}{2!}\frac{\partial^2 F}{\partial y^2} + \frac{\Delta y}{3!}\frac{\partial^3 F}{\partial y^3} - \frac{\Delta^2 y}{4!}\frac{\partial^4 F}{\partial y^4} + \ldots\right)\end{aligned}$$

If the portion of the truncation which is y-dependent is combined, the even derivatives will drop out. This defines the first term of the y-dependent truncation error term as a third derivative.

According to Anderson, the order of the first term of the truncation error determines how a numerical technique will handle a step discontinuity. If the first term is an even derivative, then the step discontinuity will be smeared over a larger distance. The term *dissipation* is used to describe the nature in which a step discontinuity is smeared. If the first term is an odd derivative, then the step discontinuity experiences spurious oscillations. The term *dispersion* is used to describe the nature in which the step discontinuity oscillates. MacCormack's method has an odd derivative as the first term in the y-dependent variable's truncation error. The results of this odd derivative can be seen in figure B.1. The effects of the odd derivative need to be address so that the spurious oscillations can be minimized.

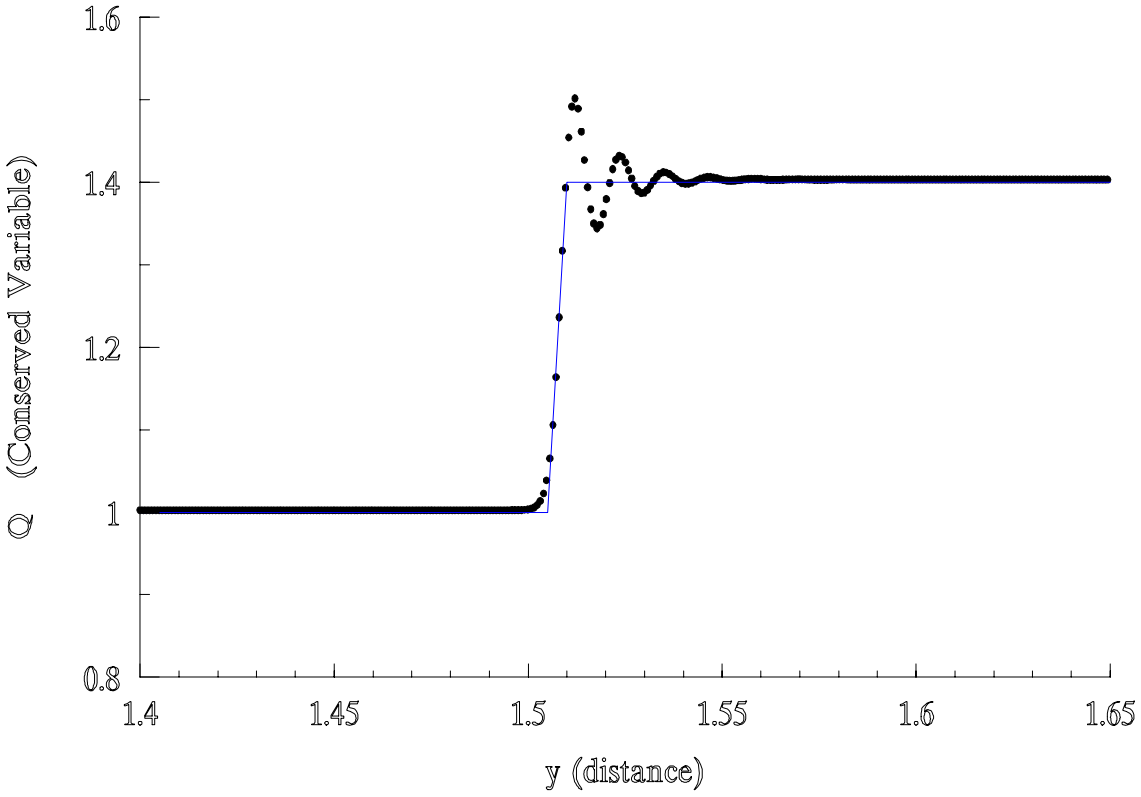

Figure B.1 Example of Dispersion on a Spatial Discontinuity

B.2 Artificial Dissipation

In order to minimize the spurious oscillations generated by the odd derivative (present in the spatial truncation error terms), a standard practice is to add an even order derivative. This introduces a term that resembles "artificial" dissipation (viscosity). This term can damp out the high frequency components introduced by the odd derivative, but at the expense of severely smearing the sharp spatial discontinuity. If smearing does not contaminant the solution of the equations, large amounts of "artificial" dissipation can effectively reduce spatial oscillations.

Numerous forms of a "artificial" dissipation have been defined by various researchers. Some examples include

$$\varepsilon \Delta y^2 \frac{|v| + a}{P} \left| \frac{\partial^2 P}{\partial y^2} \right| \qquad \text{(MacCormack and Baldwin[36])}$$

$$\varepsilon \Delta y^3 \left(|v| + a \right) \left| \frac{\partial^2 Q}{\partial y^2} \right| \qquad \text{(Steger[37])}$$

In these expressions, ε is a user defined constant that enables the magnitude of the "artificial" dissipation to be increased. Q is the vector of conserved variables, while P is the pressure, v is the velocity and a is the speed of sound. However, MacCormack suggested the use of

$$\langle SC \rangle \frac{|P - 2P + P|}{P + 2P + P} \left| \frac{\partial^2 Q}{\partial y^2} \right| \left(|v| + a \right) \qquad \text{(MacCormack)}$$

which is applied to equation D.1. This equation takes on the form of

$$\frac{\partial \vec{Q}}{\partial t} + \frac{\partial \vec{F}}{\partial y} = \langle SC \rangle \frac{\left| P_{j+1} - 2P_j + P_{j-1} \right|}{P_{j+1} + 2P_j + P_{j-1}} \left| \frac{\partial^2 Q}{\partial y^2} \right| \left(|v| + a \right) \qquad \text{[B.6]}$$

In equation B.6, the "artificial" dissipation term has a smoothing factor that is a function of pressure. This smoothing factor, which is the ratio of second centered differences of pressure, controls where the "artificial" dissipation term is applied. If there are spurious oscillations about a discontinuity, the smoothing will focus on that region without

effecting the solution away from the discontinuity. The *smoothing constant* ⟨**SC**⟩ is a user defined constant that permits the amount of "artificial" dissipation to be increased. According to MacCormack, this parameter has a maximum range, which is $0 < \mathbf{SC} < 0.5$. The equations being studied in this document actually imposes a more limited range. which turns out to be $0 < \mathbf{SC} < 0.1$.

APPENDIX C SOURCE CODE

```
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
ccccc
cccc
cc
c
c
c     TVRSHOCK COMPUTES THE PROPAGATION OF A SHOCK USING
c     MACCORMACK SCHEME WITH ARTIFICIAL VISCOSITY OR FCT SMOOTHING
c     ON THE 1-D NAVIER STOKES EQUATIONS
c
cc
cccc
ccccc
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
ccc
cc                    4-11-98
cc
cccc
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      double precision X(19400),P(19400),RH(19400),U(19400),rhest,rhest1,xxx
      double precision den(4),num(4),re,htc,qht,tau(19400),r(19400,4),r1,r2,r3,r4
      double precision QD(19400,4),F(19400,4),DF(19400,4),DL(4),DLM(4),nussult
      double precision Q(19400,4),V(19400),uex(19400),Tw,T(19400),a(19400)
      double precision XD(19400),RHD(19400),UD(19400),TD(19400),kd(19400)
      double precision VD(19400),rhd1,dx,a2,pe,pr,dtr, QC(19400,4),mu(19400)
      double precision eta,et1,et2,emu,enu,L,k1,k2,T2,war,dum,dim,pec(19400)
      double precision anx,cx,k,mu2,jsst,PD(19400),mm(19400),qw1,qw2,s(19400,4),sd(19400,4)
      double precision r01,r02,r03,r04,resid1,resid2,resid3,resid4
      integer ddd,ee
      OPEN(912,FILE='rsid1.DAT')
      OPEN(913,FILE='rsid2.DAT')
      OPEN(914,FILE='rsid3.DAT')
      OPEN(915,FILE='rsid4.DAT')
      OPEN(916,FILE='htc.DAT')
      OPEN(917,FILE='nuss.DAT')
      OPEN(918,FILE='deltat.DAT')
      open(919,file='tauh.DAT')
      open(9119,file='dv.DAT')
      OPEN(91,FILE='y06cy11.DAT')
      OPEN(92,FILE='y06cy12.DAT')
      OPEN(93,FILE='y06cy13.DAT')
      OPEN(94,FILE='y06cy14.DAT')
      OPEN(95,FILE='y06cy15.DAT')
      OPEN(96,FILE='y06cy16.DAT')
      OPEN(97,FILE='y06cy17.DAT')
```

```
      OPEN(98,FILE='y06cy18.DAT')
      OPEN(99,FILE='y06cy19.DAT')
      OPEN(80,FILE='y06cy20.DAT')
      OPEN(81,FILE='y06cy21.DAT')
      OPEN(82,FILE='y06cy22.DAT')
      OPEN(83,FILE='y06cy23.DAT')
      OPEN(84,FILE='y06cy24.DAT')
      OPEN(85,FILE='y06cy25.DAT')
      OPEN(941,FILE='x06cy11.DAT')
      OPEN(942,FILE='x06cy12.DAT')
      OPEN(943,FILE='x06cy13.DAT')
      OPEN(944,FILE='x06cy14.DAT')
      OPEN(945,FILE='x06cy15.DAT')
      OPEN(946,FILE='x06cy16.DAT')
      OPEN(947,FILE='x06cy17.DAT')
      OPEN(948,FILE='x06cy18.DAT')
      OPEN(949,FILE='x06cy19.DAT')
      OPEN(951,FILE='x06cy20.DAT')
      OPEN(952,FILE='x06cy21.DAT')
      OPEN(953,FILE='x06cy22.DAT')
      OPEN(954,FILE='x06cy23.DAT')
      OPEN(955,FILE='x06cy24.DAT')
      OPEN(956,FILE='x06cy25.DAT')
      OPEN(891,FILE='r06cy11.DAT')
      OPEN(892,FILE='r06cy12.DAT')
      OPEN(893,FILE='r06cy13.DAT')
      OPEN(894,FILE='r06cy14.DAT')
      OPEN(895,FILE='r06cy15.DAT')
      OPEN(896,FILE='r06cy16.DAT')
      OPEN(897,FILE='r06cy17.DAT')
      OPEN(898,FILE='r06cy18.DAT')
      OPEN(899,FILE='r06cy19.DAT')
      OPEN(791,FILE='p06cy11.DAT')
      OPEN(792,FILE='p06cy12.DAT')
      OPEN(793,FILE='p06cy13.DAT')
      OPEN(794,FILE='p06cy14.DAT')
      OPEN(795,FILE='p06cy15.DAT')
      OPEN(796,FILE='p06cy16.DAT')
      OPEN(797,FILE='p06cy17.DAT')
      OPEN(798,FILE='p06cy18.DAT')
      OPEN(799,FILE='p06cy19.DAT')
      OPEN(910,FILE='t06cy11.DAT')
      OPEN(920,FILE='t06cy12.DAT')
      OPEN(930,FILE='t06cy13.DAT')
      OPEN(940,FILE='t06cy14.DAT')
      OPEN(950,FILE='t06cy15.DAT')
      OPEN(960,FILE='t06cy16.DAT')
      OPEN(970,FILE='t06cy17.DAT')
      OPEN(980,FILE='t06cy18.DAT')
      OPEN(990,FILE='t06cy19.DAT')
      OPEN(991,FILE='t06cy20.DAT')
      OPEN(992,FILE='t06cy21.DAT')
      OPEN(993,FILE='t06cy22.DAT')
      OPEN(994,FILE='t06cy23.DAT')
      OPEN(995,FILE='t06cy24.DAT')
```

```
      OPEN(996,FILE='t06cy25.DAT')
      OPEN(53,FILE='hfm06c13.DAT')
      WRITE(*,*)
      WRITE(*,*)
      WRITE(*,*)
      WRITE(*,*)
      WRITE(*,*) 'Computing .....  '
      write(*,*)
      write(*,*)
      IFCT = 2
      IPR = 1
      ime = 1
      NX = 9001
      NT = 640*45
      DT = 0.0020833/45.
      GAM = 1.4
      PRT = 1.1
      ETA = 1./8.
      ET1 = 1./2
      ET2 = -1./8.
      ENL =  0.1
      SHST = 0.901
      PI = 3.1415927
      NXM = NX - 1
      ANX = NXM
      DX = 1./ANX
      JSST = SHST/DX
      DTR = DT/DX
      GMM = GAM -1.
      GMP = GAM + 1.
      GMR = GMM/GMP
      SHS = 0.5*(GMM + GMP*PRT)/GAM
      SHS = SQRT(SHS)
c
c        SET INITIAL CONDITIONS
c
c
      p2 = 101325.
      t2 = 330.00
      u2 = 0.0
      v2 = 0.52
      rh2 = 1.06985
      A2 = sqrt(gam*287.*t2)
      DUM = 1./GMR*PRT + 1.
      RHd1 = DUM/(PRT +1./(GMR))
      DIM = SQRT(2.*GAM/GMP/DUM)
      ud1 = (1./gam)*(prt-1.)*(( (2.*gam/(gam+1.))/(prt+(gmm/gmp)))**.5)
      VD1 = v2
      td1 = prt*(1. + gmr*prt)/(prt +gmr)
      L = .03
      k1 = 0.02852
      k2 = 0.2852
      k3 = 0.002852
      k = k1
      cp = 1008.0
```

```
      pe = k*t2/(rh2*a2*a2*a2*L)
      twall = 300./t2
      cx = 0.1
      pr = 0.7042
      cv = cp/gam
      mu2 = .000019876
      do 992 j=1,66
        xg=float(j)*dx*L
        t(nx-j) = (300.+30.*(1.5*(xg/0.00011) - 0.5*(xg/0.00011)*(xg/0.00011)*(xg/0.00011) ))/t2
  992 continue
      do 223 j=1,jsst
        t(j) = td1
  223 continue
      do 993 j=jsst,8934
        t(j) = 1.0
  993 continue
      do 994 j=1,60
        xg=float(j)*dx*L
        v(nx-j) = vd1*(1.5*(xg/0.0001) - 0.5*(xg/0.0001)*(xg/0.0001)*(xg/0.0001) )
  994 continue
      do 995 j=1,8940
        v(j) = vd1
  995 continue
      v(nx) = -v(nxm)
      t(nx) = 2.*twall- t(nxm)
      DO 8 J=1,jsst
        U(J) = ud1
        P(J) = Prt*p2/(rh2*a2*a2)
        a(j) = sqrt(gam*287.*t2*t(j))
        rh(j) = prt*p2/(287.*t2*t(j)*rh2)
        Mu(j) =  0.00000145*((T(j)*T2)**1.5)/(T(j)*T2 + 110.)
        kd(j) = mu(j)*cp/pr
        pec(j) = kd(j)*t2*t(j)/(rh2*a2*a2*a2*L)
    8 CONTINUE
      JHR = JSST + 1
      DO 9 J=JHR,Nx+1
         U(J) = u2
         RH(J) = p2/(287.*t2*t(j)*rh2)
         P(J) = p2/(rh2*a2*a2)
         a(j) = sqrt(gam*287.*t2*t(j))
         Mu(j) =  0.00000145*((T(j)*T2)**1.5)/(T(j)*T2 + 110.)
         kd(j) = mu(j)*cp/pr
         pec(j) = kd(j)*t2*t(j)/(rh2*a2*a2*a2*L)
    9 CONTINUE
      write(*,*) rhd1,ud1,rhd1*cv*t2*td1/(a2*a2) + .5*rhd1*ud1*ud1
      TIM = 0.
      N=0
c
c        SET INITIAL Q AND F
c
      DO 10 J=1,Nx
         AJ = J-1
         X(J) = AJ*DX
         Q(J,1) = RH(J)
         Q(J,2) = RH(J)*U(J)
```

```
        Q(J,3) = RH(J)*V(J)
        Q(J,4) =  rh(j)*cv*t(j)*t2/(a2*a2)
     *             + 0.5*rh(j)*u(j)*u(j)+0.5*rh(j)*v(j)*v(j)
        F(J,1) = Q(J,2)
        F(J,2) = P(J) + RH(J)*U(J)*U(J)
        F(J,3) = Q(J,2)*V(J)
        F(J,4) = (q(j,4) + p(j))*u(j)
        write(1000,*) x(j)*L*100.,q(j,1)
        write(1006,*) x(j)*L*100.,q(j,3)/q(j,1)
        write(1016,*) x(j)*L*100.,q(j,4)
   10 CONTINUE
        f(1,1) = f(2,1)
        f(1,2) = f(2,2)
        f(1,3) = f(2,3)
        f(1,4) = f(2,4)
c
c
cc
c       ADVANCE SOLUTION IN TIME
c
      DO 30 N = 1,NT
        AN = N
        write(*,*) '************************',t(2406)
        IF(IFCT.NE.1) GO TO 12
        ENU = ETA + 0.25*ET1*((U(1)+U(2))*DTR)**2
        DO 11 iK=1,4
   11   DF(1,iK) = ENU*( Q(2,iK) - Q(1,iK) )
   12   CONTINUE
         rhd(1) = rh(1)
         ud(1) = u(1)
         vd(1) = v(1)
         td(1) = t(1)
         qd(1,1) = rhd(1)
         qd(1,2) = rhd(1)*ud(1)
         qd(1,3) = rhd(1)*vd(1)
         qd(1,4) =  rhd(1)*cv*td(1)*t2/(a2*a2)
     *             + 0.5*rhd(1)*ud(1)*ud(1)+0.5*rhd(1)*vd(1)*vd(1)
        pd(1) = p(2)
        F(1,1) = Qd(1,2)
        F(1,2) = Pd(1) + RHd(j)*Ud(1)*Ud(1) -(mu(1)/(rh2*a2*L))*(ud(2) - ud(1))/dx
        F(1,3) = Qd(1,2)*Vd(1) -(mu(1)/(rh2*a2*L))*(vd(2) - vd(1))/dx
        F(1,4) = (qd(1,4) + pd(1))*ud(1) - pec(1)*(td(2) - td(1))/dx
     *          -(mu(1)/(rh2*a2*L))*(ud(1)*(ud(2) - ud(1)))/dx
     *          -(mu(1)/(rh2*a2*L))*(vd(1)*(vd(2) - vd(1)))/dx

c
c       OBTAIN HALF-STEP SOLUTION
c
        aaa = 2
        bbb = nxm
        ccc = 1
              Do 18 J=aaa,bbb,ccc
              DO 13 nn=1,4
             if(float(n/2.).ne.int(n/2.))QD(J,nn) = Q(J,nn)-DTR*(F(J+1,nn) - F(J,nn))
             if(float(n/2.).eq.int(n/2.))qd(j,nn) = q(j,nn)-dtr*(f(j,nn) - f(j-1,nn))
```

```
          IF(IFCT.EQ.1)ENU=ETA+0.25*ET1*((U(J)+U(J+1))*DTR)**2
1114          IF(IFCT.EQ.1)DF(J,nn)=ENU*(Q(J+1,nn) - Q(J,nn))
     s(j,nn) = cx*abs(P(j+1)-2.*p(j)+p(j-1))*(q(j+1,nn)-2.*q(j,nn)+q(j-1,nn))
   *          /(P(j+1)+2.*p(j)+p(j-1))
9065          s(j,nn) = 0.0
7741          qd(j,nn) = s(j,nn)*dtr*(abs(u(j))+a(j)/a2) +qd(j,nn)
 13       CONTINUE
          rhd(j) = QD(j,1)
          UD(j) = QD(J,2)/QD(J,1)
          VD(j) = QD(J,3)/QD(J,1)
          PD(j) = gmm*(QD(J,4) - .5*QD(j,2)*QD(j,2)/QD(j,1)
   *                        - .5*QD(j,3)*QD(j,3)/QD(J,1) )
          TD(j) = (QD(j,4)/QD(j,1)
   *         -0.5*UD(j)*UD(j) -0.5*VD(j)*VD(j) )
   *        *((a2*a2)/(cv*t2))
 18 continue
          rhd(nx) = rhd(nxm)
          ud(nx) = -ud(nxm)
          vd(nx) = -vd(nxm)
          td(nx) =2.*twall-td(nxm)
          qd(nx,1) = rhd(nx)
          qd(nx,2) = rhd(nx)*ud(nx)
          qd(nx,3) = rhd(nx)*vd(nx)
          qd(nx,4) =  rhd(nx)*cv*td(nx)*t2/(a2*a2)
   *           + 0.5*rhd(nx)*ud(nx)*ud(nx)+0.5*rhd(nx)*vd(nx)*vd(nx)
          pd(nx) = pd(nxm)
    do 800 j=2,nxm
          F(J,1) = rhd(j)*ud(j)
          if(float(n/2.).ne.int(n/2.))F(J,2) = PD(j) + rhd(j)*ud(j)*ud(j)
   *                - (mu(j)/(rh2*a2*L))*(ud(j+1)-ud(j))/dx
          if(float(n/2.).eq.int(n/2.))F(J,2) = PD(j) + rhd(j)*ud(j)*ud(j)
   *                - (mu(j)/(rh2*a2*L))*(ud(j)-ud(j-1))/dx
          if(float(n/2.).ne.int(n/2.))F(J,3) =  rhd(j)*ud(j)*vd(j)
   *                - (mu(j)/(rh2*a2*L))*(vd(j+1)-vd(j))/dx
          if(float(n/2.).eq.int(n/2.))F(J,3) =  rhd(j)*ud(j)*vd(j)
   *                - (mu(j)/(rh2*a2*L))*(vd(j)-vd(j-1))/dx
          if(float(n/2.).ne.int(n/2.))F(j,4) = ( qd(j,4) +pd(j))*ud(j)
   *                - (pec(j))*(td(j+1) - td(j))/dx
   *                - ud(j)*(mu(j)/(rh2*a2*L))*(ud(j+1)-ud(j))/dx
   *                - vd(j)*(mu(j)/(rh2*a2*L))*(vd(j+1)-vd(j))/dx
          if(float(n/2.).eq.int(n/2.))F(j,4) = ( qd(j,4) +pd(j))*ud(j)
   *                - (pec(j))*(td(j) - td(j-1))/dx
   *                - ud(j)*(mu(j)/(rh2*a2*L))*(ud(j)-ud(j-1))/dx
   *                - vd(j)*(mu(j)/(rh2*a2*L))*(vd(j)-vd(j-1))/dx
 800 continue
          Qd(nx,4) =  rhd(nx)*cv*td(nx)*t2/(a2*a2)
   *             + 0.5*rhd(nx)*ud(nx)*ud(nx)+0.5*rhd(nx)*vd(nx)*vd(nx)
          F(nx,1) = qd(nx,2)
          F(nx,2) = qd(nx,2)*ud(nx)+pd(nx) - (mu(nx)/(rh2*a2*L))*(ud(nx)-ud(nxm))/dx
          F(nx,3) = qd(nx,2)*vd(nx) - (mu(nx)/(rh2*a2*L))*(vd(nx)-vd(nxm))/dx
          F(nx,4) = ( qd(nx,4) +pd(nx))*ud(nx)
   *                - (pec(nx))*(td(nx) - td(nx-1))/dx
   *                - ud(nx)*(mu(nx)/(rh2*a2*L))*(ud(nx)-ud(nxm))/dx
   *                - vd(nx)*(mu(nx)/(rh2*a2*L))*(vd(nx)-vd(nxm))/dx
          rh(1) = rhd(1)
```

```
          u(1) = ud(1)
          v(1) = vd(1)
          t(1) = td(1)
          q(1,1) = rh(1)
          q(1,2) = rh(1)*u(1)
          q(1,3) = rh(1)*v(1)
          q(1,4) =  rh(1)*cv*t(1)*t2/(a2*a2)
     *             + 0.5*rh(1)*u(1)*u(1)+0.5*rh(1)*v(1)*v(1)
          p(1) = pd(2)
          F(1,1) = Q(1,2)
          F(1,2) = P(1) + RH(1)*U(1)*U(1)- (mu(1)/(rh2*a2*L))*(u(2)-u(1))/dx
          F(1,3) = Q(1,2)*V(1)-(mu(1)/(rh2*a2*L))*(v(2)-v(1))/dx
          F(1,4) = (q(1,4) + p(1))*u(1) - pec(1)*(t(2) - t(1))/dx
     *            - u(1)*(mu(1)/(rh2*a2*L))*(u(2)-u(1))/dx
     *            - v(1)*(mu(1)/(rh2*a2*L))*(v(2)-v(1))/dx
c
c           OBTAIN FULL-STEP SOLUTION
c
       do 914 j= aaa,bbb,ccc
       do 14 ee = 1,4
        if(float(n/2.).ne.int(n/2.))QC(j,ee) = 0.5*(Q(j,ee)+QD(j,ee))
     *                              -.5*DTR*(F(j,ee) - F(j-1,ee))
        if(float(n/2.).eq.int(n/2.))qc(j,ee) = 0.5*(q(j,ee)+qd(j,ee))
     *                              -.5*dtr*(F(j+1,ee) - F(j,ee))
          sd(j,ee) = cx*abs(Pd(j+1)-2.*pd(j)+pd(j-1))*(qd(j+1,ee)-2.*qd(j,ee)+qd(j-1,ee))/
     *     (Pd(j+1)+2.*pd(j)+pd(j-1))
          r(j,ee) = abs(qc(j,ee)-q(j,ee))/dt
 9066           sd(j,ee) = 0.0
 7742         q(j,ee) = qc(j,ee) + sd(j,ee)*(abs(u(j))+a(j)/a2)*dtr
   14        CONTINUE
          rh(j) = q(j,1)
          U(j) = Q(j,2)/Q(j,1)
          V(j) = Q(j,3)/Q(j,1)
          T(j) = (Q(j,4)/Q(j,1)
     *       -0.5*Q(j,2)*Q(j,2)/(Q(j,1)*Q(j,1))
     *       -0.5*Q(j,3)*Q(j,3)/(Q(j,1)*Q(j,1))  )
     *       *((a2*a2)/(cv*t2))
          p(j) = gmm*(q(j,4)-.5*q(j,2)*u(j)-.5*q(j,3)*v(j))
  914 continue
          rh(nx) = rhd(nxm)
          u(nx) = -1.*ud(nxm)
          v(nx) = -vd(nxm)
          t(nx) =2.*twall-td(nxm)
          q(nx,1) = rh(nx)
          q(nx,2) = rh(nx)*u(nx)
          q(nx,3) = rh(nx)*v(nx)
          q(nx,4) =  rh(nx)*cv*t(nx)*t2/(a2*a2)
     *             + 0.5*rh(nx)*u(nx)*u(nx)+0.5*rh(nx)*v(nx)*v(nx)
          F(nx,1) = q(nx,2)
          F(nx,2) = rh(nx)*u(nx)*u(nx)+p(nx)-(mu(nx)/(rh2*a2*L))*(u(nx)-u(nxm))/dx
          F(nx,3) = rh(nx)*u(nx)*v(nx)-(mu(1)/(rh2*a2*L))*(v(nx)-v(nxm))/dx
          F(nx,4) = ( q(nx,4) +p(nx))*u(nx)
     *               - (pec(nx))*(t(nx) - t(nx-1))/dx
     *               - u(nx)*(mu(nx)/(rh2*a2*L))*(u(2)-u(1))/dx
     *               - v(nx)*(mu(nx)/(rh2*a2*L))*(v(2)-v(1))/dx
```

```
          p(nx) = pd(nxm)
        do 900 jj=2,nxm
         F(jj,1) = rh(jj)*u(jj)
         if(float(n/2.).eq.int(n/2.))F(jj,2) = P(jj) + rh(jj)*u(jj)*u(jj)
     *          -(mu(jj)/(rh2*a2*L))*(u(jj) - u(jj-1))/dx
         if(float(n/2.).ne.int(n/2.))F(jj,2) = P(jj) + rh(jj)*u(jj)*u(jj)
     *          -(mu(jj)/(rh2*a2*L))*(u(jj+1) - u(jj))/dx
         if(float(n/2.).eq.int(n/2.))F(jj,3) =  rh(jj)*u(jj)*v(jj)
     *          -(mu(jj)/(rh2*a2*L))*(v(jj) - v(jj-1))/dx
         if(float(n/2.).ne.int(n/2.))F(jj,3) =  rh(jj)*u(jj)*v(jj)
     *          -(mu(jj)/(rh2*a2*L))*(v(jj+1) - v(jj))/dx
         if(float(n/2.).eq.int(n/2.))F(jj,4) = ( q(jj,4) +p(jj))*u(jj)
     *          - (pec(jj))*(t(jj) - t(jj-1))/dx
     *          - u(jj)*(mu(jj)/(rh2*a2*L))*(u(jj) - u(jj-1))/dx
     *          - v(jj)*(mu(jj)/(rh2*a2*L))*(v(jj) - v(jj-1))/dx
         if(float(n/2.).ne.int(n/2.))F(jj,4) = ( q(jj,4) +p(jj))*u(jj)
     *          - (pec(jj))*(t(jj+1) - t(jj))/dx
     *          - u(jj)*(mu(jj)/(rh2*a2*L))*(u(jj+1) - u(jj))/dx
     *          - v(jj)*(mu(jj)/(rh2*a2*L))*(v(jj+1) - v(jj))/dx
  900    CONTINUE
c
c   This is a new addition. Removing this section will produce
c    a working code with a laggin temperature profile when compared
c    to the exact solution
c
      if(ifct.ne.2) go to 667
      do 666 j=2,nxm
        if(j.ne.2) go to 404
        duc = abs(u(2)-u(1))
        if(duc.lt.1.0e-6)duc=1.0e-6
        do 444 kk=1,4
          dlm(kk)=q(2,kk) - q(1,kk)
          if(abs(dlm(kk)).lt.1.0e-6)dlm(kk)=1.0e-6*(sign(1.0,dlm(kk)))
          duc = abs(dlm(kk))
  444   dlm(kk)=duc*dlm(kk)
  404 duc = abs(u(J+1)-u(j))
      if(duc.lt.1.0e-6)duc=1.0e-6
      do 555 kk=1,4
           dl(kk)=q(j+1,kk)-q(j,kk)
           if(abs(dl(kk)).lt.1.0e-6)dl(kk)=1.0e-6*(sign(1.0,dl(kk)))
           duc = abs(dl(kk))
           dl(kk) = duc*dl(kk)
           q(j,kk) = q(j,kk) + enl*dtr*(dl(kk)-dlm(kk))
           dlm(kk)=dl(kk)
  555 continue
  666 continue
  667      write(*,*)
       IF(IFCT .EQ. 1)CALL FCT(NXM,n,ETA,ET2,DTR,Q,Df,U)
c
c          OBTAIN RH,U,V,P AND F
c
       DO 19 J=2,nxm
          rh(j) = q(j,1)
          u(j) = q(j,2)/q(j,1)
          v(j) = q(j,3)/q(j,1)
```

```
      t(j) = (q(j,4)/q(j,1)
     *     - .5*u(j)*u(j) -.5*v(j)*v(j))
     *     *((a2*a2)/(cv*t2))
      p(j) = gmm*(q(j,4)-.5*q(j,2)*u(j)-.5*q(j,3)*v(j))
      a(j) = sqrt(gam*287.*t2*t(j))
      Mu(j) =  0.00000145*((T(j)*T2)**1.5)/(T(j)*T2 + 110.)
      kd(j) = mu(j)*cp/pr
      pec(j) = kd(j)*t2*t(j)/(rh2*a2*a2*a2*L)
      F(j,1) = rh(j)*u(j)
      if(float(n/2.).eq.int(n/2.))F(j,2) = P(j) + rh(j)*u(j)*u(j)
     *          - (mu(j)/(rh2*a2*L))*(u(j)-u(j-1))/dx
      if(float(n/2.).ne.int(n/2.))F(j,2) = P(j) + rh(j)*u(j)*u(j)
     *          - (mu(j)/(rh2*a2*L))*(u(j+1)-u(j))/dx
      if(float(n/2.).eq.int(n/2.))F(j,3) =  rh(j)*u(j)*v(j)
     *          - (mu(j)/(rh2*a2*L))*(v(j)-v(j-1))/dx
      if(float(n/2.).ne.int(n/2.))F(j,3) =  rh(j)*u(j)*v(j)
     *          - (mu(j)/(rh2*a2*L))*(v(j+1)-v(j))/dx
      if(float(n/2.).eq.int(n/2.))F(j,4) = ( q(j,4) +p(j))*u(j)
     *          - (pec(j))*(t(j) - t(j-1))/dx
     *          - u(j)*(mu(j)/(rh2*a2*L))*(u(j)-u(j-1))/dx
     *          - v(j)*(mu(j)/(rh2*a2*L))*(v(j)-v(j-1))/dx
      if(float(n/2.).ne.int(n/2.))F(j,4) = ( q(j,4) +p(j))*u(j)
     *          - (pec(j))*(t(j+1) - t(j))/dx
     *          - u(j)*(mu(j)/(rh2*a2*L))*(v(j+1)-v(j))/dx
     *          - v(j)*(mu(j)/(rh2*a2*L))*(v(j+1)-v(j))/dx
19    continue
      rh(nx) = rh(nxm)
      u(nx) = -1.*u(nxm)
      v(nx) = -v(nxm)
      t(nx) = 2.*twall - t(nxm)
      p(nx) = p(nxm)
      q(nx,1) = rh(nx)
      q(nx,2) = rh(nx)*u(nx)
      q(nx,3) = rh(nx)*v(nx)
      q(nx,4) =  rh(nx)*cv*t(nx)*t2/(a2*a2)
     *          + 0.5*rh(nx)*u(nx)*u(nx)+0.5*rh(nx)*v(nx)*v(nx)
      F(nx,1) = rh(nx)*u(nx)
      F(nx,2) = rh(nx)*u(nx)*u(nx)+p(nx) - (mu(nx)/(rh2*a2*L))*(u(nx)-u(nxm))/dx
      F(nx,3) = rh(nx)*u(nx)*v(nx) - (mu(nx)/(rh2*a2*L))*(v(nx)-v(nxm))/dx
      F(nx,4) = ( q(nx,4) +p(nx))*u(nx)
     *          - u(nx)*(mu(nx)/(rh2*a2*L))*(u(nx)-u(nxm))/dx
     *          - v(nx)*(mu(nx)/(rh2*a2*L))*(v(nx)-v(nxm))/dx
     *          - (pec(nx))*(t(nx) - t(nx-1))/dx
     uex(nx)= k*t2*(t(nx-1) - (.5*(t(nx)+t(nxm)) ))/(.5*l*dx)
     xxx = 0.0021
     if(n.lt.7495)re = rh(nxm-66)*rh2*v2*a2*xxx/mu(nxm-66)
     if(n.ge.7495)re = rh(nxm-66)*rh2*v2*a2*xxx/mu(nxm-66)
      if(float(n/2.).eq.int(n/2.)) write(919,*) 1000000.*tim*l/a2,mu(nxm)*a2*v(nxm)/(l*dx*.5)
      if(float(n/2.).eq.int(n/2.)) write(9119,*) 1000000.*tim*l/a2, a2*v(nxm)/(l*dx*.5)
      qht = kd(nxm)*t2*(t(nx-1) - twall)/(.5*l*dx)
      htc = qht/(t2-300.)
      write(917,*) 1000000.*tim*l/a2,htc*xxx/k
      if(float(n/2.).eq.int(n/2.)) write(918,*) 1000000.*tim*l/a2,(t(nxm-66)*t2 - 300.)
      if(float(n/2.).eq.int(n/2.)) write(53,*) 1000000.*tim*l/a2,qht
      write(916,*) 1000000.*tim*l/a2, htc
```

```
        write(621,*) 1000000.*tim*l/a2,.5*(q(nx,4)+q(nxm,4))
        write(622,*) 1000000.*tim*l/a2,q(nx,4)
        write(623,*) 1000000.*tim*l/a2,q(nxm,4)
       TIM = AN*DT
       IF(IPR .EQ. 0) GO TO 30
  20    if(float(n/1.).eq.int(n/1.))WRITE(6,21)N,TIM*l*1000000./a2,p(nxm)*rh2*a2*a2/p2 ,q(2,1),q(2,2)
     *                ,q(2,4)   , uex(nx)
  21    FORMAT(' N=',I6,'   TIM=',f6.3,5x,f13.4,5x,f7.2,5x,f7.2,5x,f8.2,5x,f9.0)
       write(*,*) rhd1,rhd1*ud1, rh(2)*cv*t(2)*t2/(a2*a2)+.5*rh(2)*u(2)*u(2)
       if(float(n/250.).eq.int(n/250.))write(*,*)
       if(float(n/250.).eq.int(n/250.))write(*,*)'*******     ',n,'       *******'
       if(float(n/250.).eq.int(n/250.))write(*,*)
  29    FORMAT('  F=',12F6.3)
     do 418 kk = 1,nx
c
cc    Thermal Boundary Layer History
c
        if(n.eq.1) write(910,*)  t(kk)*t2       ,3.0-x(kk)*L*100.
        if(n.eq.269*45) write(920,*) t(kk)*t2   ,3.0-x(kk)*L*100.
        if(n.eq.12195) write(930,*) t(kk)*t2   ,3.0-x(kk)*L*100.
        if(n.eq.273*45) write(940,*) t(kk)*t2   ,3.0-x(kk)*L*100.
        if(n.eq.12323) write(950,*) t(kk)*t2   ,3.0-x(kk)*L*100.
        if(n.eq.276*45) write(960,*) t(kk)*t2   ,3.0-x(kk)*L*100.
        if(n.eq.12488) write(970,*) t(kk)*t2   ,3.0-x(kk)*L*100.
        if(n.eq.279*45) write(980,*) t(kk)*t2   ,3.0-x(kk)*L*100.
        if(n.eq.12645) write(990,*) t(kk)*t2   ,3.0-x(kk)*L*100.
        if(n.eq.283*45) write(991,*) t(kk)*t2   ,3.0-x(kk)*L*100.
        if(n.eq.12803) write(992,*) t(kk)*t2   ,3.0-x(kk)*L*100.
        if(n.eq.286*45) write(993,*) t(kk)*t2   ,3.0-x(kk)*L*100.
        if(n.eq.12938) write(994,*) t(kk)*t2   ,3.0-x(kk)*L*100.
        if(n.eq.289*45) write(995,*) t(kk)*t2   ,3.0-x(kk)*L*100.
        if(n.eq.600*45) write(996,*) t(kk)*t2   ,3.0-x(kk)*L*100.
c
cc   Aerodynamic Boundary Layer History
c
        if(n.eq.1) write(91,*)  q(kk,3)/q(kk,1)       ,3.0-x(kk)*L*100.
        if(n.eq.269*45) write(92,*) q(kk,3)/q(kk,1)   ,3.0-x(kk)*L*100.
        if(n.eq.12195) write(93,*) q(kk,3)/q(kk,1)   ,3.0-x(kk)*L*100.
        if(n.eq.273*45) write(94,*) q(kk,3)/q(kk,1)   ,3.0-x(kk)*L*100.
        if(n.eq.12323) write(95,*) q(kk,3)/q(kk,1)   ,3.0-x(kk)*L*100.
        if(n.eq.276*45) write(96,*) q(kk,3)/q(kk,1)   ,3.0-x(kk)*L*100.
        if(n.eq.12488) write(97,*) q(kk,3)/q(kk,1)   ,3.0-x(kk)*L*100.
        if(n.eq.279*45) write(98,*) q(kk,3)/q(kk,1)   ,3.0-x(kk)*L*100.
        if(n.eq.12645) write(99,*) q(kk,3)/q(kk,1)   ,3.0-x(kk)*L*100.
        if(n.eq.283*45) write(80,*) q(kk,3)/q(kk,1)   ,3.0-x(kk)*L*100.
        if(n.eq.12803) write(81,*) q(kk,3)/q(kk,1)   ,3.0-x(kk)*L*100.
        if(n.eq.286*45) write(82,*) q(kk,3)/q(kk,1)   ,3.0-x(kk)*L*100.
        if(n.eq.12938) write(83,*) q(kk,3)/q(kk,1)   ,3.0-x(kk)*L*100.
        if(n.eq.289*45) write(84,*) q(kk,3)/q(kk,1)   ,3.0-x(kk)*L*100.
        if(n.eq.600*90) write(85,*) q(kk,3)/q(kk,1)   ,3.0-x(kk)*L*100.
c
cc     Induced Flow History in Boundary Layer
c
        if(n.eq.1) write(941,*)  q(kk,2)/q(kk,1)       ,3.0-x(kk)*L*100.
        if(n.eq.269*45) write(942,*) q(kk,2)/q(kk,1)   ,3.0-x(kk)*L*100.
```

```
        if(n.eq.12195) write(943,*) q(kk,2)/q(kk,1)   ,3.0-x(kk)*L*100.
        if(n.eq.273*45) write(944,*) q(kk,2)/q(kk,1)   ,3.0-x(kk)*L*100.
        if(n.eq.12323) write(945,*) q(kk,2)/q(kk,1)   ,3.0-x(kk)*L*100.
        if(n.eq.276*45) write(946,*) q(kk,2)/q(kk,1)   ,3.0-x(kk)*L*100.
        if(n.eq.12488) write(947,*) q(kk,2)/q(kk,1)   ,3.0-x(kk)*L*100.
        if(n.eq.279*45) write(948,*) q(kk,2)/q(kk,1)   ,3.0-x(kk)*L*100.
        if(n.eq.12645) write(949,*) q(kk,2)/q(kk,1)   ,3.0-x(kk)*L*100.
        if(n.eq.283*45) write(951,*) q(kk,2)/q(kk,1)   ,3.0-x(kk)*L*100.
        if(n.eq.12803) write(952,*) q(kk,2)/q(kk,1)   ,3.0-x(kk)*L*100.
        if(n.eq.286*45) write(953,*) q(kk,2)/q(kk,1)   ,3.0-x(kk)*L*100.
        if(n.eq.12938) write(954,*) q(kk,2)/q(kk,1)   ,3.0-x(kk)*L*100.
        if(n.eq.289*45) write(955,*) q(kk,2)/q(kk,1)   ,3.0-x(kk)*L*100.
        if(n.eq.600*45) write(956,*) q(kk,2)/q(kk,1)   ,3.0-x(kk)*L*100.
c
cc     Density History in Boundary Layer
c
        if(n.eq.1) write(891,*)  q(kk,1)       ,3.0-x(kk)*L*100.
        if(n.eq.269*45) write(892,*) q(kk,1)   ,3.0-x(kk)*L*100.
        if(n.eq.12195) write(892,*) q(kk,1)   ,3.0-x(kk)*L*100.
        if(n.eq.273*45) write(893,*) q(kk,1)   ,3.0-x(kk)*L*100.
        if(n.eq.12323) write(892,*) q(kk,1)   ,3.0-x(kk)*L*100.
        if(n.eq.276*45) write(894,*) q(kk,1)   ,3.0-x(kk)*L*100.
        if(n.eq.12488) write(892,*) q(kk,1)   ,3.0-x(kk)*L*100.
        if(n.eq.279*45) write(895,*) q(kk,1)   ,3.0-x(kk)*L*100.
        if(n.eq.12645) write(892,*) q(kk,1)   ,3.0-x(kk)*L*100.
        if(n.eq.283*45) write(896,*) q(kk,1)   ,3.0-x(kk)*L*100.
        if(n.eq.12803) write(892,*) q(kk,1)   ,3.0-x(kk)*L*100.
        if(n.eq.286*45) write(897,*) q(kk,1)   ,3.0-x(kk)*L*100.
        if(n.eq.12938) write(892,*) q(kk,1)   ,3.0-x(kk)*L*100.
        if(n.eq.289*45) write(898,*) q(kk,1)   ,3.0-x(kk)*L*100.
        if(n.eq.600*45) write(899,*) q(kk,1)   ,3.0-x(kk)*L*100.
c
cc     Pressure History in Boundary Layer
c
        if(n.eq.1) write(791,*)  p(kk)*rh2*a2*a2/p2     ,3.0-x(kk)*L*100.
        if(n.eq.269*45) write(792,*) p(kk)*rh2*a2*a2/p2   ,3.0-x(kk)*L*100.
        if(n.eq.12195) write(792,*) p(kk)*rh2*a2*a2/p2   ,3.0-x(kk)*L*100.
        if(n.eq.273*45) write(793,*) p(kk)*rh2*a2*a2/p2   ,3.0-x(kk)*L*100.
        if(n.eq.12323) write(792,*) p(kk)*rh2*a2*a2/p2   ,3.0-x(kk)*L*100.
        if(n.eq.276*45) write(794,*) p(kk)*rh2*a2*a2/p2   ,3.0-x(kk)*L*100.
        if(n.eq.12488) write(792,*) p(kk)*rh2*a2*a2/p2   ,3.0-x(kk)*L*100.
        if(n.eq.279*45) write(795,*) p(kk)*rh2*a2*a2/p2   ,3.0-x(kk)*L*100.
        if(n.eq.12645) write(792,*) p(kk)*rh2*a2*a2/p2   ,3.0-x(kk)*L*100.
        if(n.eq.283*45) write(796,*) p(kk)*rh2*a2*a2/p2   ,3.0-x(kk)*L*100.
        if(n.eq.12803) write(792,*) p(kk)*rh2*a2*a2/p2   ,3.0-x(kk)*L*100.
        if(n.eq.286*45) write(797,*) p(kk)*rh2*a2*a2/p2   ,3.0-x(kk)*L*100.
        if(n.eq.12938) write(792,*) p(kk)*rh2*a2*a2/p2   ,3.0-x(kk)*L*100.
        if(n.eq.289*45) write(798,*) p(kk)*rh2*a2*a2/p2   ,3.0-x(kk)*L*100.
        if(n.eq.600*45) write(799,*) p(kk)*rh2*a2*a2/p2   ,3.0-x(kk)*L*100.
  418 continue
   30 CONTINUE
      close(810)
      close(820)
      close(830)
      close(840)
```

```
      close(850)
      close(855)
      close(710)
      close(720)
      close(730)
      close(740)
      close(750)
      close(755)
      close(910)
      close(920)
      close(930)
      close(940)
      close(950)
      close(53)
      close(621)
      close(622)
      close(623)
      close(631)
      close(632)
      close(633)
      close(634)
      close(770)
      close(771)
      close(772)
      close(773)
      close(774)
      close(775)
      close(776)
      close(777)
      close(778)
      close(779)
      close(780)
      close(781)
      close(762)
      close(763)
      close(764)
      close(782)
      close(783)
      close(784)
      close(785)
      close(786)
      close(787)
      close(788)
      close(789)
      close(790)
      close(791)
      close(792)
      close(793)
      close(794)
      close(795)
      close(796)
c     close(797)
c     close(798)
c     close(799)
      close(497)
```

```
      close(498)
      STOP
      END
      SUBROUTINE FCT(NXM,n,ETA,ET2,DTR,Q,DF,U)
c
c       APPLIES FLUX-CORRECTED TRANSPORT ALGORITHM TO Q(J,K)
c
      double precision   Q(19400,4),DQ(19400,4),U(19400),
     *df(19400,4),adf(19400,4),eta,et1,et2,emu,enu
c
c       COMPUTE ANTIDIFFUSIVE FLUXES
c
      EMU = ETA + 0.25*ET2*((U(1)+U(2))*DTR)**2
      nxx = nx-5
      jj = 1500
      DO 1 kK=1,4
    1    ADF(1,kK) = EMU*(Q(2,kK)-Q(1,kK))
         DO 3 J=2,NXX
           EMU = ETA + 0.25*ET2*((U(J)+U(J+1))*DTR)**2
c           emu = 0.125
           DO 2 kK=1,4
c           if(float(n/2.).ne.int(n/2.))ADF(J,kK) = EMU*(Q(J,kK) - Q(j-1,kK))
           ADF(J,kK) = EMU*(Q(J+1,kK) - Q(J,kK))
c
c       DIFFUSE THE SOLUTION
c
         Q(J,kK) = Q(J,kK) + DF(J,kK) - DF(J-1,kK)
    2    DQ(J-1,kK) = Q(J,kK) - Q(J-1,kK)
    3 CONTINUE
      DO 4 kK=1,4
    4      DQ(NXM-1,kK) = Q(NXM,kK) - Q(NXM-1,kK)
c
c       LIMIT ANTIDIFFUSIVE FLUXES
c
      DO 6 J=2,NXX
         DO 5 kK=1,4
           S = SIGN(1.0,ADF(J,kK))
           ADF(J,kK) = ABS(ADF(J,kK))
           DUM = S*DQ(J-1,kK)
           ADF(J,kK) = AMIN1(ADF(J,kK),DUM)
           DUM = S*DQ(J+1,kK)
           ADF(J,kK) = AMIN1(ADF(J,kK),DUM)
           ADF(J,kK) = AMAX1(ADF(J,kK),0.)
           ADF(J,Kk) = S*ADF(J,kK)
c
c       ANTIDIFFUSE THE SOLUTION
c
           Q(J,kK) = Q(J,kK) - ADF(J,kK) + ADF(J-1,kK)
    5    CONTINUE
    6 CONTINUE
      RETURN
      END
_
```

VITA

Terry Vincent Reid was born on August 5, 1963 in Cleveland, Ohio. He graduated from Beachwood High School in 1981. He then went to The University of Akron, in Akron Ohio on a Track and Field Scholarship. After graduating in 1986, he worked at the Factory Mutual Engineering Association - Cleveland District Office. There, he evaluated private industrial facilities for fire and explosion hazards for a variety of factories in Ohio, Indiana and Kentucky. He left this job in 1990 to return to graduate school at The University of Akron. While there, he co-directed a newly formed Minority Retention Program, which was designed to increase the retention rate of undergraduate minority engineering students. He received his Master's degree in Mechanical Engineering from The University of Akron December of 1992. He then headed to Virginia Tech to pursue his PhD in Mechanical Engineering.