2

WL-TR-91-2011

AD-A236 861

TURBULENT SWIRLING FLOW IN COMBUSTOR/EXHAUST NOZZLE SYSTEMS

WRIGHT LABORATORY

K. M. Peery
S. T. Imlay
M. Soetrisno
Amtec Engineering, Inc.
3055 112th Avenue NE, Suite 208
Bellevue, Washington 98004

29 March 1991

Final Report for Period May 1985 - January 1991

AERO PROPULSION AND POWER DIRECTORATE
WRIGHT RESEARCH DEVELOPMENT CENTER
AIR FORCE SYSTEMS COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433-6563

[illegible]

Abdollah S. Nejad
Aerospace Engineer
Experimental Research Branch
Advanced Propulsion Division
Aero Propulsion & Power Directorate

John T. Hojnacki

John T. Hojnacki, Chief
Experimental Research Branch
Advanced Propulsion Division
Aero Propulsion & Power Directorate

FOR THE COMMANDER

[illegible]

William G. Beecroft
Director
Advanced Propulsion Division
Aero Propulsion & Power Directorate

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188
Exp. Date: Jun 30, 1986

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release distribution is unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S) WL-TR-91-2011
6a. NAME OF PERFORMING ORGANIZATION Amtec Engineering Inc. 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION Aero Propulsion and Power Dir (WL/POPT) Wright Laboratory
6c. ADDRESS (City, State, and ZIP Code) 3055 112th Ave. NE, Suite 208 Bellevue, Washington 98004	7b. ADDRESS (City, State, and ZIP Code) Wright-Patterson AFB Dayton, Ohio 45433-6563
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Aero Propulsion Laboratory 8b. OFFICE SYMBOL (If applicable) WL/POPT	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F33615-85-C-2588
8c. ADDRESS (City, State, and ZIP Code) Wright-Patterson AFB, Ohio 45433-6563	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. 61102F; PROJECT NO. 2308; TASK NO. S1; WORK UNIT ACCESSION NO. 14

11. TITLE (Include Security Classification)
Turbulent Swirling Flow in Combustor/Exhaust Nozzle Systems

12. PERSONAL AUTHOR(S)
Kelton M. Peery, Scott I. Imlay, and M. Soetrisno

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Final	FROM MAY 85 TO JAN 91	91 MAR 29	151

16. SUPPLEMENTARY NOTATION

17. COSATI CODES		
FIELD	GROUP	SUB-GROUP

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
Ramjet, Dump Combustro, Viscous Swirling Flow, CFD Turbulence Modelling, Time Dependent Calculations Flux Vector Spliting, Total Variational Diminishing.

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

A Computer code named DUMPSTER was developed for the analysis of the viscous swirling nonreacting flow in dump combustors closely coupled to convergent-divergent nozzel. The development, application, and use of this code is described in this report.

DUMPSTER solves the 2D/axisymmetric compressible Navier-Stokes equation with one or more transport equations for trace chemical speicies. Turbulence is modeled with either an algebraic model or a two-equation K-E model. The equations are solved using a time-dependent finite-volume method. The fluxes are calculated using one of the following second-order accurate flux functions: 1) Steger-Warming flux vector splitting, 2) Roe's flux differnce splitting, or 3) Harten-yee's Total Variation Diminishing (TVD) Flux.

The results of the code agree reasonably well with dump combustor experimental data and other numerical simulation programs.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT [✓ UNCLASSIFIED/UNLIMITED] ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified	
22a. NAME OF RESPONSIBLE INDIVIDUAL Abdollah S. Nejad	22b. TELEPHONE (Include Area Code) 513-255-1234	22c. OFFICE SYMBOL WL/POPT

DD FORM 1473, 84 MAR
83 APR edition may be used until exhausted
All other editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE
Unclassified

Summary

A computer code named DUMPSTER was developed for the analysis of the viscous swirling nonreacting flow in dump combustors closely coupled to convergent-divergent nozzle. The development, application, and use of this code is described in this report.

DUMPSTER solves the 2D/axisymmetric compressible Navier-Stokes equations with one or more transport equations for trace chemical species. Turbulence is modeled with either an algebraic model or a two-equation k–ϵ model. The equations are solved using a time-dependent finite-volume method. The fluxes are calculated using one of the following second-order accurate flux functions: 1) Steger-Warming flux vector splitting, 2) Roe's flux difference splitting, or 3) Harten-Yee's Total Variation Diminishing (TVD) Flux.

Both a line Gauss-Siedel (LGS) and a lower-upper symmetric Gauss-Siedel (LU- SGS) methods were studied. The LGS method was capable of converging to steady state more rapidly than the LU-SGS, but required more numerical dissipation in the flux computations. The LU-SGS was selected as the best solution procedure.

The computational mesh consists of one or more blocks or zones of mesh cells stacked upon each other to make mesh generation easy. The code reads input data in a NAMELIST fashion that allows long English-like variable names.

The results of the code agree reasonably well with dump combustor experimental data and another numerical flow simulation program. The principal limit to improving the accuracy of the calculation is shown to be the modeling of turbulence.

Contents

List of Figures

1 INTRODUCTION

The design of practical combustors has been and still is for the most part an art[1]. Traditional design procedures are based on "cut-and-try" methodologies and are guided by experienced combustor designers. The complexity of the fluid dynamics and chemistry has defied rigorous detailed numerical simulation. In practical applications even axisymmetric dump combustors as shown in Figure 1 exhibit very complex flow fields. They usually have three-dimensional characteristics due to swirler vanes or injection of fuel at discrete locations. The flow may have several gaseous components, liquid droplets, and solid particles that physically and chemically react. Thermal radiation may be important with chemical reaction. Swirl flows can create a central toroidal recirculation zone (CTRZ) in addition to the corner recirculation zone (CRZ) and hub recirculation zone (HRZ). In this study the effects of a closely coupled CD nozzle also complicates the flow field. Experimental studies[2, 3] show that the existence, shape, and strength of the CTRZ is strongly influenced by details of the combustor geometry, upstream flow conditions, downstream flow conditions, combustion, and amount of swirl.

The sudden expansion dump combustor illustrated in Figure 1 is favored for use in ramjet propulsion systems. The dump combustor is simple and can be designed to be short and compact, especially if swirl is used to stabilize the combustion flame. Swirl is induced into the flow entering the combustor by use of turning vanes or tangential injection of air. The amount of swirl is measured by the swirl number (S) which is

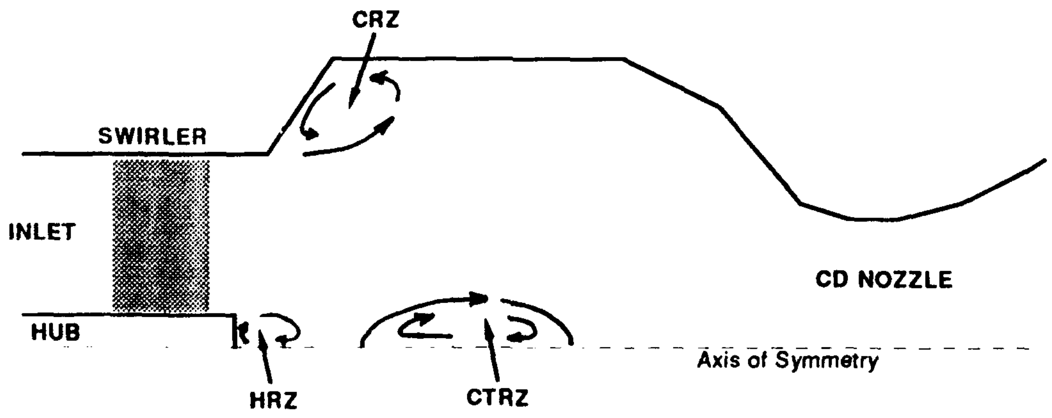

Figure 1: Schematic Diagram of a Dump Combustor Flow Field

defined[2] as

$$S = \frac{\int_{R_i}^{R_o} uwr^2}{R \int_{R_i}^{R_o} u^2 r dr} \approx \frac{2}{3} \tan(\theta_{\text{swirl}})$$

The swirl number is approximately equal to two-thirds of the tangent of the swirl vane angle (θ_{swirl}). Swirl has a strong effect upon the combustor flow field[3]. At low degrees of swirl ($S < 0.4$) the width of the air jet is increased over that without swirl. At larger swirl numbers the mixing rate of the jet increases. At swirl numbers of approximately 0.6 and greater a CTRZ forms. Formation of the CTRZ is usually described as "vortex breakdown" or a transition from supercritical swirl flow to subcritical swirl flow[3, 4]. The CTRZ can be designed to be very stable and to provide flame holding capability. The reverse flow mass flowrate in the CTRZ can be larger than the mass flowrate in the primary air jet. The enhanced mixing rate in the CTRZ improves combustion and thus allows the length of the dump combustor to be shortened. This saves weight and space which can be important in designing propulsion systems. In addition, using the CTRZ for flameholding has advantages over bluff body flame holders and simple dump combustors (without swirl) in that the hot region of the flow is not near a wall thereby reducing cooling requirements.

Making dump combustors shorter increases the influence of the exhaust nozzle upon the combustor flow. Particularly for high swirl numbers the exhaust nozzle can have a substantial effect on the CTRZ. See for example the experimental work of Escudier and Keller[4] showing substantial changes in axial velocity profiles and shape of the CTRZ with only a 15% reduction in exit diameter for high swirl number in a dump combustor. Also Lilley[5] and Yoon and Lilley[6] report effects of downstream geometry contractions on the CTRZ. Their experimental results showed that the length of the CTRZ was shortened by the contraction and the shape of the CTRZ substantially changed.

Several computer codes have been developed for simulating the fluid flow in axisymmetric dump combustors. A review of some of these is given by Lilley[7]. More calculations have been presented recently by Lilley[8], Abujelala[9], Ramos and Somer[10], and Rhode and Stowers[11] and others. Nearly all of these recent works have used a variation of the TEACH code[12] or a code using a similar solution strategy. In these codes a relaxation procedure is used to solve the steady time averaged Navier-Stokes equations with models for turbulent viscous stresses. The strong streamline curvature and recirculation regions present in the dump combustor requires that the full set of elliptic equations be solved (as opposed to a spacial marching solution of a parabolized subset of these equations).

Typically (as done in STARPIC[8]) the flowfield variables are over specified at the inflow boundary and set by zero axial gradient at the outflow boundary. Since the equations in nearly all of these above mentioned codes are essentially incompressible these boundary conditions can lead to correct and unique physical solutions. Applications of these codes to compressible flows, however, can lead to nonphysical solutions, since the boundary conditions are not consistent with method-of-characteristic theory and the absence of density variations (due to pressure changes) is inconsistent with transonic flow. Take for example the flow through a choked CD (convergent-divergent) nozzle. The specification of all flow field variables at the inflow boundary determines the net mass flow rate. This mass flow rate will not necessarily match the critical mass flow rate through the CD nozzle. Information in the form of acoustic waves must propagate upstream from the throat of the CD nozzle and adjust the flow conditions at the inflow boundary. A realistic boundary condition would be to specify total pressure and total temperature at the inflow boundary. Acoustic waves reaching the inflow boundary would change the local velocity and thus the local static pressure and temperature of the flow until the physically correct steady-state mass flow rate was entering the combustor.

Most flow analysis codes currently being used for computing steady turbulent non-reacting flows in axisymmetric dump combustors are based upon the mathematical model and solution procedure of the TEACH code. The specific problem addressed in this work arises from the fact that these computer codes can not be applied to the calculation of compressible transonic and supersonic flow in a closely coupled dump combustor and exhaust nozzle configuration due to inappropriate boundary conditions and lack of compressibility effects.

Numerous compressible flow Navier-Stokes codes do exist that have been applied routinely to the computation of flow through CD nozzles yielding accurate solutions. See for example Peery and Forester[13] and Cline[14]. Codes of this type are based upon time-dependent methods of solving the unsteady mass-averaged Navier-Stokes equations. Steady-state solutions are obtained by integrating the time-dependent equations until the solution approaches sufficiently close to the asymptotic stead-state limit. Most of these so called time-dependent codes, however, do not include swirl or advanced turbulence models. In addition, most of these time-dependent codes have not been applied routinely to low-speed recirculating flows. The reason for this is that most of these codes have numerical stability constraints that limit the size of the time step to a CFL number[1] of about 1 or achieve maximum convergence rate (to steady state) at a CFL number of about 1. These codes have used various explicit, semi-implicit, and ADI implicit solution procedures. To reach steady-state in a low-speed recirculating flow requires thousands if not tens of thousands of time steps[15] using these time-dependent codes.

Recently progress has been made in improving the computational efficiency and accuracy of solution procedures for the compressible Navier-Stokes equations. Several investigators including MacCormack[16], Thomas and Walters[17], Chakravarthy[18]

[1]CFL number is the ratio of the acoustic length scale $(u + c_s)\delta t$ to the cell size Δx

have used line Gauss-Seidel (LGS) relaxation methods with second or third-order flux-split differencing on the convection and pressure terms and second-order central differencing on diffusion terms. The methods are unconditionally stable on model equations. Convergence rates to steady state have been shown to be much less sensitive to the time step size than the popular approximately-factored (AF) implicit methods (*e.g.* Beam and Warming[19]). Once past starting transients MacCormack's method has been run with CFL numbers of one billion, thus reducing to a modified Newton iteration scheme. Convergence to steady state for some calculations[16, 17] can require as few as 20 time steps. More recently Yoon[20] has developed a Lower-Upper Symmetric Gauss Seidel (LU-SGS) solution procedure that converges rapidly and is computationally efficient. As described in this report the LU-SGS scheme makes it possible to create a flow analysis that has less numerical dissipation than the LGS methods mentioned above. Both the LGS and the LU-SGS methods were studied and compared for use in calculating dump combustor/CD nozzle flows.

In this work a computer program was developed called DUMPSTER. It solves the Navier-Stokes equations for axisymmetric flows in dump combustors and nozzles. The flow analysis is applicable to flows with low subsonic, transonic, and supersonic regions. It optionally solves one or more transport equations for trace chemical species. Turbulence is modeled by either the Baldwing-Lomax[21] algebraic model or Launder and Spalding's two-equation k-ϵ model[22]. The computational grid is constructed from stacking one or more computational blocks (or zones) to simplify the specification and generation of the computational mesh as well as efficiently utilize all of the computational cells. DUMPSTER was applied to several dump combustor flows that also had experimental measurements to which the calculated flows were compared. Results are also compared to calculations performed with a different computer flow analysis program.

2 MATHEMATICAL MODEL

2.1 Navier-Stokes Equations

The DUMPSTER code solves the Reynolds-averaged Navier-Stokes equations, trace species transport equations, and an optional two-equation turbulence model. These equations are given below in integral form[23] for a cylinderical coordinate system.

$$\frac{\partial}{\partial t}\iiint_{\mathcal{V}} U \, d\mathcal{V} + \iint_{S} \vec{P}\cdot\vec{n} \, dS + \iiint_{\mathcal{V}} N \, d\mathcal{V} = 0 \tag{1}$$

In the above equation:

t	is the time.
$\mathcal{V}$	is the volume of the domain.
U	is the solution state vector u.
S	is the surface containing the volume.
$\vec{P}$	is the flux vector.
$\vec{n}$	is the unit normal vector pointing outward from surface S.
N	is the source term that arises from cylinderical formulations.

The state vector U contains five or more variables depending upon how many trace chemical species are being calculated and which turbulence model is used. When the k–ϵ turbulence model and n chemical species are being calculated, U is defined as:

$$U = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ E \\ \rho k \\ \rho\epsilon \\ \rho f_1 \\ \vdots \\ \rho f_n \end{pmatrix}$$

The flux vector $\vec{P}$ is defined as

$$\vec{P} = F_x\vec{i} + F_r\vec{j}$$

where $\vec{i}$ and $\vec{j}$ are unit vectors pointing in the axial (x) and the radial (r) directions, respectively. F_x and F_r are the axial and radial components of the flux vector defined below.

$$F_x = \begin{pmatrix} \rho u \\ \rho uu + P + \tau_{xx} \\ \rho uv + \tau_{xr} \\ \rho uw + \tau_{x\theta} \\ u(E+P) + u\tau_{xx} + v\tau_{rx} + w\tau_{\theta x} + q_x \\ \rho ku - \Gamma_k(\partial k/\partial x) \\ \rho \epsilon u - \Gamma_\epsilon(\partial \epsilon/\partial x) \\ \rho f_1 u - \Gamma_{f_1}(\partial f_1/\partial x) \\ \rho f_2 u - \Gamma_{f_2}(\partial f_2/\partial x) \\ \vdots \\ \rho f_n u - \Gamma_{f_n}(\partial f_n/\partial x) \end{pmatrix}$$

$$F_r = \begin{pmatrix} \rho v \\ \rho vu + \tau_{rx} \\ \rho vv + \tau_{rr} + P \\ \rho vw + \tau_{r\theta} \\ v(E+P) + u\tau_{xr} + v\tau_{rr} + w\tau_{\theta r} + q_r \\ \rho kv - \Gamma_k(\partial k/\partial r) \\ \rho \epsilon v - \Gamma_\epsilon(\partial \epsilon/\partial r) \\ \rho f_1 v - \Gamma_{f_1}(\partial f_1/\partial r) \\ \rho f_2 v - \Gamma_{f_2}(\partial f_2/\partial r) \\ \vdots \\ \rho f_n v - \Gamma_{f_n}(\partial f_n/\partial r) \end{pmatrix}$$

The source term N arises in the cylinderical formulation of the equations and is defined below.

$$N = \begin{pmatrix} 0 \\ 0 \\ (-\rho w w - P - \tau_{\theta\theta})/r \\ (\rho v w + \tau_{r\theta})/r \\ 0 \\ \rho\epsilon - G_k \\ -(C_{\epsilon 1}\,\epsilon\, G_k - C_{\epsilon 2}\,\rho\,\epsilon^2)/k \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

The definitions of some of the terms and symbols used above are defined as follows:

$$\begin{aligned}
u &= x\text{-component of velocity} \\
v &= r\text{-component of velocity} \\
w &= \text{swirl-component of velocity} \\
e &= \text{specific internal energy} \\
\rho &= \text{mass density} \\
P &= \text{pressure} \\
k &= \text{turbulent kinetic energy} \\
\epsilon &= \text{turbulent energy dissipation rate} \\
E &= \rho e + \frac{\rho}{2}(u^2 + v^2 + w^2) \\
\tau_{xx} &= -\Gamma_M \left(2\frac{\partial u}{\partial x} - \frac{2}{3}\nabla \cdot \vec{V}\right) \\
\tau_{rr} &= -\Gamma_M \left(2\frac{\partial v}{\partial r} - \frac{2}{3}\nabla \cdot \vec{V}\right) \\
\tau_{\theta\theta} &= -\Gamma_M \left(2\frac{v}{r} - \frac{2}{3}\nabla \cdot \vec{V}\right) \\
\tau_{r\theta} &= -\Gamma_M \left(\frac{\partial w}{\partial r} - \frac{w}{r}\right) \\
\tau_{\theta r} &= \tau_{r\theta} \\
\tau_{x\theta} &= -\Gamma_M \left(\frac{\partial w}{\partial x}\right) \\
\tau_{\theta x} &= \tau_{x\theta} \\
\tau_{xr} &= -\Gamma_M \left(\frac{\partial u}{\partial r} - \frac{\partial v}{\partial x}\right)
\end{aligned}$$

$$\begin{aligned}
\nabla \cdot \vec{V} &= \frac{1}{r}\frac{\partial r v}{\partial r} + \frac{\partial u}{\partial x} \\
\vec{V} &= u\vec{i} + v\vec{j} + w\vec{k} \\
q_x &= -\Gamma_E(\partial T/\partial x) \\
q_r &= -\Gamma_E(\partial T/\partial r)
\end{aligned}$$

Γ_M is the effective absolute viscosity defined as the sum of the laminar and turbulent viscosities.

$$\Gamma_M = \mu_l + \mu_t$$

Γ_E is the effective thermal conductivity defined as the sum of the laminar and turbulent viscosities.

$$\Gamma_E = k_l + k_t$$

Γ_{f_i} is the effective diffusion coefficient for species i defined as

$$\Gamma_{f_i} = \Gamma_M/\sigma_{f_i}$$

where σ_{f_i} is the schmidt number for chemical species i.

There are two options for turbulence models: the Baldwin-Lomax model and the k-ϵ two-equation model. In the Baldwin Lomax model the turbulent eddy viscosities μ_t are calculated from algebraic relations based upon local quantities of the flow field. When the k-ϵ turbulence model is used, two transport equations in Equations 1 are solved. The pertinent constants and relations for this model are as follows:

$$G_k = \mu_t\left(2\left(\frac{\partial u}{\partial x}\right)^2 + 2\left(\frac{\partial v}{\partial r}\right)^2 + 2\left(\frac{v}{r}\right)^2 + \left(\frac{\partial u}{\partial r} + \frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial w}{\partial x}\right)^2 + \left(\frac{\partial w}{\partial r} - \frac{w}{r}\right)^2\right)$$

and the constants are defined as

$$\begin{aligned}
\mu_t &= C_\mu \frac{\rho k^2}{\epsilon} \\
\Gamma_k &= \frac{\mu_t}{\sigma_k} \\
\Gamma_\epsilon &= \frac{\mu_t}{\sigma_\epsilon} \\
C_{\epsilon 1} &= 1.44 \\
C_{\epsilon 2} &= 1.92
\end{aligned}$$

$$\begin{aligned} C_\mu &= 0.09 \\ \sigma_k &= 1.0 \\ \sigma_\epsilon &= 1.3 \end{aligned}$$

The values of $C_{\epsilon 1}$, $C_{\epsilon 2}$, σ_k, σ_ϵ, and C_μ may be specified by the user. The above values are the defaults.

Equation 1 represents a very simple idea—the time rate of change of mass, momentum, and energy within an arbitrarily chosen volume, $\mathcal{V}$, is equal to the apparent flux of these quantities inward through the surface, S, surrounding the volume plus the production of these quantities within the volume. The finite volume method consists of breaking the flow field up into a large number of quadrilateral finite volume cells to which the integral equations are applied.

2.2 Thermodynamics

The equations in the previous section must be supplemented by an equation of state to calculate P and T from ρ and e. The DUMPSTER code offers three options for equations of state: 1) perfect gas, 2) thermally perfect gas, 3) and Tannehill's[24] curve fits for equilibrium air.

In addition to calculating P and T, it is necessary to calculate several other thermodynamic variables which are needed to calculate the numerical fluxes. The first two are γ and $\bar{\gamma}$. For perfect gas flows γ and $\bar{\gamma}$ are identical and are equal to the ratio of specific heats, $\frac{C_p}{C_v}$. For real gases, however, the ratio of specific heats no longer has any physical significance and the quantities γ and $\bar{\gamma}$ are defined to simplify the calculation of numerical fluxes. In particular we use Vinokur's definition[25],

$$\begin{aligned} \gamma &= \frac{\rho c_s^2}{P} \\ \bar{\gamma} &= 1 + \frac{P}{\rho e} \end{aligned}$$

where c_s is the isentropic speed of sound. For numerical fluxes based on Roe's flux function Vinokur has defined the alternative variables κ and χ which are the partial derivatives of pressure P with respect to internal energy per unit volume $\tilde{e} = \rho e$ and density ρ.

$$\begin{aligned}\kappa &= \left.\frac{\partial P}{\partial \tilde{e}}\right|_{\rho} \\ \chi &= \left.\frac{\partial P}{\partial \rho}\right|_{\tilde{e}}\end{aligned}$$

2.2.1 Perfect Gas

The perfect gas equation of state is expressed with the following equations:

$$\begin{aligned}P &= (\bar{\gamma}-1)\,\rho e = \bar{\gamma} RT \\ T &= e/C_v\end{aligned}$$

The equation of state can be altered for different gases by specifying the ratio of specific heats, $\bar{\gamma}$, and the gas constant, R. The gas constant for a gas of a given molecular weight, $\mathcal{M}$, is calculated from the universal gas constant $\tilde{R} = 8.31434$J/mole-K using the formula $R = \tilde{R}/\mathcal{M}$. The five thermodynamic variables are given by the following formulas.

$$\begin{aligned}\bar{\gamma} &= \gamma = \text{constant} \\ \kappa &= \gamma - 1 \\ \chi &= 0 \\ C_v &= \frac{R}{\bar{\gamma}-1}\end{aligned}$$

2.2.2 Thermally Perfect Gas

The second option for the thermodynamic equation of state is a thermally perfect gas. For this gas the specific internal energy (e) is not linearly proportional to the temperature. In general the internal energy and temperature relate as follows:

$$e = e(T) = C_v(T)T$$

where $C_v(T)$ is the temperature-dependent specific heat capacity at constant volume. For a perfect gas C_v is a constant and the temperature can be easily calculated from the internal energy. But for a thermally perfect gas where C_v is a function of temperature, calculating temperature from internal energy requires iteration.

In DUMPSTER the user inputs a curve fit for the specific heat at constant volume as a function of absolute temperature. This curve fit has the following form:

$$C_v(T) = c_1 T^{c_2} + c_3 T + c_4$$

To avoid an iterative process every time temperature is decoded from internal energy, a mean specific heat capacity is used in DUMPSTER which defined as the ratio of the temperature to internal energy.

$$\bar{C_v} = e/T = \frac{1}{T}\int_{T_o}^{T} C_v(T)\,dT$$

where T_o is a reference temperature arbitrarily set to absolute zero ($T_o = 0$). An expression for the mean specific heat capacity at constant volume can be obtained by integrating the curve fit function for specific heat capacity with respect to temperature as follows:

$$\bar{C}_v(T) = \left(\frac{c_1}{c_2+1}\right)T^{c_2+1} + \frac{c_3}{2}T^2 + c_4 T$$

Temperature is decoded by dividing the internal energy by the mean specific heat.

$$T = \frac{e}{\bar{C_v}}$$

C_v is then updated at the new temperature for use in the next time step. As the solution converges so does the value for $\bar{C}_v$. The speed of sound (c_s) used for the calculation of γ is the isentropic speed of sound defined by

$$c_s^2 = \left(\frac{\partial P}{\partial \rho}\right)_s$$

Using this and previously given definitions, the resulting expressions for a γ and $\bar{\gamma}$ for a thermally perfect gas are

$$\begin{aligned} \gamma &= 1 + \frac{R}{C_v} \\ \bar{\gamma} &= 1 + \frac{R}{\bar{C}_v} \end{aligned}$$

The pressure derivatives, κ and χ, are needed for the Roe and Harten-Yee flux functions are defined as follows:

$$\begin{aligned}\kappa &= \frac{R}{C_v} \\ \chi &= RT\left(1 - \frac{\bar{C}_v}{C_v}\right)\end{aligned}$$

2.2.3 Equilibrium Air Curve Fits

The third option for the equation of state is Tannehill's curve fits for real air. With this option the code calls a subroutine given in reference[24] to calculate P and T from ρ and e for each finite-volume cell and each time step. This routine returns values for P, T, and c_s as a function of ρ and e for air in thermochemical equilibrium up to 25,000 K.

Since $e_0 = 0$, three variables can be calculated directly from their definitions as follows:

$$\begin{aligned}\gamma &= \frac{\rho c_s^2}{P} \\ \bar{\gamma} &= 1 + \frac{P}{\rho e} \\ \bar{C}_v &= \frac{e}{T}\end{aligned}$$

The pressure derivatives, κ and χ, must be calculated using data numerical differentiation using data from the real gas curve fit routines. To evaluate κ and χ, start with the chain rule

$$\delta P = \left(\frac{\partial P}{\partial \tilde{e}}\right)_\rho + \delta \tilde{e}\left(\frac{\partial P}{\partial \rho}\right)_{\tilde{e}} \delta\rho = \kappa\delta\tilde{e} + \chi\delta\rho$$

and the expression for speed of sound in terms of κ and χ as described by Vinokur[25, Eq. 5] (where h is the enthalpy)

$$c_s^2 = \chi + \kappa h$$

Eliminating χ from the two equations above and solving for κ yields

$$\begin{aligned}\kappa &= \frac{\delta P - c_s^2\delta\rho}{\delta\tilde{e} - h\delta\rho} \\ &= \frac{P(\rho+\delta\rho, \tilde{e}+\delta\tilde{e})P(\rho,\tilde{e}) - c_s^2\delta\rho}{\delta\tilde{e} - h\delta\rho}\end{aligned}$$

This expression may be used to calculate κ from any combination of $\delta\rho$ and $\delta\tilde{e}$. After some experimentation, $\delta\tilde{e} = 2h\delta\rho$ was chosen since it seems to avoid most spurious results near discontinuities in the curve fits. The other pressure derivative, χ, is then calculated from $\chi = c_s^2 - \kappa h$.

2.3 Transport Properties

The effective viscosity Γ_M and the effective conductivity Γ_E are defined as the sum of the laminar and turbulent components as follows:

$$\begin{aligned}\Gamma_M &= \mu_l + \mu_t \\ \Gamma_E &= k_l + k_t\end{aligned}$$

The diffusion coefficients are computer from Γ_M using Schmidt numbers. These quantities are calculated within DUMPSTER using various user specified methods as described below.

2.3.1 Viscosity

The laminar viscosity is calculated using one of three methods. The first method is to set the laminar viscosity to a constant. The second method is to assume that the viscosity depends only on temperature and that the dependence may be approximated by the following formula.

$$\mu_l = \frac{c_1 T^{c_2} + c_3 T + c_4}{c_5 T + c_6}$$

Sutherland's relation for viscosity is obtained by setting the constants in the above formula as follows where the temperature (T) is in units of degrees Kelvin and the viscosity is in units of $Kg/(m \cdot s)$.

$$c_1 = 1.451 \times 10^{-6}$$

$$\begin{aligned}c_2 &= 1.5\\ c_3 &= 0\\ c_4 &= 0\\ c_5 &= 1\\ c_6 &= 110\end{aligned}$$

The laminar viscosity may also be calculated using the equilibrium air curve fits of Tannehill[26]. A viscosity is returned when the Tannehill routines are called with the density and internal energy.

2.3.2 Conductivity

There are three methods for calculating the thermal conductivity. One method is to calculate the thermal conductivities from the viscosities using Prandtl numbers as follows:

$$k_l = \frac{\mu_l C_p}{Pr_l}$$

$$k_t = \frac{\mu_t C_p}{Pr_t}$$

where Pr_l and Pr_t are the laminar and turbulent Prandtl numbers and C_p is the heat capacity at constant pressure.

The thermal conductivity may also be specified as a polynomial function as shown below.

$$k_l = \frac{c_1 T^{c_2} + c_3 T + c_4}{c_5 T + c_6}$$

The laminar conductivity cooresponding to Sutherland's viscosity for air and a Prandtl number of 0.71 is obtained by setting the above constants as follows:

$$\begin{aligned}c_1 &= 2.053 \times 10^{-3}\\ c_2 &= 1.5\\ c_3 &= 0\end{aligned}$$

$$
\begin{aligned}
c_4 &= 0 \\
c_5 &= 1 \\
c_6 &= 110
\end{aligned}
$$

The third method for calculating the thermal conductivity is to use the equilibrium air curve fits of Tannehill. The turbulent conductivity is always calculated using the turbulent Prandtl number and the turbulent viscosity.

2.4 Boundary Conditions

The boundary conditions are an important aspect of the problem formulation. At each boundary of the flow domain, boundary conditions must be applied based on the physical nature of the boundary. The DUMPSTER code has options for several types of boundary conditions as described in the following subsections.

2.4.1 Free-Slip Walls

When an inviscid analysis is being performed, or when a known stream surface is chosen as a boundary of the computed flow domain, a free-slip boundary condition is specified. The boundary condition is simply that there is no flow through the wall. This means that the component of velocity normal to the surface is zero and that the component of velocity tangent to the surface is allowed to change as required by the governing equation. The presence of a free slip wall also affects the pressure and temperature fields. If the wall is curved, a non-zero pressure gradient is needed normal to the wall to turn the flow in the direction that the wall is curving. This pressure gradient may be calculated from the normal momentum equation at the wall and used to set the wall pressure. However, in the analysis of dump combustors and nozzles, the viscous nature of the flow near actual walls is generally so significant that a no-slip boundary condition is necessary. The free-slip boundary condition then used only along stream surfaces, such as planes of symmetry, which are not curved. The gradients of pressure and temperature normal to free-slip walls are, therefore, assumed to be zero.

2.4.2 No-Slip Walls

At no-slip walls the velocity in the boundary cells is set to the negative of the velocity components in the corresponding internal cell such that the average velocity is zero at the wall. The pressure gradient is assumed to be zero. The temperurature in the boundary cell is set so that the specified wall temperature results when the internal cell and the boundary cell temperatures are averaged.

2.4.3 k–ϵ Turbulence Model Boundary Conditions

The k–ϵ turbulence equations require values for k and ϵ at the boundaries. On inflow, outflow, symmetry, and free-slip boundaries these quantities are set in the boundary cells assumming zero gradients.

At no-slip boundaries the wall function is used to calculate the wall shear stress and calculate the values for k and ϵ at the center of the internal cell next to the no-slip wall. Resolution of turbulent boundary layers requires a very finely spaced computational mesh near the walls. To partially alleviate the requirements for fine mesh the flow near the wall may be approximated analytically using wall functions in DUMPSTER. Wall functions[22] are used to obtain the wall shear stresses. As shown below a logarithmic function relates the wall shear stress to the velocity at the center of the internal cell next to the boundary.

In the application of wall functions it is assumed that a constant shear stress region exists in the boundary layer in which the compressible law-of-the-wall[27] given below is valid.

$$\frac{u_*}{u_\tau} = \frac{1}{\kappa} \ln\left(\frac{\eta \rho u_\tau}{\mu_l}\right) + 5.1$$

The shear velocity u_τ is defined as

$$u_\tau = \sqrt{\tau_w/\rho}$$

and u_* is defined as

$$u_* = \left(\frac{u_e}{\psi^{\frac{1}{2}}}\right) \arcsin\left\{\psi^{\frac{1}{2}}\left(\frac{u}{u_e}\right)\right\}$$

and

$$\psi = \frac{\bar{\gamma}-1}{2} M_e^2 \left(1 + M_e^2 \frac{\bar{\gamma}-1}{2}\right)$$

In the above equations κ is the von Karman constant with a value used here of 0.41, M_e and u_e are the Mach number and velocity tangent to the wall at the edge of the boundary layer. ρ_w and μ_l are the density and laminar viscosity of the fluid at the temperature of the wall. η is a length scale equal to the distance from the wall to the center of the internal cell next to the boundary where u is defined.

2.4.4 Supersonic Inflow

The supersonic inflow boundary condition requires specification of all variables. This boundary condition is only appropriate where the Mach number, based on the velocity normal to the boundary, is greater than one.

2.4.5 Subsonic Inflow

At a subsonic inflow boundary the total pressure (P_T), total temperature (T_T), and flow angles are specified as a function of the radius r. Along the inflow boundary the total velocity magnitude ($|\vec{V}|$) is extrapolated from within the flow field. Using the isentropic relation below the local Mach number (M) is calculated for the given velocity magnitude and total temperature.

$$M = \sqrt{\frac{2q}{\bar{\gamma}(\gamma - 1)\bar{C}_v T_T - (\gamma - 1)q}}$$

where q is defined as $q = \frac{1}{2}(u^2 + v^2 + w^2)$. Using the isentropic relations below, static pressure and temperature can then be calculated. Prescribed flow angles (α_i) are used to set the boundary velocity components given the extrapolated velocity magnitude.

$$\begin{aligned}
P &= P_T\left(1 + \frac{1}{2}(\gamma - 1)M^2\right)^{\frac{1}{1-\gamma}} \\
T &= T_T\left(1 + \frac{1}{2}(\gamma - 1)M^2\right)^{-1} \\
u &= \alpha_x|\vec{V}| \\
v &= \alpha_r|\vec{V}| \\
w &= \alpha_z|\vec{V}|
\end{aligned}$$

2.4.6 Supersonic Outflow

Along a supersonic outflow boundary, the flow variables in the boundary cells are completely determined by the upstream flow conditions. Strictly speaking, this boundary condition is only appropriate when the Mach number, based on the velocity normal to the boundary, is greater than one. Its use is acceptable for boundary layer flow when the boundary layer is thin and the flow outside of the boundary layer is supersonic.

2.4.7 Subsonic Outflow

The static pressure on the outflow boundary is specified at the upper nozzle/combustor wall. The actual outflow boundary static pressure must vary in the radial direction when axial swirl is present. According to the simplified radial momentum equation:

$$\frac{\partial P}{\partial r} = \frac{1}{2}\rho w^2$$

This equation is integrated to obtain the outflow boundary static pressure distribution. The values for ρk, $\rho \epsilon$, and ρf_i at the outflow boundary are obtained by zero-order extrapolation from the interior.

2.4.8 Interzone Boundary Condition

At the boundary between adjacent zones, the conservative variable data are simply transferred from the internal cells of one zone to the boundary cells of the adjacent zone.

2.5 Initial Conditions

In DUMPSTER there are three options for specifying initial flow conditions as described below. For each computational zone one of the following methods may be used.

- **Uniform Conditions** The flow field is set to uniform conditions as specified for this zone.

- **One-Dimensional Solution** The total pressure and total temperature at the inflow, the area variation, and the effective throat area are specified. From this, a one-dimensional isentopic solution for u, P, T and ρ is calculated. The cross-flow velocity component v is set so that the flow is parallel to the walls. The swirl component of velocity is uniform.

- **Custom** The user may provide FORTRAN77 coding that generates custom initial conditions.

3 DISCRETIZATION

The computational domain is divided into one or more computational zones. As shown in Figure 2 these zones are stacked in the r (radial) direction. Multiple zones allow easy mesh generation for a wide range of configurations of dump combustors and nozzles. The domain within each zone is discretized into a grid of finite-volume cells as shown in the figure. Within each zone the cells are indexed by two parameters: i and j.

An individual finite-volume cell, with indices i and j, is shown in Figure 3. Applying the Navier-Stokes equations in integral form to this volume gives

$$\frac{d}{dt}(U_{i,j}V_{i,j}) = -(D_i\,\vec{P}\cdot\vec{S} + D_j\,\vec{P}\cdot\vec{S}) + N_{i,j}$$

where $U_{i,j}$ is the mean value of U in cell i,j and $D_i\vec{P}\cdot\vec{S}$, for example, represents the difference of the fluxes through opposing i–faces of the cell.

The time derivative is approximated using backward-in-time differencing where n is the old time level and $n+1$ is the new time level.

$$\frac{V_{i,j}}{\delta t}\delta U_{i,j} = -(D_i\,\vec{P}\cdot\vec{S} + D_j\,\vec{P}\cdot\vec{S})^{n+1} + N_{i,j}^{n+1} \qquad (2)$$

where

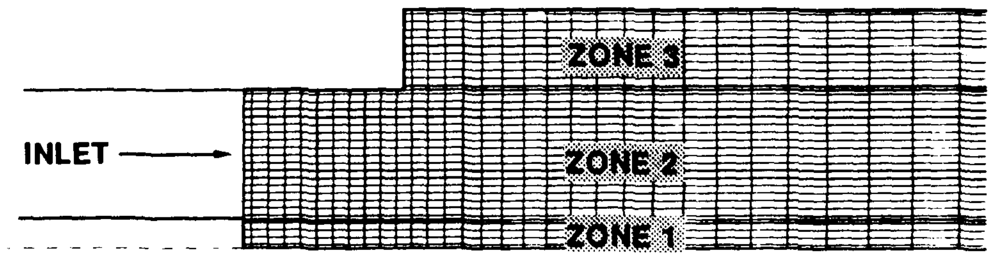

Figure 2: Multiple Computational Mesh Zones.

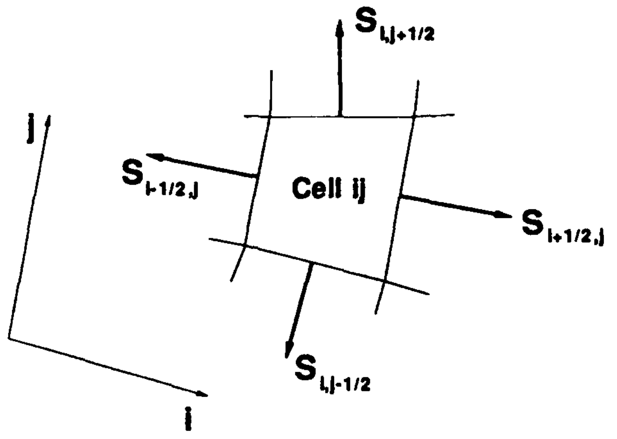

Figure 3: Finite-Volume Cell.

For the approximation of the flux through a surface the inviscid and diffusion terms of the flux vector are considered separately.

$$\vec{P}\cdot\vec{S} = \vec{P}\cdot\vec{S}^{\text{inv.}} + \vec{P}\cdot\vec{S}^{\text{diff.}}$$

These terms are then evaluated in a manner consistent with the predominant nature of the equations in the limit as $Re \rightarrow \infty$ (hyperbolic) and $Re \rightarrow 0$ (parabolic) (*i.e.*, upwind differencing for the inviscid terms and central differencing for the viscous stress and heat flux terms. Three flux functions are have been implemented into DUMPSTER:

- Steger and Warming's flux-vector-splitting[28],
- Harten and Yee's[29] second order TVD[2] scheme, and
- Roe's Approximate Reimann solver[30].

Each of the above flux functions is described in the following subsections. The diffusion terms are evaluated using standard central differences[31].

[2]Total Variation Diminishing

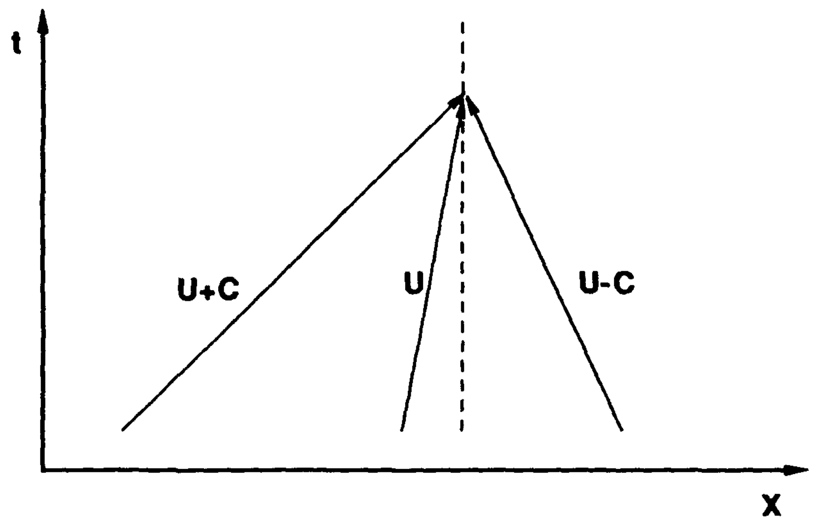

Figure 4: Mathematical Characteristics of One-Dimensional Subsonic Flow.

3.1 Steger and Warming Flux Function

Flux-vector splitting was developed by Steger and Warming[28] and Van Leer[33] as a way to upwind difference the Euler equations in regions of subsonic flow. If the flow normal to the surface of a cell is supersonic, the theory of characteristics tells us that the flow field at the surface depends only on the solution upstream of the surface. In this case, the flux at the surface may simply be evaluated using the solution in the cell upstream of the surface. The numerical flux $(\tilde{F})$ at the $i+1/2, j$-face of the i, j-cell is given as

$$\tilde{F}_{i+1/2,j} = \begin{cases} F\left(U_{i,j}\right) & \text{if } u' > c_s \\ F\left(U_{i+1,j}\right) & \text{if } u' < c_s \end{cases} \tag{3}$$

Where u' is the velocity component normal to the cell face. If the flow is subsonic, however, the flux through the $i+1/2, j$ surface depends on the solution both upstream and downstream of the surface. Then the simple upwinding applicable to supersonic flows, Equation 3, is no longer appropriate. The term upwind differencing must now be defined to mean differencing in a manner appropriate with the mathematical characteristics of the flow. As shown in Figure 4, there are characteristics running both to the right and left in subsonic flow and upwind differencing must, therefore, use backward differencing on some of the flux and forward differencing on the rest of the flux.

In flux-vector-splitting methods, the flux function is written as

$$F(U) = F^+(U) + F^-(U) \tag{4}$$

so that the flux-split Jacobians, $A^\pm = \frac{\partial F^\pm}{\partial U}$, have eigenvalues satisfying

$$\begin{aligned} \lambda_m\left(A^+\right) > 0 \qquad & \text{for all } m \\ \lambda_m\left(A^-\right) < 0 \qquad & \text{for all } m \end{aligned} \tag{5}$$

Flux splitting has divided a flux which cannot in general be upwinded into two fluxes which separately can be upwinded. With first-order upwinding the flux through the $i+1/2, j$ surface is

$$\tilde{F}_{i+1/2,j} = F^+(U_{i,j}) + F^-(U_{i+1,j})$$

Higher-order accurate methods are obtained by combined extrapolation and interpolation of the solution to the surface from nearby cells.

$$U^- = U_{i,j} + \frac{\phi}{4}\left[(1-\omega)(U_{i,j} - U_{i-1,j}) + (1+\omega)(U_{i+1,j} - U_{i,j})\right] \tag{6}$$

$$U^+ = U_{i+1,j} + \frac{\phi}{4}\left[(1-\omega)(U_{i+1,j} - U_{i+2,j}) + (1+\omega)(U_{i,j} - U_{i+1,j})\right] \tag{7}$$

Here ϕ varies the differencing between first-order, $\phi = 0$, and second order, $\phi = 1$. The second parameter, ω, varies the differencing between fully upwind, $\omega = -1$, and central differencing, $\omega = 1$. Using MUSCL-differencing[32] the second-order flux is then

$$\tilde{F}_{i+1/2,j} = F^+\left(U^-\right) + F^-\left(U^+\right)$$

Equations 4 and 5 do not uniquely define the split fluxes $F^\pm(U)$. In fact, there are many flux-splittings that have been proposed[28, 33, 34]. The two most common flux-splittings are those of Steger and Warming[28] and Van Leer[33]. Both splittings were originally developed for ideal gases and were later extended for use with real gases[25, 12].

Steger and Warming developed their splitting based on the homogeneity property, $F = AU$, of the flux vector with a thermally perfect gas. They wrote the flux Jacobian as

$$A = A^+ + A^-$$

where

$$A^\pm = X^{-1}\Lambda^\pm X$$

and $\Lambda^{\pm}$ is the diagonal matrix with elements

$$\lambda_m^{\pm} = \frac{1}{2}\left(\lambda_m \pm |\lambda_m|\right)$$

Here the matrices Λ, X, and X^{-1} are the diagonal matrix of eigenvalues, matrix of eigenvectors, and its inverse, for the flux Jacobian A. Therefore, A^+ and A^- are equal to the flux Jacobian matrix with the negative and positive eigenvalues set to zero, respectively. The split fluxes are $F^{\pm} = A^{\pm}U$. These matrix multiplies may be performed numerically for each face on each time step.

Significant savings in computer time is possible by eliminating the matrix multiplies as follows. Consider the contribution to the total flux from each of the following three unique eigenvalues separately.

$$\begin{aligned} \lambda_1 &= u' \\ \lambda_2 &= u' + c_s \\ \lambda_3 &= u' - c_s \end{aligned}$$

The λ_1 eigenvalue is repeated so that, for flows without species transport,

$$\Lambda = diag\left\{\lambda_1, \lambda_2, \lambda_1, \lambda_1, \lambda_3\right\}$$

The separate contributions from each eigenvalue are obtained by defining diagonal matrices with all but the desired eigenvalue set to zero.

$$\begin{aligned} \Lambda_1 &= diag\left\{\lambda_1, 0, \lambda_1, \lambda_1, 0\right\} \\ \Lambda_2 &= diag\left\{0, \lambda_2, 0, 0, 0\right\} \\ \Lambda_3 &= diag\left\{0, 0, 0, 0, \lambda_3\right\} \end{aligned}$$

The fluxes are then $F_m = X^{-1}\Lambda_m X U$. Performing the above operations for thermally perfect gas:

$$F_1 = \frac{\lambda_1 \rho(\gamma - 1)}{\gamma} \begin{bmatrix} 1 \\ u \\ v \\ w \\ \frac{1}{2}\left(u^2 + v^2 + w^2\right) + \rho e - \frac{c_s^2}{\gamma(\gamma-1)} \end{bmatrix} \qquad (8)$$

$$F_2 = \frac{\lambda_2 \rho}{2\gamma} \begin{bmatrix} 1 \\ u + c_s n_x \\ v + c_s n_r \\ w + c_s n_z \\ H + c_s u' \end{bmatrix} \tag{9}$$

$$F_3 = \frac{\lambda_3 \rho}{2\gamma} \begin{bmatrix} 1 \\ u - c_s n_x \\ v - c_s n_r \\ w - c_s n_z \\ H - c_s u' \end{bmatrix} \tag{10}$$

u' is defined as $u' = n_x u + n_r v + n_z w$. Vinokur[25] used an alternative approach to derive the Steger and Warming fluxes and found that the above formulas also apply to real gases, provided that γ is defined as

$$\gamma = \frac{\rho c_s^2}{P}$$

The split fluxes, $F^{\pm}$, are obtained from the F_m by splitting according to the sign of the eigenvalue. For $-c_s < u' < 0$ we therefore have

$$F^+ = F_2 \qquad \text{and} \qquad F^- = F_1 + F_3.$$

These fluxes are relatively inexpensive to calculate compared with the formulation requiring numerical matrix multiplies.

In general, some form of flux limiting is required to avoid oscillations at shock waves and where large second derivatives of the flow field variables exist. Limiting is implemented by locally varying the difference quotients in the MUSCL-differencing interpolation formulae given above in relation to the local gradients of the solution. If properly implemented a limiter will yield sharp monotonic shock waves. In this investigation, the limiter is based on the second difference of pressure. In particular

$$\phi_{i+1/2} = \text{MAX}\left(1 - C_{lim}\frac{\Delta\Delta P}{4\bar{P}},\ 0.0\right)$$

where $\Delta\Delta p$ is normal to the $i + 1/2, j$ cell face (in this case, the i-direction). The differencing parameter, ω, remains unchanged. This limiting reduces the form of differencing toward first-order upwind differencing in regions of strong pressure gradients (such as shock waves). The spacial order of accuracy remains second-order everywhere unless ϕ is actually zero.

3.2 Harten and Yee Flux Function

In the early 1980's Yee, Warming, and Harten[29], and others[35, 36, 37] presented upwind biased schemes for nonlinear scalar equations for which the *total variation* of the solution always diminished as the solution proceeds.

$$TV\left(U^{n+1}\right) \leq TV\left(U^{n}\right)$$

where

$$TV(U) = \sum_{i=-\infty}^{i=\infty} |U_{i+1} - U_i|$$

Schemes which are total variation diminishing (TVD) are guaranteed not to generate spurious oscillations and are therefore robust for applications involving shock waves. When the same scheme is applied to nonlinear systems it is no longer TVD, but it may still exhibit good accuracy and robustness for problems involving strong shock waves. A finite-volume version of Harten and Yee's scheme is included in the DUMPSTER code. The scheme is obtained by using the Harten-Yee flux function described in this section.

The Harten-Yee flux function is written as a central difference flux plus a dissipation term.

$$F_{i+1/2,j} = \frac{1}{2}\left[F_{i+1,j} + F_{i,j} + (X_A \Phi_A)_{i+1/2,j}\right] \tag{11}$$

The dissipation operators Φ_A are defined as follows:

$$(\Phi_A)_{i+1/2} = g_i + g_{i+1} - \Psi\left(\Lambda_{i+1/2} + \Gamma_{i+1/2}\right)\alpha_{i+1/2} \tag{12}$$

where

$$\alpha_{i+1/2} = (X_A)^{-1}_{i+1/2}(U_{i+1,j} - U_{i,j}) \tag{13}$$

and, in the standard, form $\Psi_{lm}(z) = |z_{lm}|$. A modified form of $\Psi(z)$ will be discussed latter in this section.

A key to the definition of the Harten-Yee flux is the eigensystem of the flux Jacobian matrix, $A = \frac{\partial F}{\partial U}$. The components of the eigensystem are the diagonal matrix of eigenvalues Λ, the matrix of right eigenvectors X, and its inverse $(X)^{-1}$. In the above equations, Φ is the dissipation term which will reduce the scheme to Roe's first-order upwind scheme in regions of steep gradients. The definition of the flux is completed by the relations

$$\begin{aligned} g_i &= S \max\left[0, \min\left(\sigma_{i+1/2}|\alpha_{i+1/2}|, S\,\sigma_{i-1/2}\alpha_{i-1/2}\right)\right] \\ S &= \operatorname{sign}(\alpha_{i+1/2}) \\ \sigma_{i+1/2} &= \frac{1}{2}\Psi\left(\Lambda_{i+1/2}\right) \end{aligned}$$

and

$$\Gamma_{i+1/2} = \begin{cases} \frac{g_{i+1}-g_i}{\alpha_{i+1/2}} & \text{if } \alpha_{i+1/2} \neq 0 \\ 0 & \text{if } \alpha_{i+1/2} = 0 \end{cases}$$

To avoid excessive roundoff error, $\Gamma_{i+1/2}$ in the above equation is actually set to zero any time $\alpha_{i+1/2}$ is near zero ($-10^{-22} < \alpha_{i+1/2} < 10^{-22}$).

This scheme is theoretically second-order accurate in space. In regions where the solution is smooth the dissipation term is turned off and the scheme reduces to a central difference scheme. Conversely, for regions with steep gradients, such as shock waves, it reduces to a first-order scheme to avoid aphysical oscillations in the solution. The first-order scheme to which it reduces is the flux-difference splitting of Roe[30].

Since the Harten-Yee flux function is based on Roe's flux-difference splitting, it shares certain abnormalities with Roe's scheme. In particular, Roe's scheme does not necessarily satisfy the entropy inequality and may, therefore, yield aphysical solutions such as expansion shock waves[35, 36, 37]. To overcome this weakness, Ψ is modified as follows:

$$\Psi_{mm}(z) = \frac{1}{2}\left(|z_{mm}| + \sqrt{z_{mm}^2 + \epsilon_m}\right)$$

Far away from zero each element of Ψ approaches the absolute value of the corresponding element of z, as it should. When z_{mm} approaches zero, however, Ψ_{mm} approaches the positive number ϵ_m.

The values of ϵ_m are calculated based on user defined constants and the type of characteristic equation which corresponds to λ_m. If the characteristic transports entropy

$$\epsilon_m = [c_5 + (|u| + c_s)]$$

If the characteristic transports acoustic waves

$$\epsilon_m = [c_6 + (|u| + c_s)]$$

Finally, if the characteristic transports momentum components tangent to the surface

$$\epsilon_m = [c_7 + (|u| + c_s)]$$

Different constants were provided for each type of characteristic. A flow problem with shocks may require a large value of c_6 to avoid an expansion shock, but a small value for c_7 to avoid excessive smearing of the boundary layer.

The CPU time for the matrix multiplies in Equations 11 and 13 would normally increase quadratically with the number of transport equations. In DUMPSTER however, the matrix multiplies are performed analytically whenever possible, and the CPU time increases nearly linearly with the number of added specie transport equations and k–ϵ equations.

To evaluate the variables at the cell surfaces, arithmetic averaging of the cell centered values is used. This kind of averaging has the advantage of computational simplicity and can be easily extended for problems in thermochemical nonequilibrium. Other kinds of averaging can also be used, but they are usually more complicated. Arithmetic averaging takes the form

$$\bar{\rho}_{i+1/2} = \frac{1}{2}(\rho_i + \rho_{i+1})$$

The same procedure is used to calculated $\bar{u}$, $\bar{v}$, $\bar{w}$, and $\bar{a}$ at the surface.

3.3 Roe's Flux Function

Roe's flux function is obtained when the g_i's in the discussion fo the Harten–Yee flux functions are set to zero. The method is first-order accurate.

4 SOLUTION PROCEDURES

4.1 Line Gauss-Seidel Method

The Line Gauss–Seidel (LGS) method is an unfactored method based on the work by Thomas and Walters[17], MacCormack[16], and Chakravarthy[18]. This type of algorithm has proven to be an efficient relaxation procedure for steady-state flow fields. The algorithm used in this investigation is an improvement on the algorithm presented by Peery and Imlay[38, 39]. A brief description of the basic algorithm is given below.

For conciseness we consider only the calculation of the flux at the $i+1/2, j$ cell face as shown in Figure 3. The left side of the face refers to the side that of the i, j cell, and the right side of the face refers to the side of the $i+1, j$ cell. For the approximation of the flux through a surface the inviscid and diffusion terms of the flux vector are considered separately.

$$\vec{P}\cdot\vec{S} = \vec{P}\cdot\vec{S}^{\text{ inv.}} + \vec{P}\cdot\vec{S}^{\text{ diff.}}$$

The total flux vector is a function of the solution in nearby cells

$$\vec{P}\cdot\vec{S}_{i+1/2} = f(U_{i-1,j}, U_{i,j}, U_{i+1,j}, U_{i+2,j})$$

The flux is evaluated at the new (unknown) time level and linearized as follows:

$$\begin{gathered}\vec{P}\cdot\vec{S}^{\,n+1}_{i+1/2} \approx \vec{P}\cdot\vec{S}^{\,n}_{i+1/2} + \\ A^{i-1}_{i+1/2,j}\,\delta U_{i-1,j} + A^{i}_{i+1/2,j}\,\delta U_{i,j} + A^{i+1}_{i+1/2,j}\,\delta U_{i+1,j} + A^{i+2}_{i+1/2,j}\,\delta U_{i+2,j}\end{gathered} \tag{14}$$

where

$$\begin{aligned}A^{i-1}_{i+1/2,j} &= \frac{\partial f^n_{i+1/2}}{\partial U_{i-1}}\\ A^{i}_{i+1/2,j} &= \frac{\partial f^n_{i+1/2}}{\partial U_{i}}\\ A^{i+1}_{i+1/2,j} &= \frac{\partial f^n_{i+1/2}}{\partial U_{i+1}}\\ A^{i+2}_{i+1/2,j} &= \frac{\partial f^n_{i+1/2}}{\partial U_{i+2}}\end{aligned}$$

The equation for the flux at the $S_{j+1/2}$ face is

$$\vec{P}\cdot\vec{S}^{\,n+1}_{j+1/2} \approx \vec{P}\cdot\vec{S}^{\,n}_{j+1/2} +$$

$$B^{j-1}_{i,j+1/2}\,\delta U_{i,j-1} + B^{j}_{i,j+1/2}\,\delta U_{i,j} + B^{j+1}_{i,j+1/2}\,\delta U_{i,j+1} + B^{j+2}_{i,j+1/2}\,\delta U_{i,j+2} \qquad (15)$$

where

$$\begin{aligned}
B^{j-1}_{i,j+1/2} &= \frac{\partial f^n_{j+1/2}}{\partial U_{j-1}} \\
B^{j}_{i,j+1/2} &= \frac{\partial f^n_{j+1/2}}{\partial U_{j}} \\
B^{j+1}_{i,j+1/2} &= \frac{\partial f^n_{j+1/2}}{\partial U_{j+1}} \\
B^{j+2}_{i,j+1/2} &= \frac{\partial f^n_{j+1/2}}{\partial U_{j+2}}
\end{aligned}$$

The above Jacobians (A and B) are calculated numerically on each cycle. Substituting Equations 14 and 15 into the discrete conservation equation (Equation 2), grouping terms and linearizing the source term ($N_{i,j}$) yields the following finite-volume equation.

$$\mathcal{V}_{i,j}\left[\frac{1}{\delta t} + \frac{\partial N_{i,j}}{\partial U_{i,j}}\right]\delta U_{i,j} +$$

$$D_i\left[A^{i-1}_{i+1/2,j}\,\delta U_{i-1,j} + A^{i}_{i+1/2,j}\,\delta U_{i,j} + A^{i+1}_{i+1/2,j}\,\delta U_{i+1,j} + A^{i+2}_{i+1/2,j}\,\delta U_{i+2,j}\right] +$$

$$D_j\left[B^{j-1}_{i,j+1/2}\,\delta U_{i,j-1} + B^{j}_{i,j+1/2}\,\delta U_{i,j} + B^{j+1}_{i,j+1/2}\,\delta U_{i,j+1} + B^{j+2}_{i,j+1/2}\,\delta U_{i,j+2}\right] = R_{i,j} \qquad (16)$$

where the residual $R_{i,j}$ is defined as

$$R_{i,j} = -\left(D_i\,\vec{P}\cdot\vec{S} + D_j\,\vec{P}\cdot\vec{S} + \mathcal{V}_{i,j}\,N_{i,j}\right)^n \qquad (17)$$

The residuals approach zero as the solution approaches steady state. On each time step the solution is advanced as follows:

1. Calculate residuals based upon the local variation of the solution for each cell using the explicit flux balance given in Equation 17.

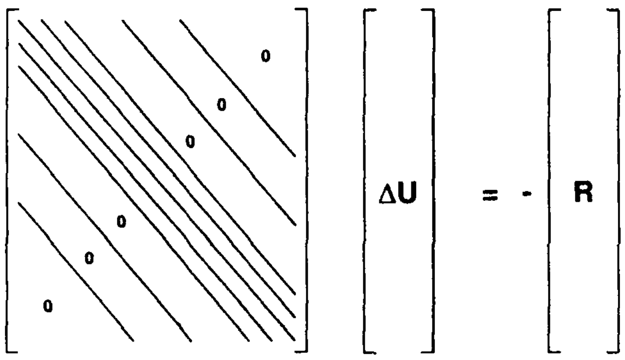

Figure 5: Linearized Algebraic System of Equations.

2. Calculate the change in the solution by solving the block-linear system of algebraic equations shown in Figure 5. The rows of this system are the block-linear algebraic relations resulting from the implicit flux balance, Equation 16, applied to each cell. This step implicitly calculates the change in the solution according to global variations in the residual.

3. Update the solution for each cell using

$$U_{i,j}^{n+1} = U_{i,j}^{n} + \xi_{i,j}\, \delta U_{i,j}$$

where $\xi_{i,j}$ is a local relaxation factor.

The first and last steps of the above algorithm are easy to implement. The second step is difficult, since it involves the solution of block banded system of linear algebraic equations, as shown in Figure 5. The coefficient matrix for this system is very large, 144,000x144,000 for a 60x60 mesh, but is sparse and well structured. Because the system is so large it is impractical to solve it directly and most investigators resort to approximately factoring this matrix into simpler matrices such as block tridiagonals. Unfortunately the error associated with the approximate factorization can severely restrict the allowable time step, and hence convergence rate. In this investigation this system was solved iteratively using a modified alternating direction line Gauss-Seidel relaxation method.

When the linear system of equations is solved accurately and the time step is taken very large, this procedure approximates a Newton method for the system of nonlinear algebraic equations resulting from differencing of the time independent Navier-Stokes equations. As with any Newton method, this procedure will only converge if the chosen initial conditions are sufficiently close to the final solution. Other similar methods[17, 16] have avoided these convergence problems by starting with small, nearly explicit, time steps and increasing the time step as the solution progresses. Unfortunately, it is impossible to choose a priori a time step sequence which will yield both a stable solution and an optimal convergence rate. Generally, when a new problem is attempted, the time step sequence is such that the solution diverges or it takes many times the optimal number of cycles to converge. As a result, many attempted runs are often required to obtain one solution.

These problems were overcome by adapting a technique for global convergence used in solving other systems of nonlinear equations. A controlling subroutine automatically adjusts the relaxation factor ($\xi_{i,j}$), the time step size, the number of explicit steps between each implicit step, the relaxation factor for the Gauss-Seidel iteration of the linear system The technique is described in more detail by Peery and Imlay[38].

The implicit time step is still set small initially, but is increased exponentially to a very large value ($CFL \approx 10^6$). The overhead for this procedure depends on the problem, but is certainly acceptable considering that it is only used if it is needed. With this technique the reliability of the Navier-Stokes solver has increased dramatically and converged blunt body solutions are routinely obtained in 10 to 100 implicit steps.

4.2 Lower-Upper Symmetric Gauss-Seidel Method (LU-SGS)

The LGS method explained above proved to very fast, but required a flux function that is numerically dissipative. The Lower-Upper Symmetric Gauss-Seidel (LU-SGS) method developed by Yoon and Jameson[20] was also studied in this work. The LU-SGS method is not as fast as the LGS method, but the LU-SGS method is more robust and reliable, especially when the flux functions have low numerical dissipation. In this work having low numerical dissipation is important for accurately simulating the recirculation regions of the dump combustor.

The TVD flux function of Yee and Harten was adopted for use with the LU-SGS method, since it exhibits low numerical dissipation. The numerical dissipation inherent with flux functions can overpower the natural action of viscosity in shear layers unless very fine meshes are used. This is especially true with the Steger and Warming flux function. The Harten-Yee TVD flux function, on the other hand, is less dissipative and, therefore, requires fewer mesh points for a given level of accuracy than either of the previous flux functions. The LGS method would not converge with the Harten-Yee TVD flux function.

The LU-SGS implicit algorithm requires less memory than the LGS algorithm. Without species transport or the k–ϵ turbulence model equations, the LGS algorithm required storage for 187 variables at each mesh cell. Under the same conditions the LU-SGS algorithm requires storage of 27 variables per mesh cell. The difference is even more dramatic when the k–ϵ turbulence model and specie transport equations are considered. With the old flux functions, the added speed of the LGS algorithm justified its inefficient memory use, but with the adoption of the TVD fluxes, that justification is gone. Another advantage of the LU-SGS is that it can easily be extended to chemically reacting flows in which the the addition of chemical species increases the computational work proportionally.

The flux vector is a function of the solution in nearby cells. Consider only the $i+1/2, j$ surface of the i, j cell as shown in Figure 3.

$$\vec{P}\cdot\vec{S}_{i+1/2} = f(U_{i-1,j}, U_{i,j}, U_{i+1,j}, U_{i+2,j}) \tag{18}$$

The flux is evaluated at the new (unknown) time level and linearized using Yoon's approximate Jacobians.

$$(\vec{P}\cdot\vec{S})^{n+1}_{i+1/2} \approx (\vec{P}\cdot\vec{S})^{n}_{i+1/2} + A^{+}_{i+1/2,j}\,\delta U_{i,j} + A^{-}_{i+1/2,j}\,\delta U_{i+1,j} \tag{19}$$

where

$$\begin{aligned} A^{+}_{i+1/2,j} &= \frac{1}{2}(A_{i,j} + r(A_{i,j})) \\ A^{-}_{i+1/2,j} &= \frac{1}{2}(A_{i+1,j} - r(A_{i+1,j})) \\ A_{i,j} &= \frac{\partial \vec{P}_{i,j}}{\partial U_{i,j}} \end{aligned}$$

and r_A is the spectral radius (maximum eigenvalue) of A. Substituting Equation 19 and the corresponding flux for the j-direction faces into the discrete conservation equation, Equation 2, yields the implicit finite-volume equation. This equation may be written as three steps:

1. Calculate the residuals for each cell using an explicit flux balance.

$$R_{i,j} = -(D_i\,\vec{P}\cdot\vec{S} + D_j\,\vec{P}\cdot\vec{S})^n + N^n_{i,j}$$

The residuals will approach zero as the solution approaches steady state.

2. Calculate the change in the solution by solving the block-linear system of algebraic equations. The rows of this system are the block-linear algebraic relations

resulting from the implicit flux balance applied to each cell.

$$\frac{\mathcal{V}_{i,j}}{\delta t}\delta U_{i,j} + \frac{\partial N_{i,j}}{\partial U_{i,j}}\delta U_{i,j} +$$

$$D_i\left[A^{+}_{i+1/2,j}\,\delta U_{i,j} + A^{-}_{i+1/2,j}\,\delta U_{i+1,j}\right] + D_j\left[B^{+}_{i,j+1/2}\,\delta U_{i,j} + B^{-}_{i,j+1/2}\,\delta U_{i,j+1}\right] = R_{i,j}$$

Because of the choice of Jacobians, this resulting block matrix multiplying $\delta U_{i,j}$ is approximately diagonal when the source Jacobian, $\frac{\partial N_{i,j}}{\partial U_{i,j}}$, is zero.

3. Update the solution using

$$U^{n+1}_{i,j} = U^{n}_{i,j} + \xi_r \delta U_{i,j}$$

where ξ_r is a relaxation factor. DUMPSTER starts with a users specified ξ_r near one and updates the solution. If the solution at time level $n+1$ results in negative pressures or temperatures the relaxation factor is halfed and the solution at $n+1$ is recalculated. This process is continued until all pressures and temperatures are non-zero or ξ_r reaches a specified minimum.

The above algorithm consists of three steps which must be done on each time step. The first step is an explicit step which calculates the residual from the local variation of the solution. The second step implicitly calculates the change in the solution according to global variations in the residual. The last step updates the solution.

The second step is the most difficult because it requires the solution of a large sparse system of linear equations. The LU-SGS algorithm approximately solves this system using two sweeps of a point Gauss-Seidel relaxation. Assume the iteration sweeps first in the direction of increasing i, j indices and then in the direction of decreasing i, j indices. The increasing i, j iteration is performed by applying the equation

$$\left[\left(\frac{\mathcal{V}_{i,j}}{\delta t} + \beta r_A + \beta r_B\right) I + \left(\frac{\partial N}{\partial U}\right)_{i,j}\right]\delta U^{*}_{i,j} - A^{+}_{i-1/2,j}\delta U^{*}_{i-1,j} - B^{+}_{i,j-1/2}\delta U^{*}_{i,j-1} = R_{i,j} \tag{20}$$

at each internal cell. The decreasing i, j iteration is performed by applying the equation

$$\left[\left(\frac{\mathcal{V}_{i,j}}{\delta t} + \beta r_A + \beta r_B\right) I + \left(\frac{\partial N}{\partial U}\right)_{i,j}\right]\delta U_{i,j} + A^{-}_{i+1/2,j}\delta U_{i+1,j} + B^{-}_{i,j+1/2}\delta U_{i,j+1} -$$
$$A^{+}_{i-1/2,j}\delta U^{*}_{i-1,j} B^{+}_{i,j-1/2}\delta U^{*}_{i,j-1} = R_{i,j} \tag{21}$$

This scheme may be written as an approximate lower-upper (LU) factorization by subtracting Equation 20 from Equation 21 and rearranging to remove $\delta U^{*}_{i,j}$ from the left hand side.

$$\left[\frac{\mathcal{V}_{i,j}}{\delta t}\left(1+\beta r_A+\beta r_B\right)I+\left(\frac{\partial N}{\partial U}\right)_{i,j}\right]\delta U_{i,j}+A^-_{i+1/2,j}\delta U_{i+1,j}+B^-_{i,j+1/2}\delta U_{i,j+1}$$
$$=\left[\frac{\mathcal{V}_{i,j}}{\delta t}\left(1+\beta r_A+\beta r_B\right)I+\left(\frac{\partial N}{\partial U}\right)_{i,j}\right]\delta U^*_{i,j} \quad (22)$$

β is a coefficient used to artificially increase the spectral radius and thereby increase the numerical stability. The coefficient is usually set to 1.0. Increasing it above 1.0 will normally reduce the convergence rate. The equation above replaces Equation 21. It yields the same results as Equation 21, but requires fewer floating point operations and less memory.

The source Jacobian, $\frac{\partial N}{\partial U}$, is generally not diagonal. However, for this investigation it was assumed to be zero with no adverse consequences. With this assumption the LU-SGS algorithm requires no block matrix inversions and is, therefore, very inexpenxive per time step compared to the LGS algorithm. This advantage is counter balanced by the comparatively slow convergence rate of the LU-SGS algorithm which may take thousands of time steps as compared to hundreds of steps for the LGS algorithm.

5 COMPUTER CODE

DUMPSTER is written in the FORTRAN77 language and may be compiled on almost any computer. It runs in batch mode, reading and writing to several input/output files. It consists of one main program module, 57 major subroutines, 5 include files (same as UPDATE common blocks), numerous minor subroutines. The sections below contain the descriptions of the following items:

- the structure of the computer code,
- the major subroutines,
- the internal field variable arrays and include file variables,
- the input/output files, and
- the input variables.

5.1 Descriptions of Subroutines

Figure 6 shows the hierarchy of the subroutines that make up DUMPSTER. In Figure 7 a flowchart shows the overall calling sequence of the subroutines. A brief description of each subroutine is given below. The intent of this section is to help the user that is interested in modifying DUMPSTER to become acquainted with the basic function of each routine. More information about the routine can be found in the FORTRAN77 listing.

`ARRAYS`	Pointer variables and array sizes are calculated for the field variable arrays (`X, U, Q, R, DU, DT,` and `TG`). This subroutine also calculates the I and J dimensions of each zone and checks that the total number of computational cells does not exceed the maximum allowed (`MAXCELLS`).
`BC`	This subroutine calls subroutine `BCZ` for each zone.
`BCBWD`	This subroutine transfers the residual variable (R) from the interior cells at the bottom of a zone to the boundary cells of the zone below along interzone boundaries for the backward sweep of the LU-SGS solution.
`BCC`	This subroutine calls subroutine `BCCZ` for each zone to set interzone boundary conditions.

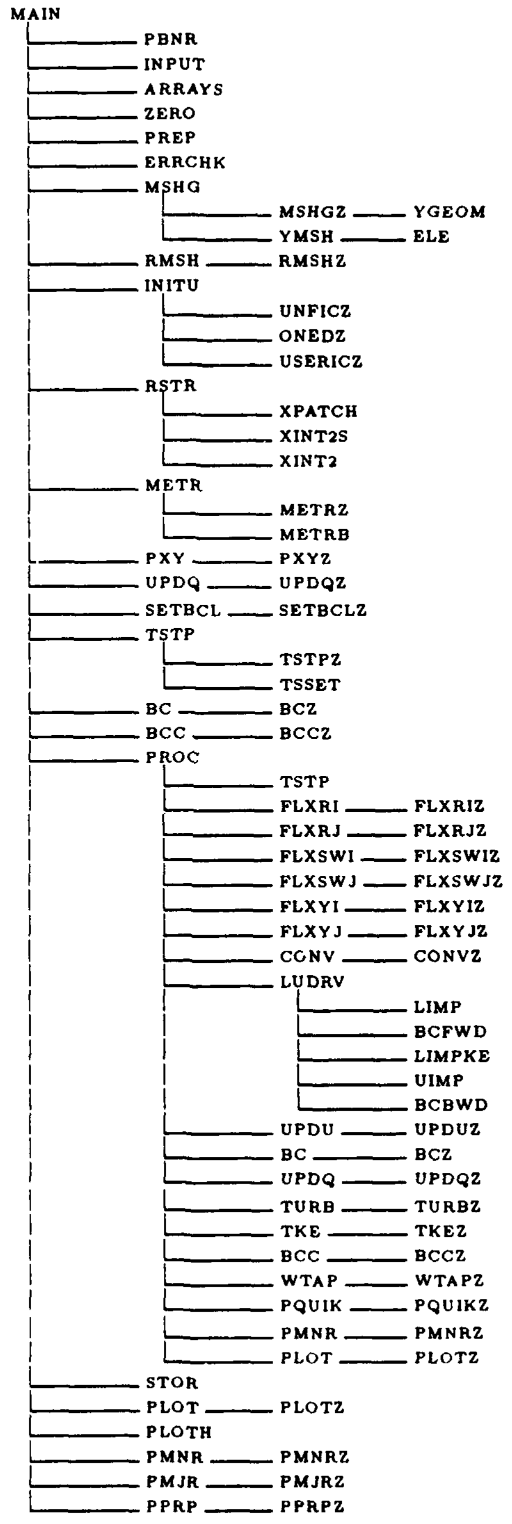

Figure 6: Hierarchy of DUMPSTER Subroutines.

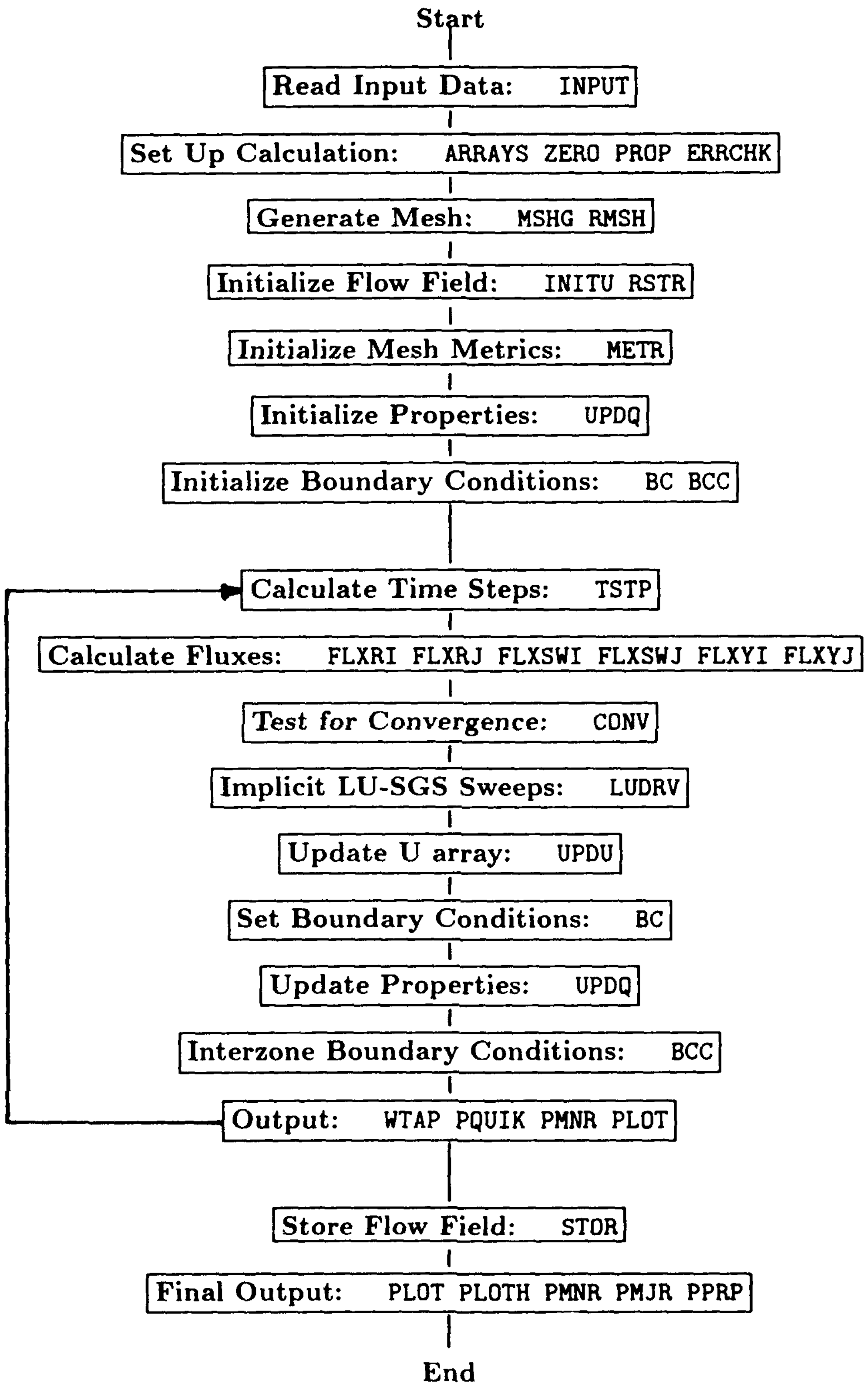
Start
Read Input Data: INPUT
Set Up Calculation: ARRAYS ZERO PROP ERRCHK
Generate Mesh: MSHG RMSH
Initialize Flow Field: INITU RSTR
Initialize Mesh Metrics: METR
Initialize Properties: UPDQ
Initialize Boundary Conditions: BC BCC
Calculate Time Steps: TSTP
Calculate Fluxes: FLXRI FLXRJ FLXSWI FLXSWJ FLXYI FLXYJ
Test for Convergence: CONV
Implicit LU-SGS Sweeps: LUDRV
Update U array: UPDU
Set Boundary Conditions: BC
Update Properties: UPDQ
Interzone Boundary Conditions: BCC
Output: WTAP PQUIK PMNR PLOT
Store Flow Field: STOR
Final Output: PLOT PLOTH PMNR PMJR PPRP
End

Figure 7: A Flowchart of the Major Subroutines of DUMPSTER.

BCCZ................	This subroutine transfers field variable data from within one zone to the overlaying boundary cells of a neighboring zone and also in the reverse direction from the interior cells of the neighboring zone to the boundary cells of the zone. This transfer takes place only on boundaries that are specified as interzone boundaries.
BCFWD..............	This subroutine transfers the residual variable (R) from the top interior cells of a zone to the bottom boundary cells of a neighboring zone above it along interzone boundaries for the forward sweep of the LU-SGS solution.
BCZ.................	Subroutine BC calls this routine for each zone for setting boundary conditions in the boundary cells. The real boundary conditions that affect the calculation of the fluxes are calculated in the FLXBC subroutine. These boundary conditions are for plotting purposes only and for transferring thermodynamic properties for real gas calculations.

CONV................ This subroutine calls subroutine CONVZ for each zone to calculate the level of convergence in each zone. The zonal convergence level (ε_n) is defined as the $\log_{10}$ of the average of the absolute value of the residuals of the mass density variable.

$$\varepsilon_n = \log_{10}\left(\sum_{ij}\left|\frac{\partial \rho_{ij}}{\partial t}\right|\right)$$

The average convergence level (CONVA) is simply the average of the zonal convergence levels for all of the zones.

$$\texttt{CONVA} = \frac{1}{N}\sum_{n=1,N}\varepsilon_n$$

N is the total number of zones. This subroutine also calculates the difference in the total mass flow rate at the left and right boundaries of each zone separately. This variable is only meaningful when a zone is not coupled to another zone and there is inflow at the left boundary and outflow at the right boundary as for simple nozzle flows.

CONVZ.............. This subroutine calculates the convergence level in a zone. See descussion for CONV for details.

ELE................. This subroutine calculates coefficients necessary to define the stretching of cells along a mesh line that starts with exponential stretching then has constant spacing of cells and then ends with exponential stretching. Typically this is a stretching that might be between two flat plates having a boundary layer on both plates. Position fine grid on each boundary that stretches out to a coarser uniform grid in the channel. The input to this routine is: distance coordinate at the bottom (Y(I,3)), distance coordinate at the top (Y(I,JDM)), cell spacing at the bottom (DELYB), cell spacing at the top (DELYT), and the number of cells in the bottom exponential, uniform middle, and top exponential regions.

ERRCHK	Numerous checks are made for the accuracy of the input data in this subroutine. If an error is detected then a message is printed to the OPRNT file and execution is stopped.
FLXRI	This subroutine calls FLXRIZ subroutine for each zone to calculate the fluxes at the I-faces of all of the cells and calculate the contribution of the I-fluxes to the explicit change of the field varibles (DU) using Roe's flux-difference flux function.
FLXRIZ	This subroutine calculates the fluxes at the I-faces of all of the cells within a zone using Roe's flux-differencing. Both the inviscid and viscous fluxes are calculated. The change in the conservative solution vector (U) is updated for the I-fluxes. The fluxes at solid boundaries are directly set in this routine.
FLXRJ	Same as FLXRI, but for the J-faces of the cells.
FLXRJZ	Same as FLXRIZ, but for the J-faces of the cells.
FLXSWI	Same as FLXRI, except that Steger–Warming flux-vector splitting is used to calculate fluxes.
FLXSWIZ	Same as FLXRIZ, except that Steger–Warming flux-vector splitting is used to calculate fluxes.
FLXSWJ	Same as FLXRJ, except that Steger–Warming flux-vector splitting is used to calculate fluxes.
FLXSWJZ	Same as FLXRJZ, except that Steger–Warming flux-vector splitting is used to calculate fluxes.
FLXYI	Same as FLXRI, except that Harten–Yee flux-vector splitting is used to calculate fluxes.
FLXYIZ	Same as FLXRIZ, except that Harten–Yee flux-vector splitting is used to calculate fluxes.
FLXYJ	Same as FLXRJ, except that Harten–Yee flux-vector splitting is used to calculate fluxes.
FLXYJZ	Same as FLXRJZ, except that Harten–Yee flux-vector splitting is used to calculate fluxes.
GSEC	This subroutine is used to call a system function to get the number of CPU seconds that have been used so far in the job. This subroutine must be customized for each computer system. The standard

distribution subroutine does not call a system routine and returns 0 seconds of CPU time used. The user inputs the maximum number of CPU seconds that the job is allowed to run. DUMPSTER periodically checks the amount of CPU time that remains. When the CPU time is almost used up DUMPSTER saves the current solution and terminates smoothly so that a restart run can easily be made.

INITU	This subroutine controls the initialization of the flowfield. For "NEW" runs for each zone this routine calls either subroutine UNFICZ, ONEDZ, or USERICZ to initialize the flow field.
INPUT	This subroutine controls the reading of the input variables from the input file called IDATA. An extended NAMELIST type input procedure is used to read the data. The NAMELIST routines are contained within the two files called XNUTIL and XNREAD. Most variables are first set to the default value. If a variable is in the input file, then the default value is overwritten. The value that is read from the input file is checked for valid range. Error messages are printed to OPRNT and execution stops if input errors are detected while reading input data.
LIMP	This subroutine performs the forward sweep of the LU-SGS iteration at each cell in a zone when the $k - \epsilon$ model is not activated.
LIMPKE	This subroutine performs the forward sweep of the LU-SGS iteration at each cell in a zone when the k-ϵ model is activated.
LUDRV	This is the controller subroutine that calls the LIMP, UIMP, LIMPKE, UIMPKE, BCFWD, and BCBWD subroutines that actually do the LU-SGS calculations within specific zones.
MAIN	This is the program module where execution begins. There are sections in this program module that are optionally commented out (C in column 1) that deal with setting up the memory for storing the field variables. For most computer systems with virtual memory it is satisfactory to hardwire the maximum number of cells (MXNCLS) in a parameter statement. On Cray computers memory may be allocated dynamically. DUMPSTER will compute and then request the necessary storage for the field variables automatically. So a user only has to compile the program once. The dimensions of the field variables are defined in this routine either explicitely or dynamically depending upon which lines of coding the user has commented out.
METR	This subroutine calls the subroutine METRZ and METRB to calculate the metric variables for each cell in each zone.
METRB	After the METRZ subroutine calculates the metrics for each zone, this subroutine transfers metrics from the interior of zones to the boundary cells of neighboring zones across interzone boundaries.

METRZ	This subroutine calculates the metrics for each cell within a zone. These metrics are the face area projections in the x and r directions for the top and right faces, the volume of the cell, and a metric used for calculating the axisymmetic source term.
MSHG	This subroutine calls MSHGZ to calculate a computational mesh in each zone.
MSHGZ	This subroutine calculates a computational mesh for each zone. The mesh generation procedure is described in Section 5.10.
ONEDZ	This subroutine calculates initial conditions for a zone using one-dimensional gas dynamics relations. These initial conditions are useful for setting initial conditions for nozzles and combustors with simple geometry.
PBNR	This utility routine prints a openning banner to the print out file OPRNT.
PHDR	This utility routine advances the print out to a new page and writes the TITLE character string at the top of the page.
PLOT	This subroutine writes the header information to the plot file (OPLTF) in the format for TECPLOT (Amtec Engineering's interactive plotting program.) PLOT then calls PLOTZ for each zone to write out the field variable data to the plot file. PLOT is always called at the end of a calculation. The user may specify in the input data to create plot files at intermediate steps during the calculation. These plots are used for plotting meshes, contours, and vector fields for visualizing the calculated solution data.
PLOTH	During the calculation DUMPSTER may (depending upon what the user specifies in the input data) write out the TAP data to a scratch file from subroutine WTAP. At the end of the calculation the PLOTH subroutine is called to read the TAP data from the scratch file, reorganize it, and then write it out to the file called OPLTH in TECPLOT format ready for plotting. TAP data consist of the field variables at a set of mesh cells at various computational steps. These data is useful for plotting convergence history of various field variables.
PLOTZ	This subroutine writes out the field variables data for a zone for creating field plots.
PMJR	This subroutine controls the printing of the major printout. It calls PMJRZ for each zone to print out field variable data.
PMJRZ	At user specified columns (I=constant) for a zone the coordinates, velocity components, pressure, temperature, density, turbulence quantities, total pressure, and total temperature are printed to file OPRNT for each cell.

PMNR................	This subroutine controls the printing of the minor print out. This is a selected amount of data that is printed several time during a calculation to monitor the solution progress. PMNRZ is called for each zone.
PMNRZ..............	This subroutine prints various field variables for a zone along th I-direction of the zone on the top, middle, and bottom boundaries. Field variables are also printed at user selected I-columns.
PPRP................	This subroutine controls the printing of thermodynamic and transport properties. PPRPZ is called for each zone.
PPRPZ..............	The thermodynamic and transport properties are printed to OPRNT for a zone at the I-columns specifed for minor prints in the input data.
PQUIK..............	This subroutine controls the printing of a quick and short amount of data to the OPRNT file and to standard out for monitoring the progress of the solution procedure. The computational step, CFL mulitplier, and the convergence level is printed. PQUIKZ is called for each zone to print out a sampling of the flow field data.
PQUIKZ............	This subroutine writes out a sampling of the data for a zone. This subroutine is easily modifiable so that the user may customize it for various calculations.
PREP................	After the input data has been read, and the array memory allocated, the program calles PREP subroutine to calculate and initialize various parameters and variables. These include setting metrics between zones along interzone boundaries, setting up temperature-dependent property coefficients, writing header information to plot files, and initializing the step counter and the CFL multiplier.
PROC................	The main iteration loop in which ISTEP in the index variable is contained in this subroutine. Each pass through the loop constitutes an iteration or computational step. The CFL multiplier and the relaxation coefficient are automatically adjusted in this routine to help maintain stability. If the time step falls below the explicit stability limit the code will deactivate the implicit solution procedure
PXY.................	The user has the option to print out the x and y (r) coordinates at the beginning of the calculation. This subroutine calls the subroutine PXYZ to print out the x and y coordinates for each zone.
PXYZ................	The x and y coordinates for a zone are printed out to file OPRNT in this subroutine.
RMSH................	The computational mesh may be read from an input file IMESH or generated internally. RMSH is the subroutine that controls the reading

of the external mesh. It calls subroutine RMSHZ to read the mesh for each zone.

RMSHZ	This subroutine reads the mesh coordinates for a zone from the input file IMESH.
RSTR................	At the end of a calculation DUMPSTER saves the state of the flow field in a restart file called ORSTR. This subroutine can read in this restart file (renamed as IRSTR) to continue with the calculation at a later time. The restart file contains the field variables X and U and some information about the calculation when it was saved that will allow DUMPSTER to continue the calculation as if it were not interupted.
SETBCL	DUMPSTER calls SETBCL at the beginning of the calculation to set up some arrays that store the boundary conditions at the left inflow boundary (I=1). For subsonic inflow boundary conditions this includes flow angle, total pressure, and total temperature distributions in the radial direction. In this subroutine these input arrays are interpolated into another set of arrays that have a one to one mapping to the boundary cells at the inflow boundary. When boundary conditions are applied at the inflow boundary, the flow angle and total conditions are readily available for each cell.
SETBCLZ	This subroutine calculates the boundary conditions arrays for the left inflow boundaries for a zone.
SOLVE	This subroutine is a utility subroutine that solves a system of N linear algebraic equations using a form of Gaussian elimination.
STOR................	At the end of a calculation this subroutine stores the flow field in a restart file called ORSTR.
TBLU................	This utility function performs linear and/or quadratic interpolation into a table of floating-point numbers.
TBLUI	This utility function is the same as TBLU except that it interpolates into a table of two variables—one integer and one floating- point.
TKE.................	This subroutine controls the calls to subroutine TKEZ to calculate the eddy viscosities when using the k-ϵ turbulence model.
TKEZ................	This subroutine calculates the eddy viscosity from the values of the turbulent kinetic energy (k) and turbulence dissipation rate (ϵ) in a zone when the k–ϵ turbulence model is used. The values of k and ϵ are also calculated near walls for coupling with the wall function.
TSSET	When the time step is uniform, this subroutine is used to set the time step in a zone. It is called from subroutine TSTP.

TSTP	This subroutine is used to control the calculation of the time step at each computational cell. The time step may be uniform or based on local conditions.
TSTPZ	This subroutine calculates the local time step in each cell of a zone.
TURB	This subroutine controls the calculation of the eddy viscosity in the zones when using the Baldwin-Lomax turbulence model. It calls TURBZ for each zone.
TURBZ	This subroutine calculates the eddy viscosity and eddy thermal conductivity in a zone. Only boundary layers on upper (J=JD) and lower (J=1) walls are allowed.
UIMP	This subroutine performs the backward sweep of the LU-SGS iteration at each cell in a zone when the k-ϵ model is not activated.
UNFICZ	This subroutine sets the field variables in a zone to some uniform initial conditions.
UPDQ	This subroutine controls the calls to subroutine **UPDQZ** to sets the thermodynamic and transport properties.
UPDQZ	This subroutine calculates and set the thermodynamic and transport properties in a zone.
UPDU	This subroutine controls the update of the conservative field variable array U at the end of a computational step. Subroutine **UPDUZ** is called to update the U-array for each zone.
UPDUZ	The conservative field variable array U is updated in this subroutine. DU is added to the U-array from the last time step. This subroutine also may limit the size of the change in U (*i.e.* DU) to a percentage of U.
USERICZ	The user may optionally add custom coding to DUMPSTER for initializing the flow field. This coding may be added to this subroutine.
WERROR	This utility subroutine writes error messages to the print file **OPRNT**.
WMSSG	This utility subroutine writes messages to the print file **OPRNT**.
WTAP	This subroutine controls the writting of tap data to the output file OTAP. It calls the subroutine **WTAPZ** to write out the data for each tap in each zone.
WTAPZ	When this subroutine is called it writes out the field variables at each tap for a zone. A tap is just a mesh cell location where the user may request (via the input data file IDATA) that field variable data be stored periodically during a calculation. After the calculation is completed, this data is organized for plotting with TECPLOT and written to the plot file **OPLTH**.

WWARNG	This subroutine writes out a warning message to the print file OPRNT.
XINT2	This subroutine is used to interpolated from a non-rectangular 2D mesh to a point. It returns the cell where the interpolation was done and the weighting coefficients for the bi-linear interpolation. These coefficients are used in the calling subroutine XPATCH to calculate the values of the field variables at the interpolated point.
XINT2S	This subroutine sets up the arrays for XINT2.
XPATCH	DUMPSTER may restart a calculation by initializing the flow field by interpolation from a flow field on a coarser mesh. This process is controlled in this subroutine. It calls XINT2S and XINT2 to perform the searching and interpolations.
YGEOM	The user may supply custom coding for generating complex geometries. This coding should be placed in this subroutine.
YMSH	This subroutine calculates a distribution of mesh points along a vertical line. It is called from MSHGZ.
(XN ROUTINES)	The extended NAMELIST routines are contained in the two files: XNREAD and XNUTIL. These routines should not be altered.
ZERO	This routine initializes the arrays to zero at the start of a calculation.

5.2 Description of Field Variable Arrays

This section gives a brief description of the field varibles that are used in the DUMP-STER coding. Field variables are those that are defined at each computational cell.

Internal Array Name	Math Symbol	Description
X	X	Metrics: coordinates, area projections, and volumes
U	U	Conservative fluid dynamic variables
DU	ΔU	Changes in conservative fluid dynamic variables.
R	R	Residual of fluid dynamics equations (similar to DU)
Q	Q	Thermodynamic and transport properties
DT	δt	Local time steps. (One element.)
TG	G_k	Turbulent dissipation function used in the k-ϵ model. (One dimensional)

Each of the above arrays contain many elements as shown in the following list. The last subscript in the arrays shown below are integer parameters defined in Section 5.4.

X(I,J,NX)	(x) Axial coordinate
X(I,J,NY)	(r or y) Radial or vertical coordinate
X(I,J,NAXT)	(A_x^T) Area projection of the face at the top $i,j+\frac{1}{2}$ of cell i,j in the x direction
X(I,J,NAYT)	(A_y^T) Area projection of the face at the top $(i,j+\frac{1}{2})$ of cell i,j in the y direction
X(I,J,NAXR)	(A_x^R) Area projection of the face at the right side $(i+\frac{1}{2},j)$ of cell i,j in the x direction
X(I,J,NAYR)	(A_y^R) Area projection of the face at the right side $(i+\frac{1}{2},j)$ of cell i,j in the y direction
X(I,J,NVOL)	($\mathcal{V}$) Volume of cell i,j
X(I,J,NAZ)	(A_z) Area projection of front and back of cell i,j in the y direction
U(I,J,NR)	(ρ) Mass density
U(I,J,NRU)	(ρu) x momentum

`U(I,J,NRV)`	(ρv) y momentum
`U(I,J,NRW)`	(ρw) Swirl momentum
`U(I,J,NE)`	(E) Total energy density, $E = \rho e + \frac{\rho}{2}(u^2 + v^2 + w^2)$
`U(I,J,NRK)`	(ρk) Turbulence energy
`U(I,J,NRE)`	$(\rho \epsilon)$ Turbulence dissipation rate
`U(I,J,NRF1)`	(ρf_1) Species number 1
`U(I,J,NRF2)`	(ρf_2) Species number 2
`U(I,J,NRF3)`	(ρf_3) Species number 3
`U(I,J,NRF4)`	(ρf_4) Species number 4
`U(I,J,NRF5)`	(ρf_5) Species number 5
`Q(I,J,NVIS)`	(μ) Effective viscosity
`Q(I,J,NCND)`	(k) Effective conductivity
`Q(I,J,NGAM)`	$(\bar{\gamma})$ Effective ratio of specific heat capacities for thermal equation of state. $\bar{\gamma} = P/(\rho e)$
`Q(I,J,NEXP)`	(γ) Defined as the ratio of the speed of sound squared times density to the pressure. $\gamma = \frac{\rho c^2}{P}$
`Q(I,J,NCVM)`	$(\bar{C}_v)$ Mean specific heat capacity at constant volume
`Q(I,J,NCHI)`	(χ) Thermodynamic function used in Roe's fluxes
`Q(I,J,NKPA)`	(κ) Thermodynamic function used in Roe's fluxes

Since the elements in the arrays `U`, `DU`, and `R` correspond one to one, only the `U` elements of the array are listed above. For an example: `U(I,J,NRW)` is the swirl momentum, `DU(I,J,NRW)` is the change in the swirl momentum, and `R(I,J,NRW)` is the residual of the swirl momentum equation (almost the same as `DU`). The third array index is specified indirectly using a variable such as `NRW` in the above example.

5.3 Description of the AGLBL Include File

The include file AGLBL is included in all subroutines and functions. It contains the following coding.

```
IMPLICIT REAL*8 (A-H,O-Z)
COMMON /DEBUG/ IDBUG(10)
```

This file makes all floating point variables double precision on a 32-bit computer. To compile DUMPSTER on a 64-bit computer like the Cray-Y/MP, the above include file must be changed to the following.

```
IMPLICIT REAL (A-H,O-Z)
COMMON /DEBUG/ IDBUG(10)
```

5.4 Description of the AGNRL Include File

AGNRL contains numerous parameters that either do not change during a calculation or change only rarely. The important parameters in this include file are listed below. The standard settings for version 1.0 of DUMPSTER for these variables are indicated. These settings may be changed.

MAXZON	Maximum number of zones (MAXZON=3)
MAXZONP	Maximum number of zones plus one (MAXZONP=4)
MAXID	Maximum number of cells in i–direction (MAXID=200)
MAXJD	Maximum number of cells in j–direction (MAXJD=50)
MAXNBC	Maximum number of points used to set inflow boundary conditions distribution (MAXNBC=100)
MAXIJD	Maximum of MAXID and MAXJD (MAXIJD=200)
MAXGEO	Maximum of number of geometry data points (MAXGEO=50)
MAXCOL	Maximum number of columns in minor printouts (MAXCOL=10)

MAXPRT	Maximum number of printouts (MAXPRT=100)
MAXPLT	Maximum number of plots (MAXPLT=100)
MAXTPS	Maximum number of field variable taps (MAXTPS=20)
MAXNEX	Maximum number of elements in X–array (MAXNEX=8)
MAXNEU	Maximum number of elements in U–array (MAXNEU=8)
MAXNEQ	Maximum number of elements in Q–array (MAXNEQ=8)
MAXNEDT	Maximum number of elements in DT–array (MAXNEDT=1)
MAXNETG	Maximum number of elements in TG–array (MAXNETG=1)
MAXNSPC	Maximum number of species in U–array (MAXNSPC=5)
MAXNKEU	Maximum number of elements used for the $k-\epsilon$ turbulence model (MAXNKEU=2)
MAXCLLS	Maximum number of cells as set in subroutine **MAIN**
IDMX	Dimension for X–array as set in subroutine **ARRAYS**
IDMU	Dimension for U–array as set in subroutine **ARRAYS**
IDMQ	Dimension for Q–array as set in subroutine **ARRAYS**
IDMR	Dimension for R–array as set in subroutine **ARRAYS**
IDMDU	Dimension for DU–array as set in subroutine **ARRAYS**
IDMDT	Dimension for DT–array as set in subroutine **ARRAYS**
IDMTG	Dimension for TG–array as set in subroutine **ARRAYS**
AXI	Coordinate switch: =1 for Axisymmetric, =0 for 2D planar
TWD	Coordinate switch: =0 for Axisymmetric, =1 for 2D planar
LOSTD	Standard out file (LOSTD=6)
LOSCRA	Scratch file (LOSCRA=9)
LIDATA	IDATA input file (LIDATA=10)
LIMESH	IMESH input file (LIMESH=11)
LIRSTR	IRSTR input file (LIRSTR=12)
LORSTR	ORSTR output file (LORSTR=20)
LOPRNT	OPRNT output file (LOPRNT=21)

LOPLTF	OPLTF output file (LOPLTF=22)
LOPLTH	OPLTH output file (LOPLTH=23)
LOTAPS	OTAPS output file (LOTAPS=30)
LOCONV	OCONV output file (LOCONV=31)

The following parameters in the AGNRL include file are used as subscripts in the field variable arrays. For example the x–component of momemtum in the U–array is U(I,J,NRU) whose mathematical symbol is ρu.

NX	x–coordinate (NX = 1)
NY	y– or r–coordinate (NY=2)
NAXT	Area Projection in x-direction at top face (NAXT=3)
NAYT	Area Projection in y-direction at top face (NAYT=4)
NAXR	Area projection in x-direction at right face (NAXR=5)
NAYR	Area projection in y-direction at right face (NAYR=6)
NVOL	Volume of cell (NVOL=7)
NAZ	Area projection of sides of cell in y-direction (NAZ=8)
NR	Mass density (NR=1)
NRU	x–momentum (NRU=2)
NRV	y–momentum (NRV=3)
NRW	Swirl momentum (NRW=4)
NE	Total energy densiy (NE=5)
NRK	Turbulence kinetic energy ρk (NRK=6)
NRE	Turbulence dissipation rate $\rho\epsilon$ (NRE=7)
NRF1	Mass fraction for species 1 (NRF1=8)
NRF2	Mass fraction for species 2 (NRF2=9)
NRF3	Mass fraction for species 3 (NRF3=10)
NRF4	Mass fraction for species 4 (NRF4=11)

NRF5	Mass fraction for species 5 (NRF5=12)
NVIS	Viscosity (NVIS=1)
NCND	Thermal conductivity (NCND=2)
NGAM	Ratio of specific heat capacities $= 1 + \frac{P}{\rho e}$ (NGAM=3)
NCVM	Mean specfic heat capacity at constant volume (NCVM=4)
NEXP	Ratio of ρc_s^2 to P (NEXP=5)
NCHI	Thermodynamic function used for Roe's fluxes (NCHI=6)
NKPA	Thermodynamic function used for Roe's fluxes (NKPA=7)

5.5 Description of the GCTRL Include File Variables

The variables in the GCTRL include file are described below. These variables are global in use (*i.e.* they are applicable to all zones.)

ICNTRL	Execution Method
NSTEPS	Number of steps to take for this run
NZONES	Total number of zones that are used
ICRD	Coordinate system switch: =0 for 2D planar, =1 for axisymmetric
NEU	Number of elements in the U-array. Normally NEU=5 for ρ,ρu, ρv, ρw, and E
CNVTL	Convergence tolerance. The number of orders of magnitude that the convergence level must be reduced before convergence is reached
SECJOB	Alloted number of seconds for the job to run
ITHRMO	Thermodynamic model selection integer
IVISC	Viscosity model selection integer
ITHINL	Thin-layer option for diffustion terms
ICOND	Thermal conductivity model selection integer
NSPC	Number of chemical species being transported
NTKE	Number of field variables used in turbulence model. This is normally 0 or 2 (for k-ϵ model)

GAMC................	Ratio of specific heat capacities
GASC................	Perfect gas constant (Normally = 287.Joules/Kg-Deg.K)
PRDL................	Laminar Prandtl number
PRDT................	Turbulent Prandtl number
CVCON(4)...........	Coefficients for the specific heat capacity
CVMCON(4)..........	Coefficients for the mean specific heat capacity
VISCON(6)..........	Coefficients for the viscosity
CNDCON(6)..........	Coefficients for the thermal conductivity
SPCCON(MAXNSPC)...	Diffusion coefficients for species transport
TKECON(5)..........	Coefficients for the k-ϵ turbulence model
ICDT................	Control integer for time step calculation
IFLXF	Flux function control interger
RFGS................	Gauss-Seidel relaxation coefficient
RFTS................	Time step relaxation coefficient
BETA................	Coefficient in the LU-SGS relaxation
CHGMAX	Fraction of change allowed when updating U-array
CFLBEG	CFLM used at the beginning of the calculation
CFLFAC	A factor that multiplies the CFLM to increase it from one step to the next
CFLMAX	The maximum value for the CFLM
FLXCFS	Coefficients used in the flux functions
IMSHG	Mesh generation method control integer
ISTEP	Current computational step counter
ICTOL	Control integer for
IMPL................	Impicit/Explicit switch: =0 for explicit =1 for implicit
CONVA	Current average convergence level
CFLM................	Current CFL multiplier
TIME................	Current time

SECCPU	Current number of seconds used in calculation
NPLT	Number of plots to be make
IPRT	Number of minor print outs to be make
NPRT	Steps at which minor print outs are to be made
IPLT	Steps at which plots are to be made
IFQPX	Interval of steps for quick print
ISKP	Skip interval for the i-index for major print outs
JSKP	Skip interval for the j-index for major print outs
IPRTQ	Properties print out control integer
IPRTM	Mesh coordinates and metrics print out control integer

5.6 Description of the ZCTRL Include File Variables

ZCTRL include file contains variables that define attributes of each zone. The dimension of the variables are also shown. MAXZON is the maximum number of zones allowed and MAXZONP is equal to MAXZON plus one.

IDZ(MAXZON)	Number of cells in i–direction (including four boundary cells)
JDZ(MAXZON)	Number of cells in j–direction (including four boudary cells)
IPNTX(MAXZONP)	Array pointer variable for X-array
IPNTU(MAXZONP)	Array pointer variable for U-array
IPNTQ(MAXZONP)	Array pointer variable for Q–array
IPNTR(MAXZONP)	Array pointer variable for R–array
IPNTDU(MAXZONP)	Array pointer variable for DU–array
IPNTDT(MAXZONP)	Array pointer variable for DT–array
IPNTTG(MAXZONP)	Array pointer variable for TG–array
ISHUMZ(3,MAXZON)	Difference in i index for neighboring cells that are on opposite sides of an interzone boundary Upper zone i–index minus lower zone i–index.
IIC(MAXZON)	Initial conditions control integer

PRESI(MAXZON).....	Initial condition pressure
TEMPI(MAXZON).....	Initial condition temperature
VVELI(3,MAXZON)...	Initial condition velcity components
TKI(MAXZON).......	Initial condition turbulence kinetic energy
TEI(MAXZON).......	Initial condition turbulence dissipation rate energy
FMI(MAXNSPC,MAXZON)	Initial condition species mass fraction
THMACH(MAXZON)....	Throat Mach number sometimes used for initial conditions
KBCB(3,MAXZON)....	Boundary condition type on bottom boundary segments
KBCT(3,MAXZON)....	Boundary condition type on top boundary segments
KBCL(MAXZON)......	Boundary condition type on left boundary
KBCR(MAXZON)......	Boundary condition type on right boundary
IBCB(4,MAXZON)....	Cell index array for defining start and end of boundary condition segments on the bottom boundary
IBCT(4,MAXZON)....	Cell index array for defining start and end of boundary condition segments on the top boundary
NTRPBC(MAXZON)....	Interpolation option for setting up the boundary conditions at the left inflow boundary
NBCVPT(MAXZON)....	The number of points that define the boundary condition array for the left boundary for the velocity angles, pressure, and temperature
NBCTKE(MAXZON)....	The number of points that define the boundary condition array for the left boudnary for the turbulence model
NBCSPC(MAXZON)....	The number of points that define the boundary condition array for the left boundary for the species mass fraction
PEX(MAXZON).......	The static pressure for outflow boundary conditions
TEX(MAXZON).......	Static temperature for outflow boundary conditions
BCCFS(9,MAXZON)...	Boundary condition coefficients
TWL(MAXZON).......	Temperature of left wall
TWR(MAXZON).......	Temperature of right wall
TWB(MAXZON).......	Temperature of bottom wall
TWT(MAXZON).......	Temperature of top wall

BCVPT(MAXNEU+1,MAXNBC,MAXZON)	Boundary conditions array for defining velocity components, pressure, and temperature distributions at the left boundary
BCTKE(MAXNKEU+1,MAXNBC,MAXZON)	Boundary conditions array for defining turbulence quantities at the left boundary
BCSPC(MAXNSPC+1,MAXNBC,MAXZON)	Boundary conditions array for defining species mass fractions at the left boundary
IDIM(MAXZON)	Number of internal cells in i-direction
JDIM(MAXZON)	Number of internal cells in j-direction
NSB(MAXZON)	Number of points defining the geometry of the bottom surface of zone
NTRPGB(MAXZON)	Interpolation method for bottom geometry
NST(MAXZON)	Number of points defining the geometry of the top surface of zone
NTRPGT(MAXZON)	Interpolation method for top geometry
ISW(MAXZON)	Control switch for generating mesh in the i-direction
NCX(4,MAXZON)	Number of cells in four segments of mesh in the i-direction
JSW(MAXZON)	Control switch for generating mesh in the j-direction
NJBK(MAXZON)	Number of blocks of cell in the j-direction
JBK(5,MAXZON)	Number of cells in block
NIJBKMAXZON)	Number of i-indices for defining mesh
IJBK(10,MAXZON)	An array of i-indices where mesh is defined
XYCONV(MAXZON)	Conversion factor that multiplies coordinates of externally generated mesh when it is read into DUMPSTER
XSBO(MAXZON)	x-coordinate of the origin of the geometry array for bottom surface
YSBO(MAXZON)	y-coordinate of the origin of the geometry array for bottom surface
XSTO(MAXZON)	x-coordinate of the origin of the geometry array for top surface
YSTO(MAXZON)	y-coordinate of the origin of the geometry array for top surface
XSB(MAXGEO,MAXZON)	x-coordinate of the geometry array for bottom surface
YSB(MAXGEO,MAXZON)	y-coordinate of the geometry array for bottom surface
XST(MAXGEO,MAXZON)	x-coordinate of the geometry array for top surface

YST(MAXGEO,MAXZON)	y-coordinate of the geometry array for top surface
XCTR(MAXZON)	x-coordinate of the mesh origin
DELX(MAXZON)	Cell spacing in the x-direction at the mesh origin
SLX(4,MAXZON)	A lenth of the mesh segement of a stretch factor for the mesh segment in the x-direction
DYBJBK(10,MAXZON)	Cell spacing in the y-direction at the bottom of the zone
DYTJBK(10,MAXZON)	Cell spacing in the y-direction at the top of the zone
EJBK(10,5,MAXZON)	Stretch factors for mesh generation in the y-direction
ITCHK	Check sum for number of taps
NTAPS(MAXZON)	Number of taps in zone
ITAP(MAXTPS,MAXZON)	i-indices of taps
JTAP(MAXTPS,MAXZON)	j-indices of taps
NCOLPRT(MAXZON)...	Number of columns in minor print
ICOLPRT(MAXCOL,MAXZON)	i-indices of columns in minor print
XMFR(MAXID,MAXZON)	Mass flow rate across an i-plane
CNVP(4,MAXZON)....	Convergence parameters

5.7 Input Files

DUMPSTER reads data from one or more of the files listed in the table below. When running DUMPSTER the files IRSTR and IMESH must be present even if they are not used.

Logical Unit Name	Description	Type	File Name
LIDATA	Input Variable File	ASCII	IDATA
LIRSTR	Input Restart Data File	binary	IRSTR
LIMESH	Input External Mesh File	ASCII	IMESH

5.8 Output Files

DUMPSTER writes data to one or more of the files listed in the table below.

Logical Unit Name	Description	Type	File Name
LOPRNT	Output Printer File	ASCII	OPRNT
LORSTR	Output Restart File	binary	ORSTR
LOPLTF	Output Plot File	ASCII	OPLTF
LOPLTH	Output Plot History File	ASCII	OPLTH
LOTAPS	Output Taps History File	ASCII	OTAPS
LOCONV	Output Convergence File	ASCII	OCONV
LOSTD	Standard Output	ASCII	(Unit 6)

The file OPRNT is suitable for printing on a line printer or viewing on a computer screen. ORSTR may be renamed IRSTR for restarting a calculation. Files OPLTH, OPLTF, OCONV, and OTAPS are in an ASCII format that is readable by the plotting program called TECPLOT[40].

The logical unit name LOSCRA is used for a scratch file called SCRA. This file is used for temporary storage of the tap data.

5.9 Dimensional Units

The dimensional units of the input and output variables, as well as the internal variables in DUMPSTER, are in MKS (Meter-Kilogram-Second). The units for common variables are tabulated below.

Physical Quantity	Units in MKS	Basic Units
Length	meters	m
Area		m^2
Volume		m^3
Time	seconds	s
Velocity		m/s
Mass	kilogram	Kg
Density		Kg/m^3
Temperature	deg. Kelvin	^oK
Force	Newtons (N)	$Kg \cdot m/s^2$
Pressure	Pascals (Pa)	N/m^2
Energy	Joules (J)	$N \cdot m$
Energy density		$J/Kg = m^2/s^2$
Heat transfer rate		$J/(m^2 \cdot s)$
Viscosity (absolute)		$Kg/(m \cdot s)$
Thermal conductivity		$J/(m \cdot s \cdot {}^oK)$

5.10 Input Variable Description

The variables that are specified in the input data file IDATA are defined below. These variables are input in several extended NAMELIST blocks. There are defaults for most input variables as shown in the descriptions below. It is, therefore, not necessary to specify every input variable in the NAMELIST blocks in the IDATA file.

5.10.1 CONTROL NAMELIST

TITLE .... A descriptive title consisting of up to 60 characters. The character string must be delimited with single quotes.

Internal Name = TITLE
Default Value = 'DUMPSTER: AMTEC DUMP COMBUSTOR CODE.'

EXECUTION.MODE.... The mode of execution of DUMPSTER may be set to one of the following:

='NEW' Start a new calculation from scratch. Generate a mesh and initialize the flow field.

='RESTART' Start a new calculation using the data from a restart file (IRSTR) to initialize the flow field. The mesh in the restart file may also be used by setting the input variable MESH.GENERATION.MODE to 'RESTART'. The flow field variables from the restart file will be interpolated on to the mesh that is generated for this calculation.

='CONTINUE' Continue a calculation saved on a restart file. The mesh and flow field variables from a restart file (will be used for initial conditions. The calculation will continue as if the previous calculation did not stop. The CFLM, for example, will begin at the value it had at the end of the previous calculation saved in the restart file. The mesh cell indices (I and J) must be the same as those in the restart file.

Internal Name = ICNTRL
Default Value = 'NEW'

NUMBER.OF.STEPS... The number of computational steps that the calculation will run. If you are restarting an old calculation with EXECUTION.MODE = CONTINUE then the step counter (ISTEP) will start at the step in the restart file, otherwise the step counter always starts at 0.

Internal Name = NSTEPS
Default Value = 0

CONVERGENCE.TOLERANCE Specify the number of orders of magnitude that the convergence level must be reduced to achieve convergence. The convergence level

(ε_n) of a zone is defined as the $\log_{10}$ of the average of the absolute value of the residuals of the mass density variable.

$$\varepsilon_n = \log_{10}\left(\sum_{ij}\left|\frac{\partial \rho_{ij}}{\partial t}\right|\right)$$

The average convergence level (**CONVA**) is simply the average of the zonal convergence levels for all of the zones.

$$\texttt{CONVA} = \frac{1}{N}\sum_{n=1,N}\varepsilon_n$$

where N is the total number of zones. When **CONVA** reduces by more than the **CONVERGENCE.TOLERANCE** then the calculation stops.

Internal Name = CNVTL
Default Value = -6.0

CPU.SECONDS.MAXIMUM The calculation will terminate safely before the number of CPU seconds specified with this input variable is reached. A restart file will be saved. This feature may not be implemented on the version of DUMPSTER on your computer system.

Internal Name = SECJOB
Default Value = 999999.0

NUMBER.OF.ZONES... The number of zones in the calculation.

Internal Name = NZONES
Default Value = 1

COORDINATE.SYSTEM Specify the type of coordinate system.

='AXISYMMETRIC'	Axisymmetric (X=axial and Y=radius)
='PLANAR'	2D planar

Internal Name = ICRD
Default Value = 'AXISYMMETRIC'

DEBUG.FLAGS .... Set these inputs to turn on/off debug and standard out printout statements.
IDBUG(1) = 1, Print convergence data to standard out.
IDBUG(1) = 2, Print convergence data to standard out for each zone.
IDBUG(2) = 1, Print pointers in subroutine **ARRAYS**.
IDBUG(3) = 1, Print U-array at beginning of subroutines **FLXRI** and **FLXRJ**.

Internal Name = (IDBUG(L),L=1,10)
Default Value = 0

5.10.2 PROPERTIES NAMELIST

THERMODYNAMIC.MODEL Selection for the thermodynamics model.

='PERFECT.GAS' Thermally and calorically perfect gas. You must also specify **GAMMA** and **GAS.CONSTANT** below.

='THERMALLY.PERFECT.GAS' Thermally perfect gas (calorically imperfect) temperature-dependent properties. Also specify: **GAMMA**, GAS.CONSTANT, and SPECIFIC.HEAT.COEFS

Internal Name = ITHRMO
Default Value = 'PERFECT.GAS'

GAMMA .... The ratio of specific heat capacities used in the perfect gas thermal equation of state: $P = (\bar{\gamma} - 1)\rho e$

Internal Name = GAMC
Default Value = 1.4

GAS.CONSTANT ...... Perfect gas constant as used in the relation for specific heat capacity $C_v = R/(\bar{\gamma} - 1)$.

Internal Name = GASC
Default Value = 287.0

SPECIFIC.HEAT.COEFS Coefficients for temperature-dependent specific heat capacities at constant volume for the *function shown below.*

$$C_v(T) = c_1 T^{c_2} + c_3 T + c_4$$

the units of the CVCON coefficients ($c_i, i = 1,4$) must be choosen to be consistent with the units of temperature in degrees-Kelvin and the units of C_v in the units of $m^2/(s^2 \cdot {}^oK)$.

Internal Name = (CVCON(L),L=1,4)
Default Value = 0.0, 1.0, 0.0, 717.5 (for air)

VISCOSITY.MODEL... Selection for the viscosity model.

='INVISCID' No viscosity or heat conduction.

='CONSTANT' Viscosity is uniform and constant. Also specify the constant viscosity as the first element in **LAMINAR.VISCOSITY.COEFS**

='LAMINAR' Temperature-dependent laminar viscosity. Also specify the viscosity coefficients in **LAMINAR.VISCOSITY.COEFS**

='TURBULENT' Use the Baldwin/Lomax algebraic model for eddy viscosity and temperature-dependent laminar viscosity. Also specify the laminar viscosity coefficients in **LAMINAR.VISCOSITY.COEFS**

='KE.TWO.EQUATION' k-ϵ two-equation model. Also specify the k-ϵ coefficients in 'KE.CONSTANTS' and the laminar coefficients in LAMINAR.VISCOSITY.COEFS

Internal Name = `IVISC`
Default Value = `'INVISCID'`

`THIN.LAYER.OPTION` The thin-layer option is an approximation to the diffusion terms. When this option is on, the cross derivatives in the diffusion terms are elimenated.

`=YES` Use only the thin-layer diffusion terms.

`=NO` Use full diffusion terms.

Internal Name = `ITHINL`
Default Value = `'NO'`

`LAMINAR.VISCOSITY.COEFS` Coefficients for temperature-dependent laminar viscosity (μ_l) are used in the following formula.

$$\mu_l(T) = \frac{c_1 T^{c_2} + c_3 T + c_4}{(c_5 T + c_6)}$$

The units of the coefficients ($c_i, i = 1,6$) must be choosen to be consistent with the units of temperature in degrees-Kelvin and the units of viscosity of $Kg/(m \cdot s)$.

Internal Name = `(VISCON(L),L=1,6)`
Default Value = `1.4519E-6, 1.5, 0.0, 0.0, 1.0, 110.0` (Sutherland's relation for air)

`KE.CONSTANTS......` The constants used in the k-ϵ turbulence model are specified in this array as follows.
TKE(1) = $C_{\epsilon 1}$, Used in the source term of the ϵ equation.
TKE(2) = $C_{\epsilon 2}$, Used in the source term of the ϵ equation.
TKE(3) = C_{μ}, Used in the definition of the eddy viscosity.
TKE(4) = σ_k, Used in the diffusion term of the k equation.
TKE(5) = σ_ϵ, Used in the diffusion term of the ϵ equation.

Internal Name = `(TKECON(L),L=1,5)`
Default Value = `1.44, 1.92, 0.09, 1.0, 1.3`

`CONDUCTIVITY.MODEL` Selection for the thermal conductivity model.

`='CONSTANT'` Enter constant as `CONDUCTIVITY.COEFS(1)`

`='PRANDTL.NUMBERS'` Use laminar and turbulent Prandtl numbers to compute conductivities from viscosities. Also specify `LAMINAR.PRANDTL.NUMBER` and `TURBULENT.PRANDTL.NUMBER`

`='TEMPERATURE.DEPENDENT'` Use turbulent Prandtl number in calculating turbulent conductivity, and conductivity coefficients to calculate the laminar conductivity. Also specify `CONDUCTIVITY.COEFS` and `TURBULENT.PRANDTL.NUMBER`.

Internal Name = `ICOND`
Default Value = `'PRANDTL.NUMBERS'`

`LAMINAR.PRANDTL.NUMBER` Laminar Prandtl number. May be used to compute the thermal conductivity from the viscosity when the thermal conductivity model (`CONDUCTIVITYY.MODEL` is set equal to `'PRANDTL.NUMBERS'`.)

Internal Name = `PRDL`
Default Value = `0.71`

`TURBULENT.PRANDTL.NUMBER` Turbulent Prandtl number. May be used to compute the turbulent thermal conductivity from the viscosity.

Internal Name = `PRDT`
Default Value = `0.9`

`CONDUCTIVITY.COEFS` The coefficients for temperature-dependent thermal conductivity are used in the following formula.

$$k(T) = \frac{(c_1 T^{c_2} + c_3 T + c_4)}{(c_5 T + c_6)}$$

For Sutherland's relation for air as perfect gas, use the default values of the coefficients and also set `GASC=287.0`, `GAMMA=1.4`, and `PRDL=0.71`. The coefficients must be choosen to be consistent with the units of temperature in degrees-Kelvin and the units of conductivity of $Kg \cdot m/^oK$.

Internal Name = `(CNDCON(L),L=1,6)`
Default Value = `2.0541E-3, 1.5, 0.0, 0.0, 1.0, 110.0`

`NUMBER.OF.SPECIES` The number of chemical species calculated. This number must not be greater than `MAXNSPC` (see Section 5.4). Trace chemical species have no effect upon the other species or the fluid dynamics of the flow.

Internal Name = `NSPC`
Default Value = `0`

`SPECIES.COEFS.....` Schmidt numbers used to calculated the diffusion coefficients for the chemical species from the viscosity coefficient of the flow. The diffusion coefficient, Γ_{f_i}, is calculated by dividing the viscosity Γ_M by the Schmidt number, σ_{f_i}, of the i^{th}-species.

$$\Gamma_{f_i} = \Gamma_M / \sigma_{f_i}$$

Internal Name = `(SPCCON(L),L=1,NSPC)`
Default Value = `0`

5.10.3 MESH NAMELIST

MESH.GENERATION.MODE The mesh generation mode.

='INTERNAL' Use the internal mesh generator. If EXECUTION.MODE = 'NEW', an internal mesh is generated. If EXECUTION.MODE = 'RESTART' or 'CONTINUE', the mesh is read from the input restart file IRSTR.

='EXTERNAL' Read an external mesh from the file IMESH. DUMPSTER checks the number of cells in the *i* and *j* directions of the external mesh (not counting the boundary cells). They should be the same as the number of cells read from the IDATA file as NUMBER.I.CELLS and NUMBER.J.CELLS.

Internal Name = IMSHG
Default Value = 'INTERNAL'

5.10.4 OUTPUT NAMELIST

QUICK.PRINT.FREQUENCY The step interval for printing the 'quick print out'. This is a short listing of a the times step, CFLM, and a listing of a few variables along the zone walls. This prints to the screen (Unix standard out).

Internal Name = IFQPX
Default Value = 99999

NUMBER.OF.PRINTS . Number of minor prints to make at the computational steps defined in STEPS.TO.PRINT. A minor print lists various field variables along each zone boundary and some along the middle row of cells (J=JD/2). Also the field variable in zero or more columns of cells (i=constant), may be printed out depending upon the input values of set in the ZONE.OUTPUT NAMELIST for each zone.

Internal Name = NPRT
Default Value = 0

STEPS.TO.PRINT.... This is an array of computational steps at which minor printouts are done. There are NPRT values.

Internal Name = IPRT
Default Value = 0

NUMBER.OF.PLOTS... Number of plots to make at the computational steps defined in STEPS.TO.PLOT.

Internal Name = NPLT
Default Value = 0

STEPS.TO.PLOT..... This is an array of computational steps at which a plot is made. These plots are in TECPLOT format and appended together in the output file OPLTF. The last plot is done automatically.

Internal Name = IPLT
Default Value = 0

MAJOR.PRINT.SKIP.I At the end of a calculation all of the field variables in all of the cells is printed to OPRNT. Since the amount of data in a major print out can be very large, used MAJOR.PRINT.SKIP.I and MAJOR.PRINT.SKIP.J to skip points that are printed out.

Internal Name = ISKP
Default Value = 999

MAJOR.PRINT.SKIP.J Same as MAJOR.PRINT.SKIP.I except this applies to the j-index in the major print out.

Internal Name = JSKP
Default Value = 999

`PRINT.PROPERTIES` .	Selection for printing properties in the minor printouts.
	='NO' Do not print properties.
	='YES' Print properties.
	Internal Name = IPRTQ
	Default Value = 'NO'
`PRINT.METRICS`.....	Selection for printing mesh coordinates and metrics.
	='NO' Do not print coordinates and metrics.
	='XY' Print x and y coordinates only.
	='VOLUMES' Print volume, and **NAZ** metrics only.
	='ALL' Print x, y, volume, and all the mesh metics.
	Internal Name = IPRTM
	Default Value = 'NO'

5.10.5 NUMERICS NAMELIST

TS.RELAXATION.FACTOR	Relaxation coefficient for the overal time step. This is a factor between 0 and 1 that multiplies DU before it is added to U. This should not be used for explicit calculations. It is useful for damping implicit calculations. **Internal Name** = RFTS **Default Value** = 1.0
LU.BETA............	This is a coefficient used in the LU-SGS relaxation. It is the β factor in Eq. 22. It should normally be set to 1.0. If the LU-SGS has stability problems increase LU.BETA up to a maximum of about 5.0. **Internal Name** = BETA **Default Value** = 1.0
TIME.STEP.MODE....	This is the method used to calculate the base time steps. For each cell the base time step is multiplied by the CFLM factor to get the actual time step for that cell. The base time step is calculated using one of the following options. ='LOCAL' The local explicit time-step. ='UNIFORM' A uniform minimum explicit time step. ='LOCAL.J.COLUMN' A time step is calculated for each column of cells (same value of I). The time step is explicit in the i-direction. It may be larger than the explicit limit in the j direction. ='UNIFORM.NEGLECT.J' This is almost the same as 'LOCAL.J.COLUMN' except that the time step is uniform throughout the zone. ='UNIFORM.NEGLECT.I' This is the same as 'UNIFORM.NEGLECT.J' except explicit in the j-direction. **Internal Name** = ICDT **Default Value** = 'LOCAL'
DU.CHANGE.MAXIMUM	The relative changes in the mass density ($\Delta\rho$) and the total internal energy density (ΔE) in one time step can be limited to be no larger than DU.CHANGE.MAXIMUM. If either exceeds this allowable maximum relative change, then the factor necessary to reduce them to the limit is also applied to the other elements of the U array. Normally this option is not used. Setting it to 1.0 turns it off. Use only in cases of severe numerical instabilities. **Internal Name** = CHGMAX **Default Value** = 1.0
CFLM.BEGIN........	This is the beginning CFL multiplier (CFLM) for a run. CFLM multiplies the base time step at each cell (see the input variable

TIME.STEP.MODE) to obtain the actual time step to use in the calculation. On restart runs the CFLM.BEGIN is usually ignored and the initial CFLM is set equal to the value from the restart file. If, however, a minus sign is placed before the value assigned to (CFLM.BEGIN), then the CFLM is set equal to absolute value of CFLM.BEGIN.

Internal Name = CFLBEG
Default Value = 1.0

CFLM.FACTOR .... This factor multiplies the CFL multiplier (CFLM) on each computational step during the calculation causing CFLM to geometrically increase.

$$\text{CFLM}^n = \text{CFLM.FACTOR} * \text{CFLM}^{n-1}$$

Internal Name = CFLFAC
Default Value = 1.0

CFLM.MAXIMUM ...... This is the maximum value that the CFL multiplier is allowed to be.

Internal Name = CFLMAX
Default Value = 1000000.0

FLUX.FUNCTION.TYPE DUMPSTER allows you to select form the following methods for calculating the inviscid fluxes at the faces of computational cells.

='ROE'	Roes's flux-difference flux function.
='STEGER.WARMING'	Steger-Warming flux splitting.
='HARTEN.YEE'	Harten and Yee flux-difference flux function.

Internal Name = IFLXF
Default Value = 'STEGER.WARMING'

FLUX.FUNCTION.COEFS These coefficients are used in the flux functions above. The definitions of the coefficients and their default values depend upon which flux function is selected. See Section 3

	FLUX.FUNCTION.COEFS(L)						
Flux Function	1	2	3	4	5	6	7
ROE Default Values					c_1 0.05	c_2 0.05	c_3 0.05
STEGER.WARMING Default Values	ϕ_i 1.0	ϕ_j 1.0	κ_i 0.0	κ_j 0.0	c_5 0.05	c_6 0.05	c_7 0.05
HARTEN.YEE Default Values					c_5 0.05	c_6 0.05	c_7 0.05

Internal Name = (FLXCFS(L),L=1,7)

5.10.6 ZONE.INITIAL.CONDITIONS NAMELIST

ZONE.NUMBER	Number of the zone for this NAMESLIST block. The zones must be numbered from bottom to top starting with the number 1. **Internal Name** = NZ **Default Value** = 1
IC.METHOD	This selects the method in which the flow field is initialized for this zone. The flow field conditions are given below. ='UNIFORM.CONDITIONS' Uniform conditions. ='1D.NOZZLE' One-dimensional nozzle solution. The flow through this zone is calculated as if it were isentropic and one-dimensional. The PRESSURE and TEMPERATURE are taken as total. The Mach number at the minimum flow area of the channel is specified as THROAT.MACH.NUMBER. ='USER.SUBROUTINE' User supplied subroutine called USERIC is supplied for the user to add custom coding for initializing the flow in this zone. **Internal Name** = IIC(NZ) **Default Value** = 'UNIFORM'
PRESSURE	May be static or total pressure used for initializing the flow field of this zone depending upon the selection for IC.METHOD. **Internal Name** = PRESI(NZ) **Default Value** = 101325.0
TEMPERATURE	May be static or total temperature used for initializing the flow field of this zone depending upon the selection for IC.METHOD for this zone. **Internal Name** = TEMPI(NZ) **Default Value** = 100.0
U.VELOCITY	x-component of velocity used to initialize the flow field of this zone. **Internal Name** = VVELI(1,NZ) **Default Value** = 100.0
V.VELOCITY	r-component of velocity used to initialize the flow field of this zone. **Internal Name** = VVELI(2,NZ) **Default Value** = 0.0
W.VELOCITY	Swirl-component of velocity used to initialize the flow field of this zone. **Internal Name** = VVELI(3,NZ) **Default Value** = 0.0
TURBULENT.ENERGY .	The turbulent kinetic energy k used to initialize this zone if the k-ϵ turbulence model is selected.

Internal Name	= `TKI(NZ)`
Default Value	= `0.0`

`TURBULENT.DISSIPATION` The turbulent dissipation rate ϵ used to initialize this zone if the k-ϵ turbulence model is selected.

Internal Name	= `TEI(NZ)`
Default Value	= `0.0`

`MASS.FRACTIONS....` The mass fractions of the chemical species initialized in this zone. Enter one value for each species.

Internal Name	= `(FMI(N,NZ),N=1,NSPC)`
Default Value	= `0.0`

`THROAT.MACH.NUMBER` When the `ONE.DIMENSIONAL.NOZZLE` is selected for the initialization method, this variable specifies the Mach number at the minimum area of the zone assuming flow in the i-direction.

Internal Name	= `THMACH(NZ)`
Default Value	= `1.0`

5.10.7 ZONE.BOUNDARY.CONDITIONS NAMELIST

In this NAMELIST block the boundary conditions are selected for the six segments of a zone. Some boundary condition types may only be selected for certain boundary segments. The allowable boundary condition types for each boundary segment are tabulated below.

	Zone Boundaries				Also Specify:
B.C. Type	Top	Bottom	Left	Right	
'FREE.SLIP.WALL'	Y	Y	Y	Y	
'NO.SLIP.WALL'	Y	Y	Y	Y	
'WALL.FUNCTION'	Y	Y	Y	Y	
'INTERZONE'	Y	Y	N	N	
'SUBSONIC.INFLOW'	N	N	Y	N	
'SUPERSONIC.INFLOW'	N	N	Y	N	Static pressure, static temperature, and velocity components
'SUBSONIC.OUTFLOW'	N	N	N	Y	Ambient static pressure (PEX)
'SUPERSONIC.OUTFLOW'	N	N	N	Y	
'MIXED.OUTFLOW'	N	N	N	Y	Ambient static pressure
'PERIODIC'	N	N	Y	Y	
'VORTEX.OUTFLOW'	N	N	N	Y	Ambient static pressure

For supersonic inflows specify static conditions for all variables. For subsonic inflows specify total conditions for pressure and temperature and specify the respective direction cosines for the velocities such that the square root of the sum of the squares of the direction cosines is equal to 1.0.

ZONE.NUMBER .... Number of the zone for this NAMESLIST block. The zones must be numbered from bottom to top starting with the number 1.

Internal Name = NZ
Default Value = 1

BC.TYPE.BOTTOM.S1 Boundary condition type on the first segment on the bottom boundary of the zone.

Internal Name = KBCB(1,NZ)
Default Value = 'FREE.SLIP.WALL'

BC.TYPE.BOTTOM.S2 Boundary condition type on the second segment on the bottom boundary of the zone.

Internal Name = KBCB(2,NZ)
Default Value = 'FREE.SLIP.WALL'

BC.TYPE.BOTTOM.S3 Boundary condition type on the third segment on the bottom boundary of the zone.

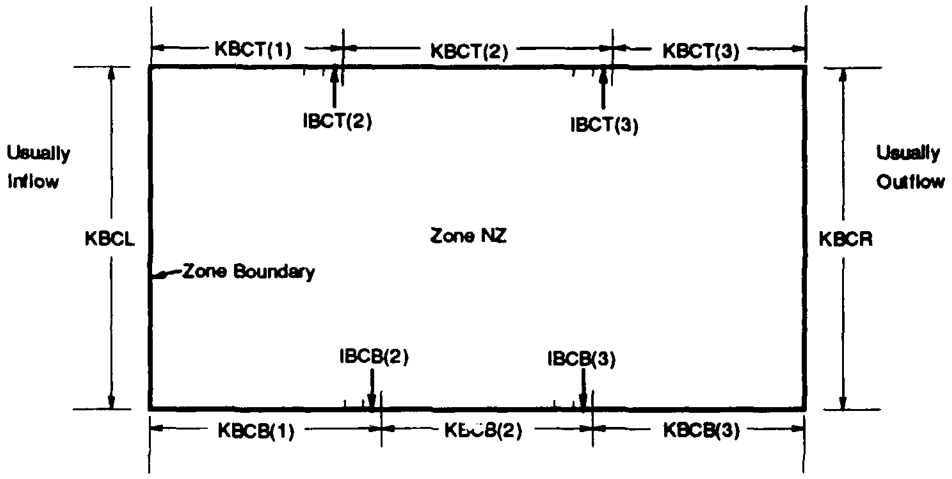

Figure 8: Zone Boundary Condition Segments Shown in Computational Space.

Internal Name	= KBCB(3,NZ)
Default Value	= FREE.SLIP.WALL'

BC.TYPE.TOP.S1.... Boundary condition type on the first segment on the top boudary of the zone.

Internal Name	= KBCT(1,NZ)
Default Value	= 'FREE.SLIP.WALL'

BC.TYPE.TOP.S2.... Boundary condition type on the second segment on the top boundary of the zone.

Internal Name	= KBCT(2,NZ)
Default Value	= 'FREE.SLIP.WALL'

BC.TYPE.TOP.S3.... Boundary condition type on the third segment on the top boundary of the zone.

Internal Name	= KBCT(3,NZ)
Default Value	= 'FREE.SLIP.WALL'

BC.TYPE.LEFT...... Boundary condition type on the left boundary of the zone. This is usually the inflow boundary.

Internal Name	= KBCL(NZ)
Default Value	= 'SUBSONIC.INFLOW'

`BC.TYPE.RIGHT.....` Boundary condition type on the right boundary of the zone. This is usually the outflow boundary.

Internal Name	= `KBCR(NZ)`
Default Value	= `'SUPERSONIC.OUTFLOW'`

`AMBIENT.PRESSURE.` This is the static pressure used for setting the outflow boundary conditions.

Internal Name	= `PEX(NZ)`
Default Value	= `101325.0`

`AMBIENT.TEMPERATURE` This is the static temperature that is used for setting outflow boundary conditions. This is only used for the case when the flow reverses and flows into the boundary.

Internal Na~~m~~e	= `TEX(NZ)`
Default Value	= `273.15`

`WALL.TEMPERATURE.LEFT` This is the wall temperature on the left boundary used for computing the surface heat transfer rate.

`=0.0` If set to zero, the boundary is adiabatic.

=greater than`0.0` If nonadiabatic, set equal to the wall temperature.

Internal Name	= `TWL(NZ)`
Default Value	= `0.0`

`WALL.TEMPERATURE.RIGHT` Same as above for the right boundary.

Internal Name	= `TWR(NZ)`
Default Value	= `0.0`

`WALL.TEMPERATURE.BOTTOM` Same as above for the bottom boundary.

Internal Name	= `TWB(NZ)`
Default Value	= `0.0`

`WALL.TEMPERATURE.TOP` Same as above for the top boundary.

Internal Name	= `TWT(NZ)`
Default Value	= `0.0`

`BC.I.INDEX.BOTTOM` The bottom boundary of the zone is divided into three segments as shown in Figure 8. A different boundary condition type may be specified for each boundary segment. The boundary segments are defined by the ending i–index for each segment. `BC.I.INDEX.BOTTOM(1)` is the ending i–index for segment one and `BC.I.INDEX.BOTTOM(2)` is the ending i-index for segment two.

Internal Name	= `(IBCB(L,NZ),L=2,3)`
Default Value	= `2,2`

BC.I.INDEX.TOP.... Same as above for the top boundary.

Internal Name = (IBCT(L,NZ),L=2,3)
Default Value = 2,2

INTERPOLATION.METHOD The inflow boundary conditions on the left boundary may be specified by a profile for each variable. This profile is defined in a table of data points. The boundary conditions at a cell are calculated from these data points by interpolation. The method used to interpolate from the input data table to the boundary cells may be linear or quadratic as specified here.

='LINEAR' Linear interpolation between data points.

='QUADRATIC' Quadratic interpolation between data points. Quadratic interpolation can result in interpolated points being much different from the values of the source data points from which they are interpolated. This is especially when the source data points are not smooth. When transitioning from one smooth region to another smooth region use several points very close together.

Internal Name = NTRPBC(NZ)
Default Value = 'LINEAR'

NPTS.UVWPT.ARRAY . Number of data points defining the inflow boundary condition profile on left boundary. If only one data point is defined, then a uniform profile is assumed.

Internal Name = NBCVPT(NZ)
Default Value = 1

UVWPT.ARRAY .... This is an array that contains data points defining an inflow boundary condition profile at the left boundary of the zone. The internal name for this array is BCVPT(L,N,NZ) It has three subscripts; the first is for the data point number (L), the second indicates the variable (N) which varies from 1 to NBCVPT(NZ), and the third indicates the zone number (NZ).

BCVPT(L,1,NZ) = y– or r–coordinate
BCVPT(L,2,NZ) = x–velocity
BCVPT(L,3,NZ) = y–velocity
BCVPT(L,4,NZ) = z–velocity
BCVPT(L,5,NZ) = pressure
BCVPT(L,6,NZ) = temperature

The default values are zero for all elements.

Internal Name = (BCVPT(L,N,NZ),N=1,6),L=1,NBCVPT(NZ)
Default Value = 0.0

NPTS.KE.ARRAY..... Number of data points defining the k–ϵ turbulence variables for the inflow boundary condition profile on the left boundary.

Internal Name = NBCTKE(NZ)
Default Value = 1

KE.ARRAY .... This array contains the data points for defining the k–ϵ turbulence variables for the inflow boundary condition on the left boundary of zone NZ. The internal name for this input variable is **BCTKE(L,N,NZ)**. The first subscript denotes the data point number (ranging from 1 to NPTS.KE.ARRAY), the second indicates the variable, and the third indicates the zone number.

BCTKE(L,1,NZ) = y-coordinate (y)
BCTKE(L,2,NZ) = turbulence energy (k)
BCTKE(L,3,NZ) = turbulence dissipation (ϵ)

The default values are zero for all elements.
Internal Name = (BCTKE(L,N,NZ),N=1,2),L=1,NBCTKE(NZ))
Default Value = 0.0

NPTS.S.ARRAY ...... Number of data points defining the chemical species profile for the inflow boundary condition on the left boundary.

Internal Name = NBCSPC(NZ)
Default Value = 1

S.ARRAY .... This array contains the data points for defining the chemical species profile for the inflow boundary condition on the boundary of zone NZ. The internal name for this input variable is BCSPC(L,N,NZ). The first subscript denotes the data point number (ranging from 1 to NPTS.S.ARRAY), the second indicates the variable, and the third indicates the zone number.

BCSPC(L,1,NZ) = y-coordinate
BCSPC(L,2,NZ) = Mass fraction for specie 1 (f_1)
BCSPC(L,3,NZ) = Mass fraction for specie 2 (f_2)
$\vdots$
BCSPC(L,NSPC+1,NZ) = Mass fraction for species NSPC

The default values are zero.
Internal Name =
((BCSPC(L,N,NZ),N=1,NSPC+1),L=1,NBCSPC(NZ))
Default Value = 0.0

5.10.8 ZONE.GEOMETRY NAMELIST

The geometry definition arrays may extend beyond the actual boundaries of the mesh that is to be generated. The geometry arrays must be single-valued functions of x. For nozzle and dump combustor calculations, the mesh must be oriented so that the flow is directed in the positive x–direction.

If NUMBER.OF.POINTS.BOTTOM = 0 then the geometry of the bottom surface is obtained from the user supplied subprogram function **YGEOM(NZ,IBT,XN)**, where: **NZ** is the zone number, IBT is the integer control specifying bottom (IBT=1) or top (IBT=2) boundary, and **XN** is the x–coordinate that is input to the function. The r–coordinate (or y–coordinate) value is returned as the value of **YGEOM**.

ZONE.NUMBER .... Number of the zone for this NAMESLIST block. The zones must be numbered from bottom to top starting with the number 1.

Internal Name = NZ
Default Value = 1

CONVERSION.FACTOR The input coordinates for wall geometry are multiplied by CONVERSION.FACTOR before being used. This input variable can be used for converting the units of the input geometry data to units of meters.

Internal Name = XYCONV(NZ)
Default Value = 1.0

NUMBER.OF.POINTS.BOTTOM The number of data points defining the geometry of the bottom of the zone.

Internal Name = NSB(NZ)
Default Value = 2

INTERPOLATION.METHOD.BOTTOM Selection of the method used for interpolating the actual geometry from the geometry data points input with the **X.BOTTOM** and Y.BOTTOM arrays. See notes on linear and quadratic interpolation for the **INTERPOLATION.METHOD** input variable in the ZONE.BOUNDARY.CONDITIONS NAMELIST block.

='LINEAR'

='QUADRATIC'

Internal Name = NTRPGB(NZ)
Default Value = 'LINEAR'

X.ORIGIN.BOTTOM... This is the origin of the coordinate system used for inputting the geometry for the bottom of the zone. The final x and y coordinates used in the code are equal to the sum of the input values $\hat{x}$ and $\hat{y}$

plus the x and y origins (x_o and y_o) as follows.

$$\begin{aligned} x &= x_o + \hat{x} \\ y &= y_o + \hat{y} \end{aligned}$$

Internal Name = XSTO(NZ)
Default Value = 0.0

Y.ORIGIN.BOTTOM... The y origin for the geometry of the bottom of the zone.

Internal Name = YSTO(NZ)
Default Value = 0.0

X.BOTTOM........... This is a vector of values that define the x–coordinates ($\hat{x}$) of the bottom geometry of the zone. The actual geometry that is used is interpolated form these values and may be shifted and scaled using CONVERSION.FACTOR and X.ORIGIN.

Internal Name = (XSB(L,NZ),L=1,NSB(NZ))
Default Value = 0.0,2.0

Y.BOTTOM........... This is a vector of values that define the y (or r) coordinates ($\hat{y}$) of the bottom geometry of the zone. The actual geometry that is used is interpolated form these values and may be shifted and scaled using CONVERSION.FACTOR and Y.ORIGIN.

Internal Name = (YSB(L,NZ),L=1,NSB(NZ))
Default Value = 0.0,2.0

NUMBER.OF.POINTS.TOP Same as NUMBER.OF.POINTS.BOTTOM except for the geometry at the top of the zone.

Internal Name = NST(NZ)
Default Value = 2

INTERPOLATION.METHOD.TOP Same as NUMBER.OF.POINTS.BOTTOM except for the geometry at the top of the zone.

='LINEAR'

='QUADRATIC'

Internal Name = NTRPGT(NZ)
Default Value = 'LINEAR'

X.ORIGIN.TOP...... Same as NUMBER.OF.POINTS.BOTTOM except for the geometry at the top of the zone.

Internal Name = XSTO(NZ)
Default Value = 0.0

Y.ORIGIN.TOP...... Same as NUMBER.OF.POINTS.BOTTOM except for the geometry at the top of the zone.

Internal Name	= YSTO(NZ)
Default Value	= 0.0

X.TOP .... Same as NUMBER.OF.POINTS.BOTTOM except for the geometry at the top of the zone.

Internal Name	= (XST(L,NZ),L=1,NSB(NZ))
Default Value	= 0.0,2.0

Y.TOP .... Same as NUMBER.OF.POINTS.BOTTOM except for the geometry at the top of the zone.

Internal Name	= (YST(L,NZ),L=1,NSB(NZ))
Default Value	= 0.0,2.0

5.10.9 ZONE.MESH NAMELIST

ZONE.NUMBER .... Number of the zone for this NAMESLIST block. The zones must be numbered from bottom to top starting with the number 1.

Internal Name = NZ
Default Value = 1

NUMBER.OF.CELLS.I The number of internal cells (not counting boundary cells) for this zone in the i–direction. On the Cray computer and other computers with banked memory you should use odd number of cells to prevent bank conflicts when accessing memory. This variable must be consistent with the mesh generation input data defined below, or consistent with an externally generated mesh.

Internal Name = IDIM(NZ)
Default Value = 21

NUMBER.OF.CELLS.J The number of internal cells (not counting boundary cells) in the j–direction. See discussion above.

Internal Name = JDIM(NZ)
Default Value = 21

If an external computational mesh is read from the IMESH file or from the restart file IRSTR (*i.e.* MESH.GENERATION.MODE = 'EXTERNAL' or 'RESTART'), then the mesh generation data below will be ignored.

The internal mesh generation routine creates meshes that have straight vertical i–lines (constant i index). The mesh is generated in four segments in the i–direction (which is often the same as the x–direction) and one segment in the j–direction a schematic diagram is shown in Figure 9.

Do not include boundary cells when calculating the following input variables.

The i–index at the first interior cell on the left boundary is three, since their are two boundary cells. The i–index of the interior cell that is farthest to the right on the right boundary has an i–index of ID-2. ID is the total number of interior and exterior cells in the i–directions. JD is the total number of interior and exterior cells in the j–directions. Subtract 4 from ID or JD to calculated the number of internal cells in the i or j–direction.

X.CENTER........... x–coordinate of the line between left and right center regions as shown in Figure 9.

Internal Name = XCTR(NZ)
Default Value = 0.0

DELTA.X .... x–direction mesh increment at X.CENTER.

Internal Name = DELX(NZ)
Default Value = 1.0

X.STRETCH.MODE.... Selection of mesh stretching mode in the x–direction.

='STRETCH.FACTORS' Specify stretch factors for the STRETCH.LENGTH.x (where x is LEFT, RIGHT, LEFTCENTER, or RIGHTCENTER) input variables below. The mesh increments will be stretched away from the X.CENTER.

='LENGTHS' Specify lengths of each region for the STRETCH.LENGTH input variables below. The mesh spacing is calculated such that the number of cells specified in the NUMBER.OF.CELLS.x (where x is LEFT, RIGHT, LEFTCENTER, or RIGHTCENTER) input variables will extend the specified length.

Internal Name = ISW(NZ)
Default Value = 'LENGTHS'

NUMBER.OF.CELLS.LEFT The number of cells in the segment that is on the left boundary.

Internal Name = NCX(1,NZ)
Default Value = 0

NUMBER.OF.CELLS.LEFTCENTER The number of cells in the segment that is just left of X.CENTER.

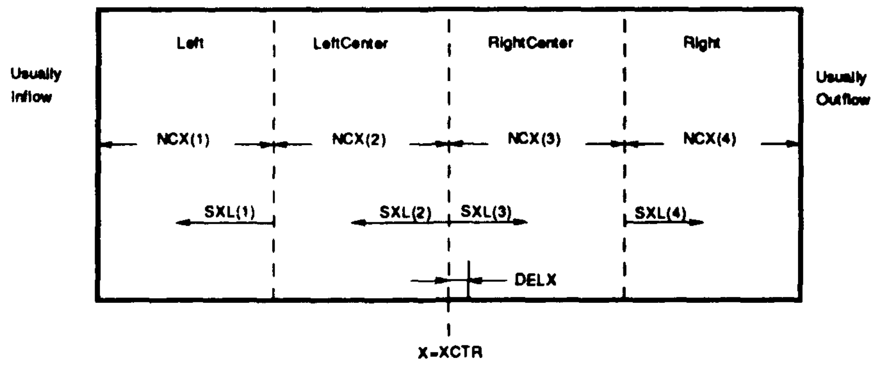

Figure 9: Mesh Generation Schematic for the x-Direction.

Internal Name = NCX(2,NZ)
Default Value = DEF=0

NUMBER.OF.CELLS.RIGHTCENTER The number of cells in the segment that is just right of X.CENTER.

Internal Name = NCX(3,NZ)
Default Value = DEF=0

NUMBER.OF.CELLS.RIGHT The number of cells in the segment that is on the right boundary.

Internal Name = NCX(4,NZ)
Default Value = 0

STRETCH.LENGTH.LEFT This is either the stretch factor (E) or the length of the LEFT segment depending upon X.STRETCH.MODE.

Internal Name = SLX(1,NZ)
Default Value = 1.0

STRETCH.LENGTH.LEFTCENTER This is either the stretch factor (E) or the length of the LEFTCENTER segment depending upon X.STRETCH.MODE.

Internal Name = SLX(2,NZ)
Default Value = 1.0

STRETCH.LENGTH.RIGHTCENTER This is either the stretch factor (E) or the length of the RIGHTCENTER segment depending upon X.STRETCH.MODE.

Internal Name = SLX(3,NZ
Default Value = 1.0

STRETCH.LENGTH.RIGHT This is either the stretch factor (E) or the length of the RIGHT segment depending upon X.STRETCH.MODE.

Internal Name = SLX(4,NZ)
Default Value = 1.0

J.STRETCH.MODE.... This is the selection for the mode in which the mesh cell spacing in the vertical direction is calculated.

='STRETCH.FACTORS' Specify stretch factors and the number of blocks of j-cells.

='DELY.BOTTOM' Specify DELY.BOTTOM at the bottom of the zone ($j = 3$) and the number of j-cells
in J.BLOCK.NUMBER.OF.CELLS(1). The code will then calculate a stretch factor to fill the distance from the bottom to the top of the zone.

='DELY.TOP' Specify DELY.TOP at top of the zone (j = JDM) and the number of j-cells in J.BLOCK.NUMBER.OF.CELLS(1). The code will then calculate a stretch factor to fill the distance from the top to the bottom of the zone.

='DELY.TOP.AND.BOTTOM' Specify DELY.TOP at top of the zone (j = JDM), DELY.BOTTOM at the bottom of the zone, and the number of j–cells in J.BLOCK.NUMBER.OF.CELLS(1). The code will then calculate a stretch factor to fill the distance between the top and the bottom of the zone.

Internal Name = JSW(NZ)
Default Value = 'STRETCH.FACTORS'

NUMBER.OF.J.BLOCKS The number of j–blocks of cells in zone. The maximum number is 5. The stretch factor is constant within each j–block. The y–increment in the lowest cell of blocks is equal to the y–increment of the cell below it times the stretch factor in the block. This variable is ignored if J.STRETCH.MODE is not set to STRETCH.FACTORS.

Internal Name = NJBK(NZ)
Default Value = 1

J.BLOCK.NUMBER.OF.CELLS The number of j-cells in j–block L, where L goes from 1 to NUMBER.OF.J.BLOCKS. The first block is at the bottom of the zone. j–blocks 2 through NUMBER.OF.J.BLOCKS are stacked above it.

Internal Name = (JBK(L,NZ),L=1,NJBK)
Default Value = 21

J.BLOCK.NUMBER.OF.I.COLUMNS The number of i–cells at which the following y–mesh parameters are defined. The number of j–blocks (NUMBER.OF.J.BLOCKS) and the number of j–cells in each j–block are the same at each i–cell. The maximum is five.

Internal Name = NIJBK(NZ)
Default Value = 1

J.BLOCK.I.COLUMNS The i–cells at which the following j–mesh parameters are defined. The first i–cell must equal 1 and the last must ID.

Internal Name = (IJBK(L,NZ),L=1,NIJBK)
Default Value = 1

J.BLOCK.DELY.BOTTOM The y–cell increment at the bottom of the zone that is used to generate the y–mesh. This is used if J.STRETCH.MODE is set to DELY.BOTTOM or DELY.TOP.AND.BOTTOM.

Internal Name = (DYBJBK(L,NZ),L=1,NIJBK)
Default Value = 0.0

J.BLOCK.DELY.TOP . The y cell increment at the top of zone that is used to generate the y-mesh. This is used if J.STRETCH.MODE is set to DELY.TOP or DELY.TOP.AND.BOTTOM.

Internal Name = (DYTJBK(L,NZ),L=1,NIJBK)
Default Value = 0.0

`J.BLOCK.STRETCH.FACTORS` Stretch factors defined for j–blocks at the i–cells defined by `J.BLOCK.I.COLUMNS`. This variable is a two-dimensional array. The first subscript varies from 1 to the number of `NIJBK` and the second subscript varies from 1 to `NJBK`. The stretch factor in each block of j–cells is constant throughout the block. This variable is ignored if `J.STRETCH.MODE` is not equal to `STRETCH.FACTORS`.

Internal Name = `((EJBK(L,J,NZ),L=1,NIJBK),J=1,NJBK)`
Default Value = 1

5.10.10 ZONE.OUTPUT NAMELIST

ZONE.NUMBER	Number of the zone for this NAMESLIST block. The zones must be numbered from bottom to top starting with the number 1. **Internal Name** = NZ **Default Value** = 1
NUMBER.OF.I.COLUMNS	The minor printout occurs during the calculation for a select number of columns of *i*–cells. This variable specifies the number of *i*–cell columns to print. **Internal Name** = NCOLPRT(NZ) **Default Value** = 0
I.COLUMNS..........	This is an array of *i*–cell columns that will be printed in the minor printout. **Internal Name** = (ICOLPRT(L,NZ),L=1,NCOLPRT(NZ)) **Default Value** = 2
NUMBER.OF.TAPS....	You may select computational cells in the zone where the flow field values will be monitored during the calculation. At the end of the calculation the tap values are formated for input into TECPLOT. Tap data allows you to plot time histories of various flow field variables at specific places in the flow field. **Internal Name** = NTAPS(NZ) **Default Value** = 0
TAP.I.INDEX	This is an array of *i*–indexes of the taps for this zone. **Internal Name** = (ITAP(L,NZ),L=1,NTAPS) **Default Value** = 1
TAP.J.INDEX	This is an array of *j*–indexes of the taps for this zone. **Internal Name** = (JTAP(L,NZ),L=1,NTAPS) **Default Value** = 1

5.[illegible] 1 Summary of Input Variables

The following is a listing of the input data showing the format of the input data. The values shown are the default values.

```
.............................................................................
$CONTROL
  EXECUTION.MODE              = "NEW",
  NUMBER.OF.STEPS             = 0,
  CONVERGENCE.TOLERANCE       = -6.0,
  CPU.SECONDS.MAXIMUM         = 99999.9,
  NUMBER.OF.ZONES             = 1,
  COORDINATE.SYSTEM           = "AXISYMMETRIC",
  DEBUG.FLAGS                 = 10*0,
$END
.............................................................................
$PROPERTIES
  THERMODYNAMIC.MODEL         = "PERFECT.GAS",
  GAMMA                       = 1.4,
  PERFECT.GAS.CONSTANT        = 287.0,
  SPECIFIC.HEAT.COEFS         = 0.0, 1.0, 0.0, 717.5,
  VISCOSITY.MODEL             = "INVISCID",
  THIN.LAYER.OPTION           = "OFF",
  LAMINAR.VISCOSITY.COEFS     = 1.4519E-06, 1.5, 0.0, 0.0, 1.0, 110.0,
  KE.CONSTANTS                = 1.44, 1.92, 0.09, 1.0, 1.3,
  CONDUCTIVITY.MODEL          = "NONCONDUCTING",
  LAMINAR.PRANDTL.NUMBER      = 0.7,
  TURBULENT.PRANDTL.NUMBER    = 0.9,
  CONDUCTIVITY.COEFS          = 0.0020541, 1.5, 0.0, 0.0, 1.0, 110.0,
  NUMBER.OF.SPECIES           = 0
  SPECIES.COEFS               = 5*0.0,
$END
.............................................................................
$MESH
  MESH.GENERATION.MODE        = "INTERNAL",
  ADAPT.STEP.BEGIN            = 99999,
  ADAPT.STEP.END              = 99999,
  ADAPT.STEP.INTERVAL         = 99999,
  ADAPT.RELAXATION.FACTOR     = 0.2,
$END
.............................................................................
$OUTPUT
  QUICK.PRINT.FREQUENCY       = 99999
  NUMBER.OF.PRINTS            = 0
  STEPS.TO.PRINT              = 100*0,
  NUMBER.OF.PLOTS             = 0
  STEPS.TO.PLOT               = 100*0,
  MAJOR.PRINT.SKIP.I          = 999
  MAJOR.PRINT.SKIP.J          = 999
  PRINT.PROPERTIES            = "NO"
  PRINT.METRICS               = "NO"
$END
...........................................................................
$NUMERICS
  TS.RELAXATION.FACTOR        = 1.0,
  LU.BETA                     = 1.0,
  TIME.STEP.MODE              = "LOCAL",
  DU.CHANGE.MAXIMUM           = 1.0,
  CFLM.BEGIN                  = 1.0,
  CFLM.FACTOR                 = 1.0,
  CFLM.MAXIMUM                = 1.0E+09,
  FLUX.FUNCTION.TYPE          = "ROE",
  FLUX.FUNCTION.COEFS         = 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
$END
...........................................................................
$ZONE.INITIAL.CONDITIONS
  ZONE.NUMBER                 = 1,
  IC.METHOD                   = "UNIFORM.CONDITIONS",
  RRESSURE                    = 1.01325E+05,
  TEMPERATURE                 = 100.0,
  U.VELOCITY                  = 100.0,
  V.VELOCITY                  = 0.0,
  W.VELOCITY                  = 0.0,
  THROAT.MACH.NUMBER          = 1.0,
  TURBULENT.ENERGY            = 0.0,
  TURBULENT.DISSIPATION       = 0.0,
  MASS.FRACTIONS              = 0.0,
$END
...........................................................................
$ZONE.BOUNDARY.CONDITIONS
  ZONE.NUMBER                 = 1,
  BC.TYPE.BOTTOM.S1           = "FREE.SLIP.WALL",
  BC.TYPE.BOTTOM.S2           = "FREE.SLIP.WALL",
  BC.TYPE.BOTTOM.S3           = "FREE.SLIP.WALL",
  BC.TYPE.TOP.S1              = "FREE.SLIP.WALL",
  BC.TYPE.TOP.S2              = "FREE.SLIP.WALL",
  BC.TYPE.TOP.S3              = "FREE.SLIP.WALL",
  BC.TYPE.LEFT                = "SUBSONIC.INFLOW",
  BC.TYPE.RIGHT               = "SUPERSONIC.OUTFLOW",
  AMBIENT.PRESSURE            = 1.01325E+05,
  AMBIENT.TEMPERATURE         = 273.15,
  BOUNDARY.CONDITION.COEFS       = 9*0.0,
  WALL.TEMPERATURE.LEFT          = 0.0,
  WALL.TEMPERATURE.RIGHT         = 0.0,
  WALL.TEMPERATURE.BOTTOM        = 0.0,
  WALL.TEMPERATURE.TOP           = 0.0,
  BC.I.INDEX.BOTTOM              = 2, 2,
  BC.I.INDEX.TOP                 = 2, 2,
  INTERPOLATION.METHOD           = "LINEAR",
  NPTS.UVWPT.ARRAY               = 1,
  UVWPT.ARRAY                    = 6*0.0,
  NPTS.KE.ARRAY                  = 1,
  KE.ARRAY                       = 3*0.0,
  NPTS.S.ARRAY                   = 1,
  S.ARRAY                        = 0.0,
$END
...........................................................................
$ZONE.GEOMETRY
  ZONE.NUMBER                    = 1,
  CONVERSION.FACTOR              = 1.0,
  NUMBER.OF.POINTS.BOTTOM        = 2,
  INTERPOLATION.METHOD.BOTTOM    = "LINEAR",
  X.ORIGIN.BOTTOM                = 0.0,
  Y.ORIGIN.TOP                   = 0.0,
  X.BOTTOM                       = 50*0.0,
  Y.BOTTOM                       = 50*0.0,
  NUMBER.OF.POINTS.TOP           = 2,
  INTERPOLATION.METHOD.TOP       = "LINEAR",
  X.ORIGIN.TOP                   = 0.0,
  Y.ORIGIN.TOP                   = 0.0,
  X.TOP                          = 50*0.0,
  Y.TOP                          = 50*0.0,
$END
...........................................................................
$ZONE.MESH
  ZONE.NUMBER                    = 1,
  NUMBER.OF.CELLS.I              = 21,
  NUMBER.OF.CELLS.J              = 21,
  ADAPT.ZONE                     = "NO",
  X.CENTER                       = 0.0,
  DELTA.X                        = 1.0,
  X.STRETCH.MODE                 = "STRETCH.FACTORS",
  NUMBER.OF.CELLS.LEFT           = 0,
  NUMBER.OF.CELLS.LEFTCENTER     = 0,
  NUMBER.OF.CELLS.RIGHTCENTER    = 21,
  NUMBER.OF.CELLS.RIGHT          = 0,
  STRETCH.LENGTH.LEFT            = 1.0,
  STRETCH.LENGTH.LEFTCENTER    = 1.0,
  STRETCH.LENGTH.RIGHTCENTER   = 1.0,
  STRETCH.LENGTH.RIGHT         = 1.0,
  J.STRETCH.MODE               = "STRETCH.FACTORS",
  NUMBER.OF.J.BLOCKS           = 1,
  J.BLOCK.NUMBER.OF.CELLS      = 21,
  J.BLOCK.NUMBER.OF.I.COLUMNS  = 1,
  J.BLOCK.I.COLUMNS            = 1,
  J.BLOCK.DELY.BOTTOM          = 0.0,
  J.BLOCK.DELY.TOP             = 0.0,
  J.BLOCK.STRETCH.FACTORS      = 1.0,
$END
............................................................................
$ZONE.OUTPUT
  ZONE.NUMBER                  = 1,
  NUMBER.OF.I.COLUMNS          = 0,
  I.COLUMNS                    = 10*2,
  NUMBER.OF.TAPS               = 0,
  TAP.I.INDEX                  = 20*1,
  TAP.J.INDEX                  = 20*1,
$END
```

6 DUMP COMBUSTOR CALCULATIONS

In this section the results of several calculations using DUMPSTER are described. The first section describes the results of our study on the use of several flux functions. The next section presents the results of a set of calculations of a dump combustor with comparisons to experimental data and another numerical flow analysis program. In the last section the results of the application of DUMPSTER to a dump combustor that is closely coupled with a CD nozzle are presented.

6.1 Comparison of Flux Functions

The LGS method requires a flux function that is sufficiently dissipative so that the Gauss-Seidel iteration process converges. The LGS method was used with the naturally dissipative Steger-Warming flux function and with Roe's flux function with eigenvalue smoothing. The added eigenvalue smoothing not only helps the Gauss-Seidel iteration converge, but also helps prevent aphysical phenomena like shock expansions. The use of added dissipation and the naturally dissipative flux function by Steger-Warming raised concerns about the accuracy of the code especially for resolving viscous shear layers.

Additional numerical dissipation tends to diffuse shear layers. In this study several flux functions were used to calculated a laminar boundary-layer flow on a flat plate for a freestream Mach number of 2.0, laminar Prandtl number of 0.7, and a viscosity relation as follows.

$$\mu = \mu_\infty \left(\frac{T}{T_\infty}\right)$$

This case was selected because an approximate analytical solution is available for comparison[41]. The flow field schematic is shown in Figure 10. A 20x20 mesh was used with the inflow boundary of the domain at the leading edge of the plate. The calculated velocity and temperature profiles are compared to the approximate analytical solution in Figures 11 through 14. The second-order Roe's scheme with $\kappa = 0.0$ is seen to give the best agreement with the analytical solution, and the first-order Steger Warming flux splitting scheme the worst. The Steger Warming scheme is particularly poor in resolving the temperature profiles in all cases. The first-order Roe's scheme is substantially better than the first and second-order Steger Warming schemes.

Since the LU-SGS solution method does not require a flux function with as much dissipation as the LGS method, the Steger-Warming flux function is used only for

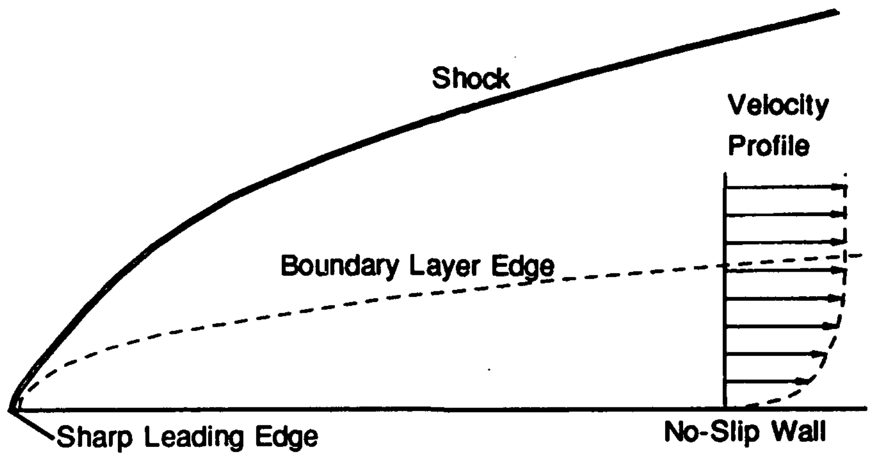

Figure 10: Schematic Diagram of Test Case of Flat Plate Boundary Layer with Freestream Mach Number of 2.0.

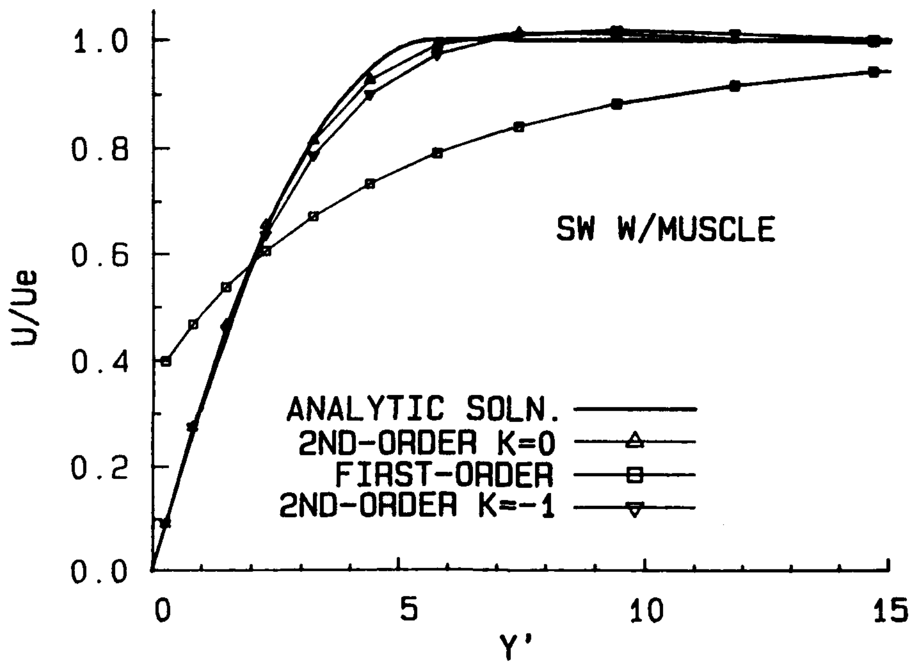

Figure 11: Comparison of Analytical and Calculated Velocity Profiles Using Steger-Warming Fluxes for Flat Plate Boundary Layer.

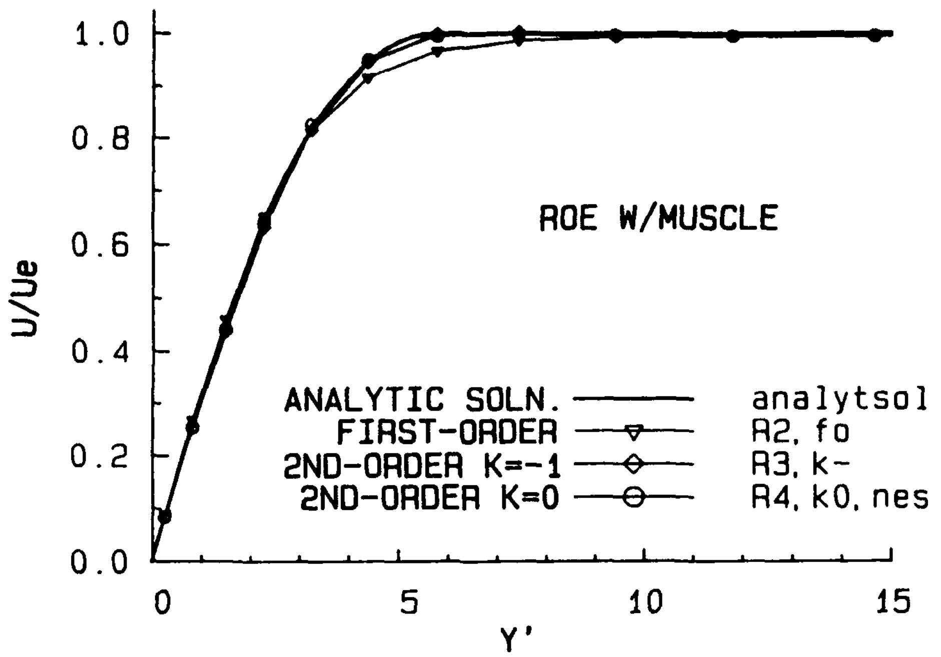

Figure 12: Comparison of Analytical and Calculated Velocity Profiles Using Roe's Fluxes for Flat Plate Boundary Layer.

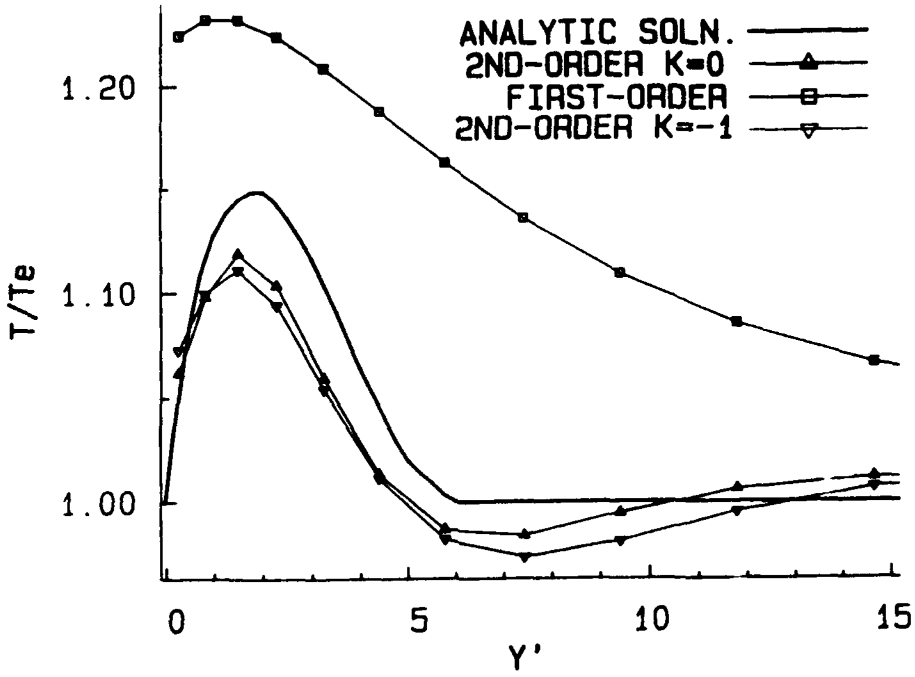

Figure 13: Comparison of Analytical and Calculated Temperature Profiles Using Steger-Warming's Fluxes for Flat Plate Boundary Layer.

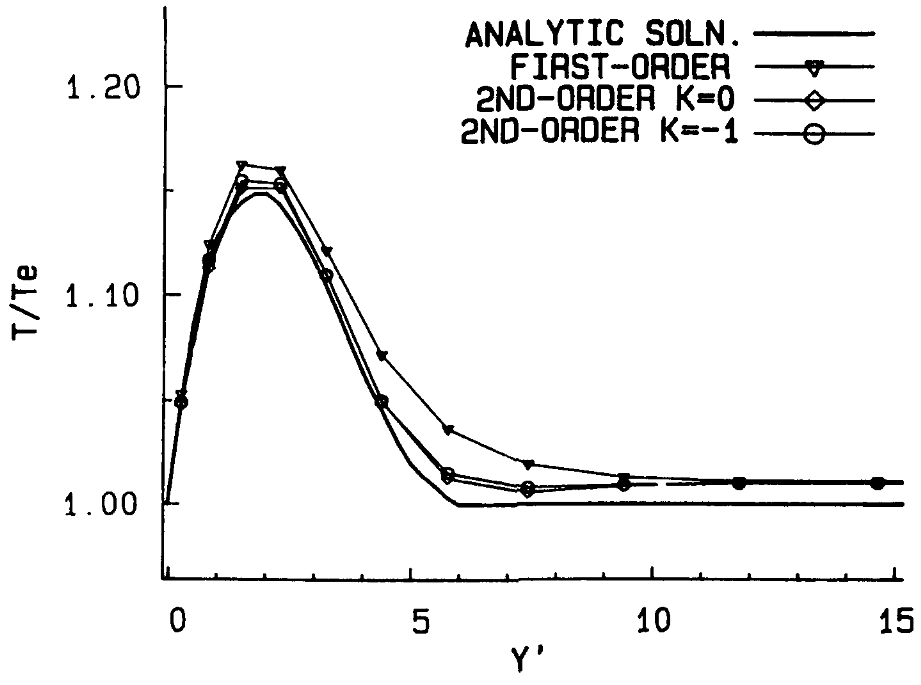

Figure 14: Comparison of Analytical and Calculated Temperature Profiles Using Roe's Fluxes for Flat Plate Boundary Layer.

"roughing out" poor initial conditions. For low-speed flows the LU-SGS method can be used with either the Roe's or Harten-Yee's flux functions without significant additive dissipation.

6.2 Dump Combustor Calculations

In this section calculated results from DUMPSTER are presented for the axisymmetric non-reacting dump combustor shown in Figure 15. This configuration is a simple 50.8mm radius round pipe dumping into a larger concentric pipe with a radius of 76.2mm. The step height, H, is the difference in radii of the two pipes 25.4mm (1.0 inch). Three different swirl numbers are considered: $S = 0.0$ (no swirl), $S = 0.3$, and $S = 0.5$. Calculations used the k-ϵ turbulence model with the standard constants and with modified constants. The computed results are compared to experiment and to calculations using a different flow analysis computer code[42].

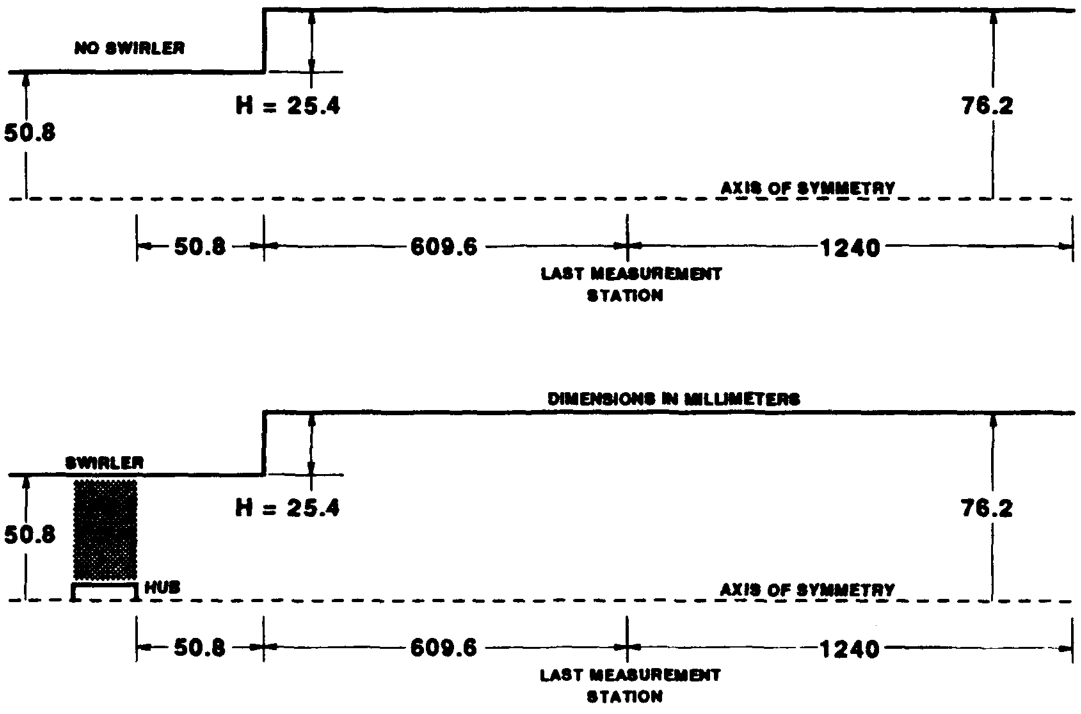

Figure 15: Axisymmetric Dump Combustor Geometry With Swirler (bottom) and Without Swirler (top).

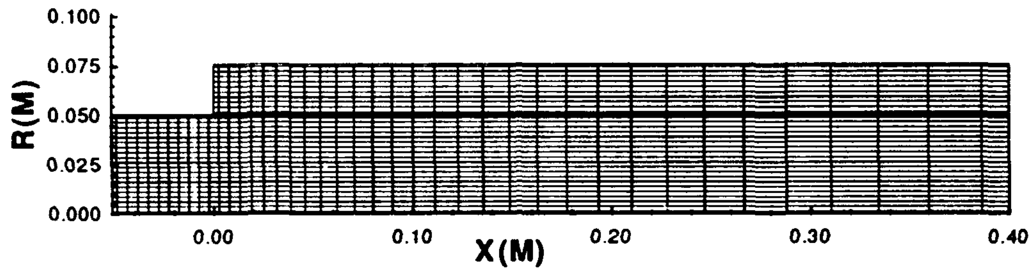

Figure 16: Two-Zone Computational Mesh for Dump Combustor with $S = 0$.

6.2.1 No Swirl: $S = 0.0$

The computational mesh used in this calculation is shown in Figure 16. In this calculation, a two-zone mesh was used consisting of 42x22 mesh cells in the lower zone (No. 1), and 32x12 in the upper zone (No. 2) mesh. The domain extends sixteen step heights downstream of the step (the dump). At the inlet, uniform profiles of velocity direction, total pressure, and total temperature are imposed as boundary conditions. The upper and lower wall boundary conditions are free-slip, non-permeable walls. At the subsonic outflow boundary the pressure was fixed to P_{exit}.

$$\begin{aligned}
P_T &= 101500.0\ Pa \\
T_T &= 300.6\ ^{o}K \\
P_{\text{exit}} &= 101325.0\ Pa(\text{on upper wall}) \\
u/|\vec{V}| &= 1.0 \\
v/|\vec{V}| &= 0.0 \\
w/|\vec{V}| &= 0.0 \\
k &= 0.5\ m^2/s^2 \\
\epsilon &= 23.0\ m^2/s^3
\end{aligned}$$

The Reynolds number based upon the combustor inlet diameter (101.6 mm) was approximately 125,000. Calculated velocity vectors, streamlines, and contours of turbulent kinetic energy are shown in Figures 17, 18, and 19. The corner recirculation zone is about six step heights long as compared to eight step heights found exper-

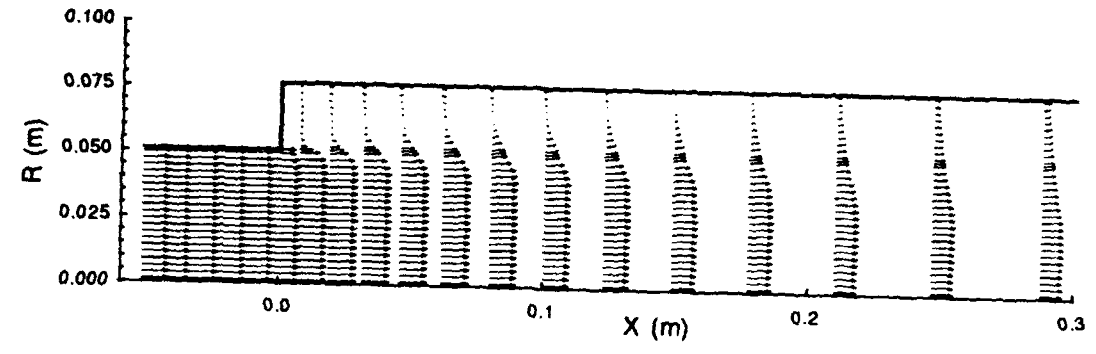

Figure 17: Calculated Axial-Radial Velocity Vectors (U-V) for $S = 0$.

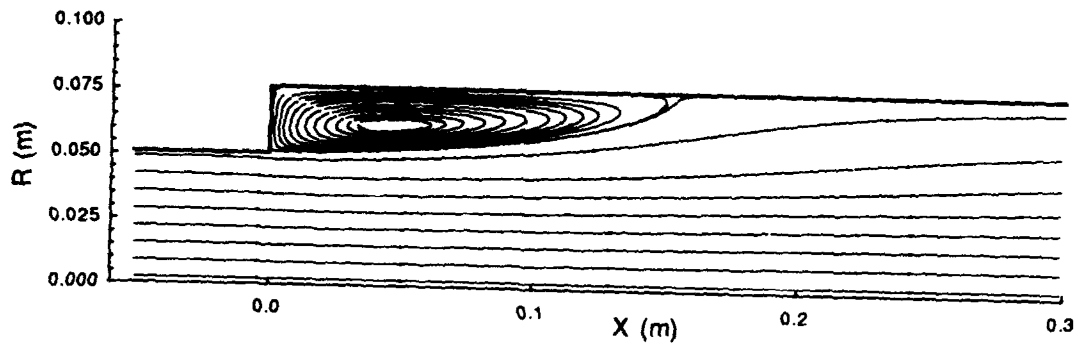

Figure 18: Calculated U-V Streamlines for $S = 0$.

imentally. This is apparently due to the use of the standard constants in the k-ϵ model used in the calculation. Some numerical experiments are described in the next section that show the sensivity of the corner recirculation zone to the k-ϵ constants.

Figure 20 shows profiles of measured and computed mean axial velocities at eight axial locations. U_R is a reference velocity approximately 560 mm upstream of the dump plane. U_R has a value of approximately 19.2 m/s. Due to the free-slip boundary conditions used on the combustor wall, some disagreement is shown at regions near the dump plane where a secondary recirculating region is not predicted in the computed result.

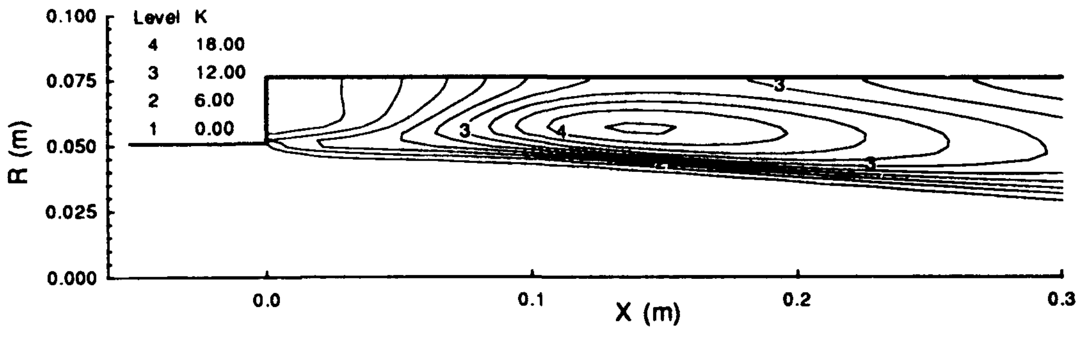

Figure 19: Calculated Turbulent Kinetic Energy Contours for $S = 0$.

Figure 21 shows profiles of measured and calculated turbulent kinetic energy (k) at the same axial locations. U_R^2 is used to nondimensionalize the kinetic energy. From these figures the standard k–ϵ–model appears to overpredict the turbulent kinetic energy which results in higher eddy viscosity through out the region.

This viscosity overprediction is evidenced by a shorter calculated corner recirculation region (Figure 18). However, the overall physics of the no-swirl dump combustor is captured with fair agreement with experimental data using the k–ϵ model with the standard values for the k–ϵ constants.

The deviation of the calculated u and k profiles at $X/H = 0.38$ indicated that the boundary condition inflow velocity profile was not uniform as was used in the calculations above. To investigate the influence of the upstream boundary conditions upon the flow this case was rerun with the following inflow total pressure distribution.

Radial Position (mm)	Total Pressure (Pa)
00.0	1.01470×10^5
10.2	1.01470×10^5
20.0	1.01465×10^5
25.4	1.01462×10^5
35.6	1.01450×10^5
45.7	1.01405×10^5
48.3	1.01380×10^5
50.8	1.01330×10^5

The above total pressure profile was derived by trial and error. It was adjusted until

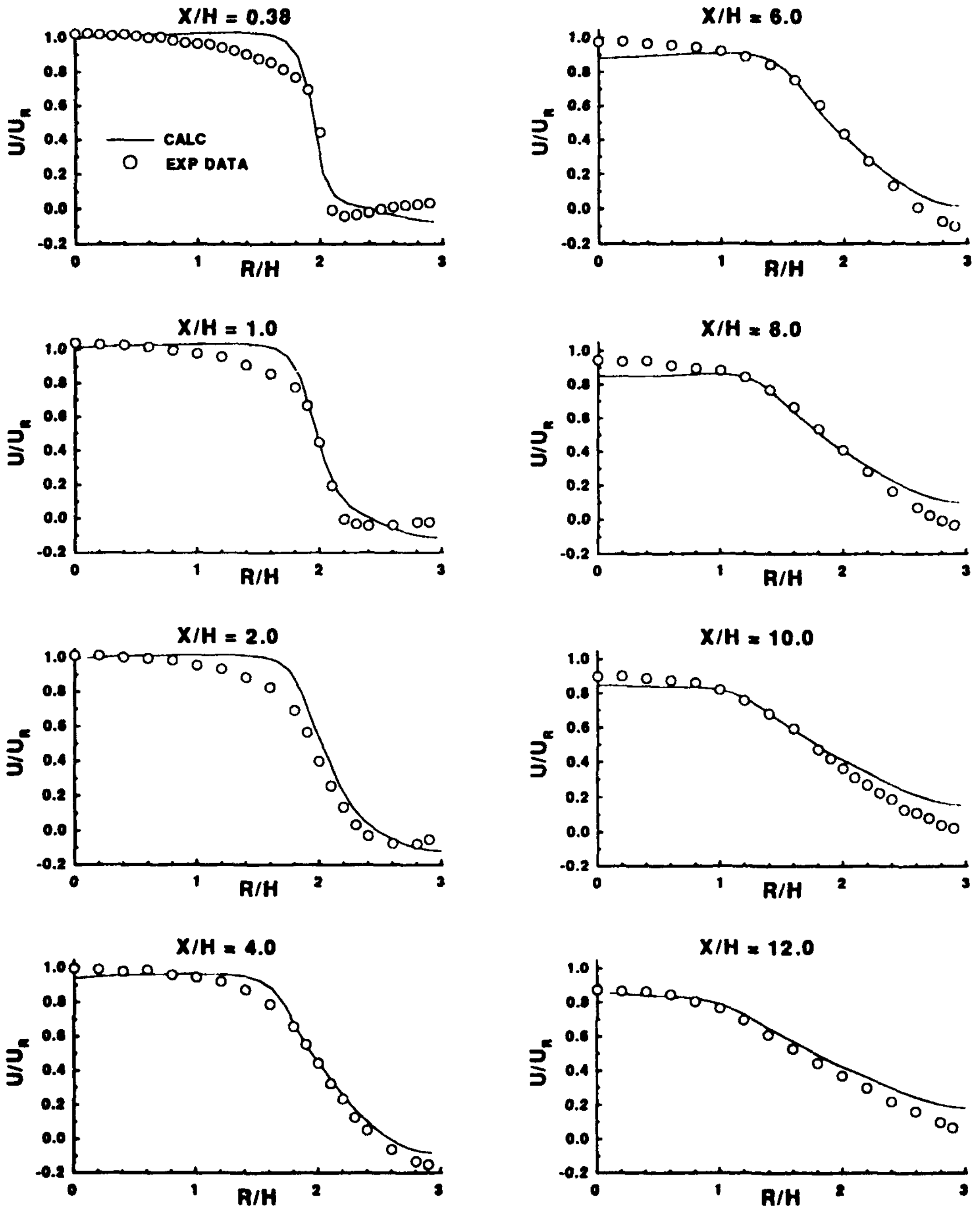

Figure 20: Axial Velocity Profile for $S = 0.0$.

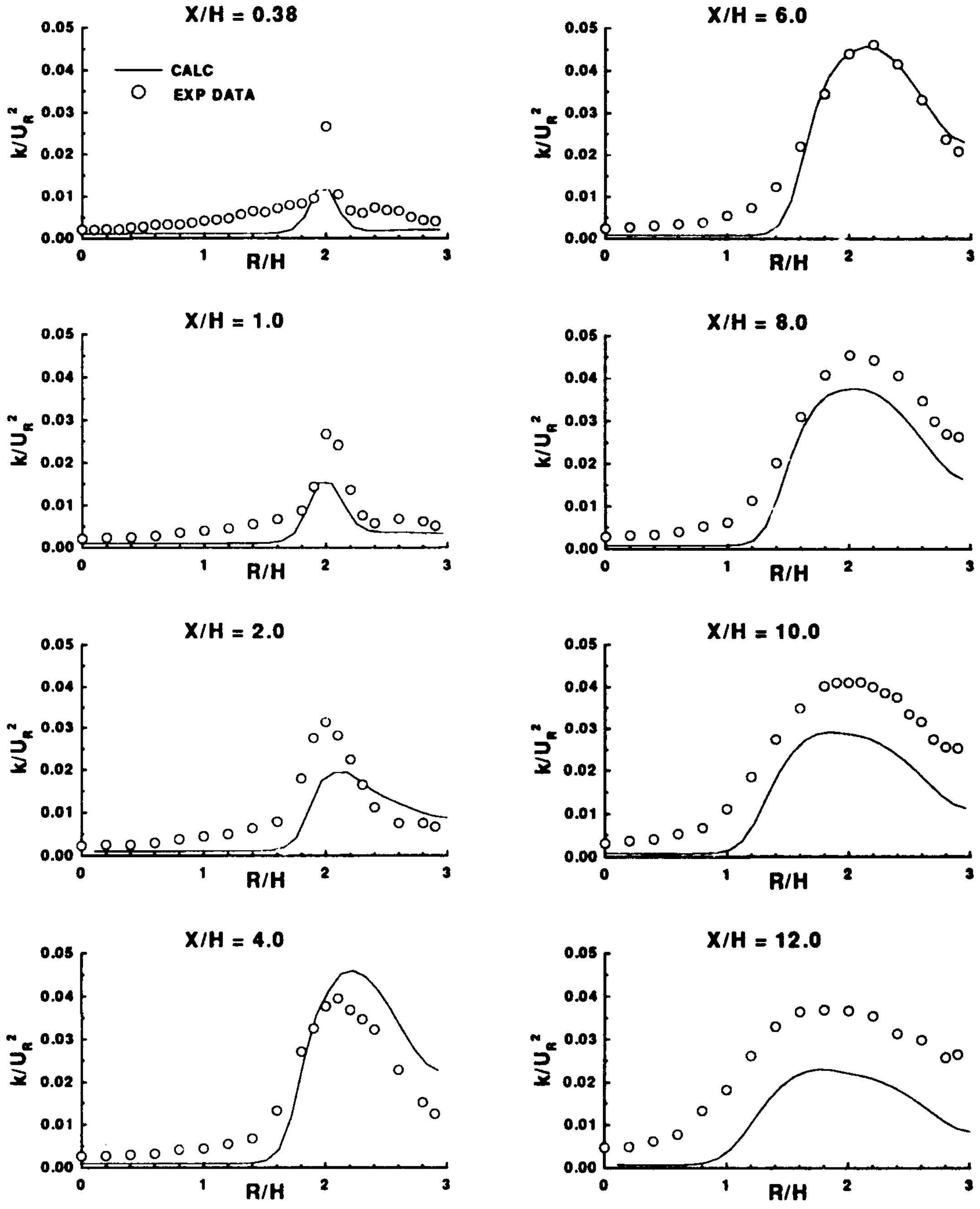

Figure 21: Turbulent Kinetic Energy Profiles for $S = 0.0$.

the calculated axial velocity profile best fit the experimentally measured profile at $X/H = 0.38$. The comparisons of the calculated profiles to the measured profiles are shown in Figures 22 and 23. Although the axial velocity profiles agree better at $x/H = 0.38$ and the turbulenct kinetic energy was generally better predicted, the calculated axial velocity downstream of $x/H = 2.0$ agreed less well. The reason for the deviation between the calculated and measured profiles is due to a variety of things including at least: the inflow boundary conditions, wall boundary conditions, turbulence model, and numerical effects. To reduce the number of variables in the study all following calculations were performed with a uniform inflow total pressure profile. The fact that other factors have significant influence upon the calculated flow field must be considered when comparing the calculations with experimental data.

6.2.2 Moderate Swirl: $S = 0.3$

The dump combustor flow was calculated for a swirl number of 0.3. In this calculation three zones were used to construct the computational mesh shown in Figure 24. The effects of the hub ring were modeled by using the first zone (bottom) (which consists of 42x6 mesh cells) in the 9 *mm*-high region between the axis of symmetry and the outer radius of the hub. The mesh in zone 2 consists of 42x19 mesh cells and the top zone (zone 3) consists of 32x12 mesh cells. All zones extended approximately sixteen step heights downstream of the dump plane.

The inflow and outflow boundary conditions were the same as for the $S = 0.0$ case with uniform inflow boundary conditions except for the inflow swirl angle which was set to a swirl number of 0.3 between the hub and the outer radius of the inlet. Free-slip walls were used on all solid boundaries. The direction cosines for the inflow velocity are as follows:

$$\begin{aligned} u/|\vec{V}| &= 0.9164 \\ v/|\vec{V}| &= 0.0 \\ w/|\vec{V}| &= 0.4 \end{aligned}$$

Results for $S = 0.3$ case are shown in Figure 25 through 28. Figures 29 and 30 show profiles of measured and calculated mean axial velocities (u) and swirl velocities (w) at eight axial locations for $S = 0.3$. The agreement between the computation and experiment is reasonably good. Significant discrepancies between calculated and measured velocity profiles appear downstream of $x/H = 6.0$. These disrepancies are explained by examining the turbulent kinetic energy profiles at the same locations (Fig. 31). The measured high level of turbulent kinetic energy near the axis of symmetry is not predicted by the k-ϵ model for downstream stations. This behavior has

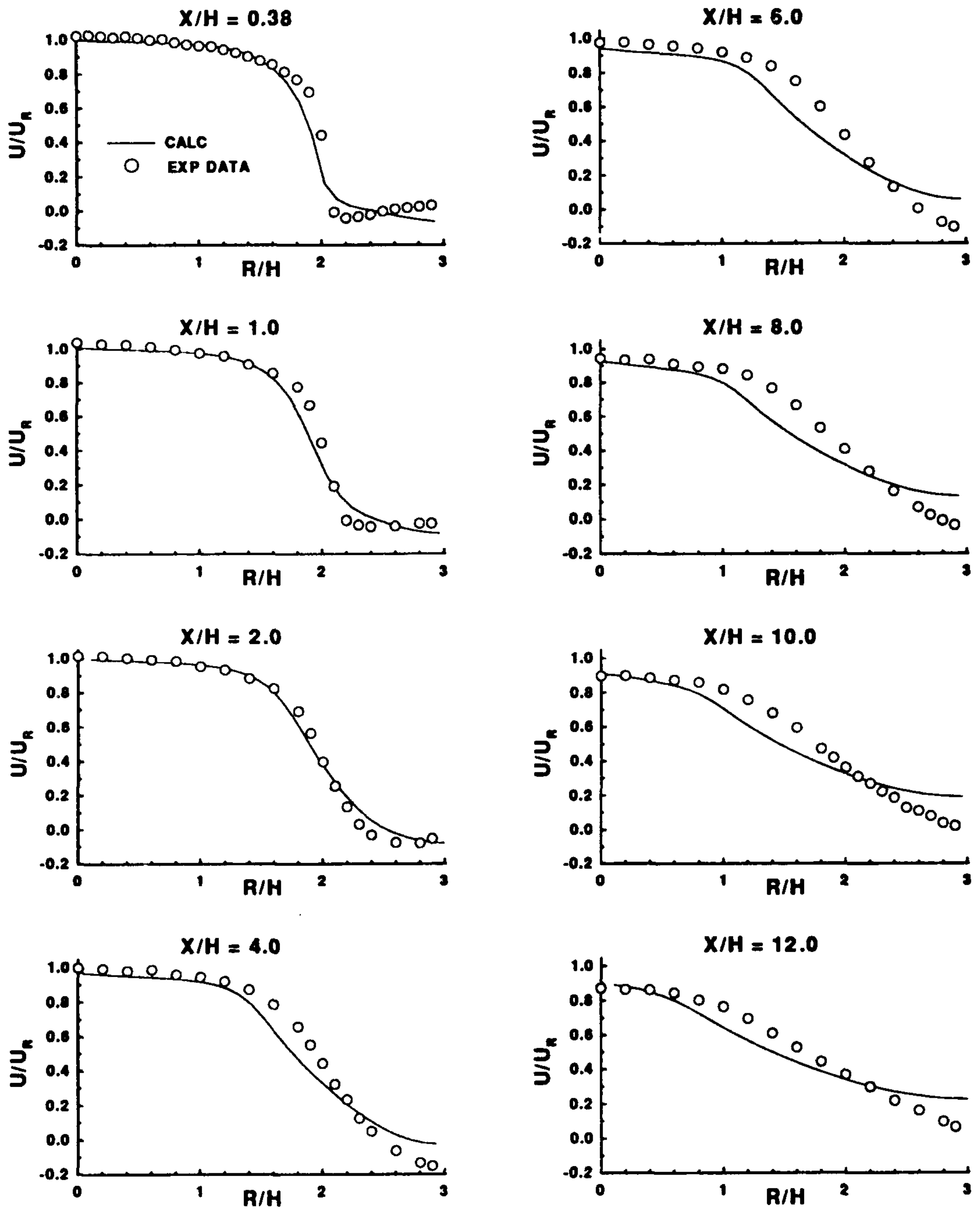

Figure 22: Axial Velocity Profiles for $S = 0.0$.

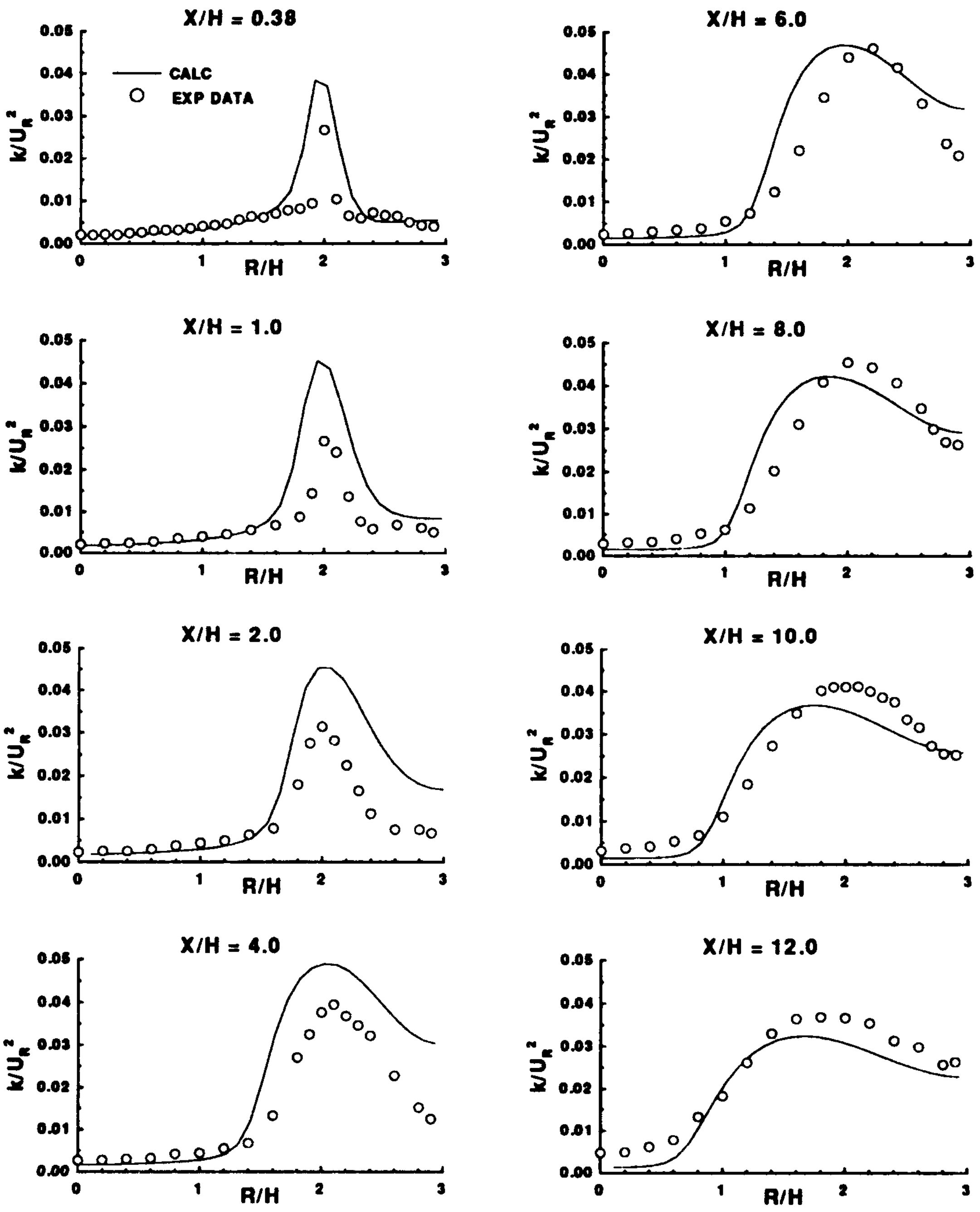

Figure 23: Turbulent Kinetic Energy Profiles for $S = 0.0$.

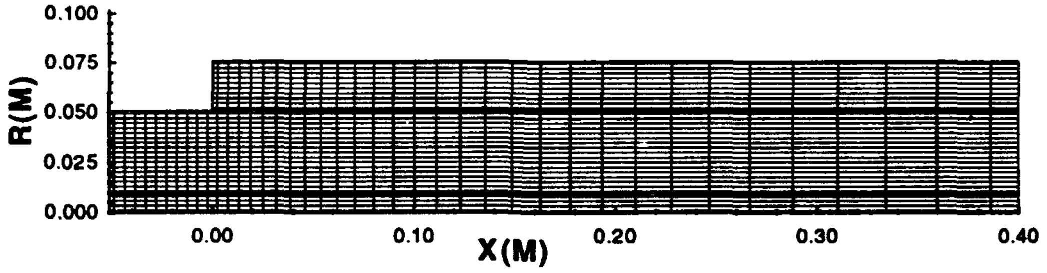

Figure 24: Computational Mesh for $S = 0.0$.

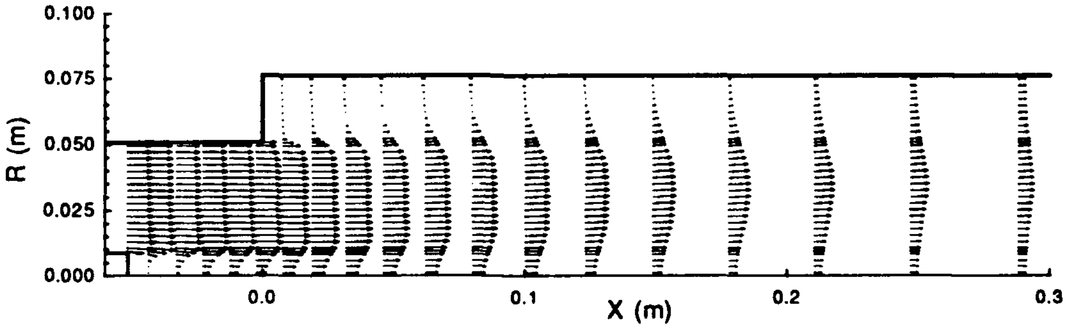

Figure 25: Calculated Axial-Radial Velocity Vectors (U-V) for $S = 0.3$.

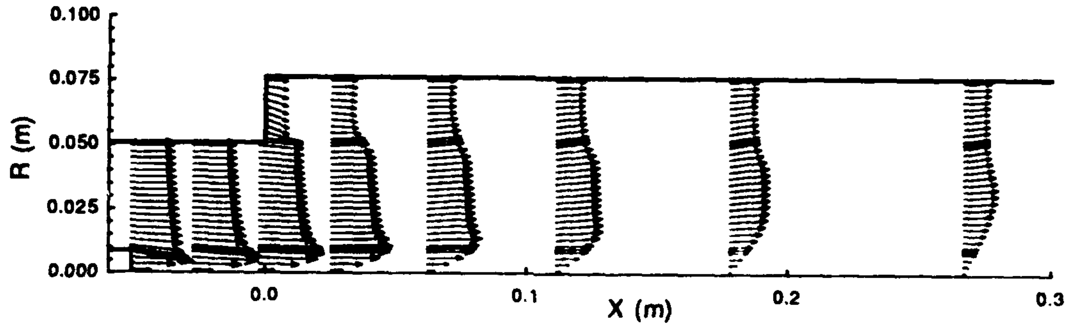

Figure 26: Calculated Swirl-Radial Velocity Vectors (W-V) for $S = 0.3$.

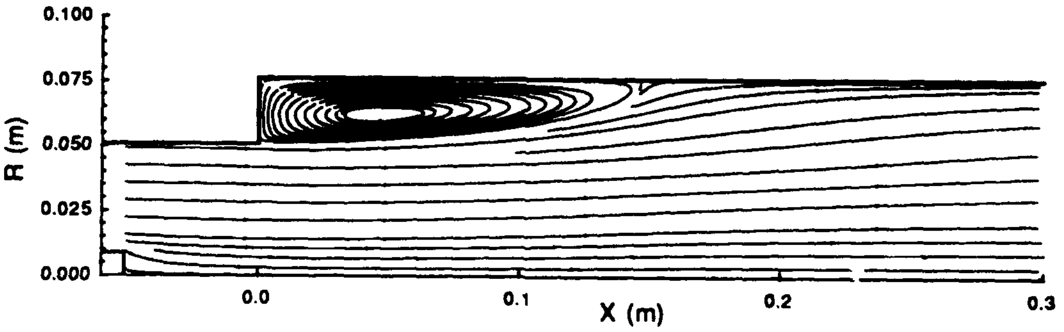

Figure 27: Calculated U-V Streamlines for $S = 0.3$.

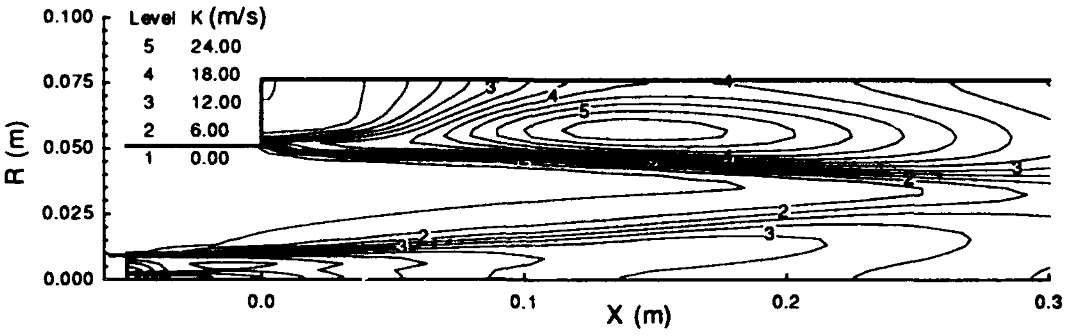

Figure 28: Calculated Turbulent Kinetic Energy Contours for $S = 0.3$.

also been observed in the other studies that were mentioned previously. The primary corner recirculating region length is calculated to be about 4.5 step heights which is about the experimental measured value of 4.3.

As in the experiment, no vortex breakdown is observed in the computed results. Even though the k–ϵ model did not produce the high turbulent kinetic energy due to the swirl near the centerline, the code again simulated the general features of the flow reasonably well.

The above calculations can be compared to the calculations of Vanka, et al.[42] shown in Figures 32, 33, and 34. Comparison of the two calculations is complicated because of the numerous differences in the computer codes and in their application. One major difference is that in the calculations by Vanka et al. the upstream boundary conditions were set at $x/H = 0.38$ using the measured conditions. Even with this "exact" inflow boundary condition the calculations are quite similar. Both calculations significantly underpredict the turbulence kinetic energy near the centerline downstream of $x/H = 1.0$. Computed turbulent kinetic energy profile shapes are substantially different from the measured profiles for both computer codes. In both calculations the axial velocity profiles had a bump at $r/H = 1.4$ at the downstream station ($x/H = 12$). In both calculations the swirl velocity in the corner recirculation region was under predicted, probably as a result of not resolving the secondary corner recirculation. Both calculations did very well in simulating the swirl velocity profiles upto $x/H = 8.0$. Downstream of this station the experimental profiles of swirl were much flatter than the either of the numerical simulations. The strong similarity between the two flow codes gives some credibility to the present calculations.

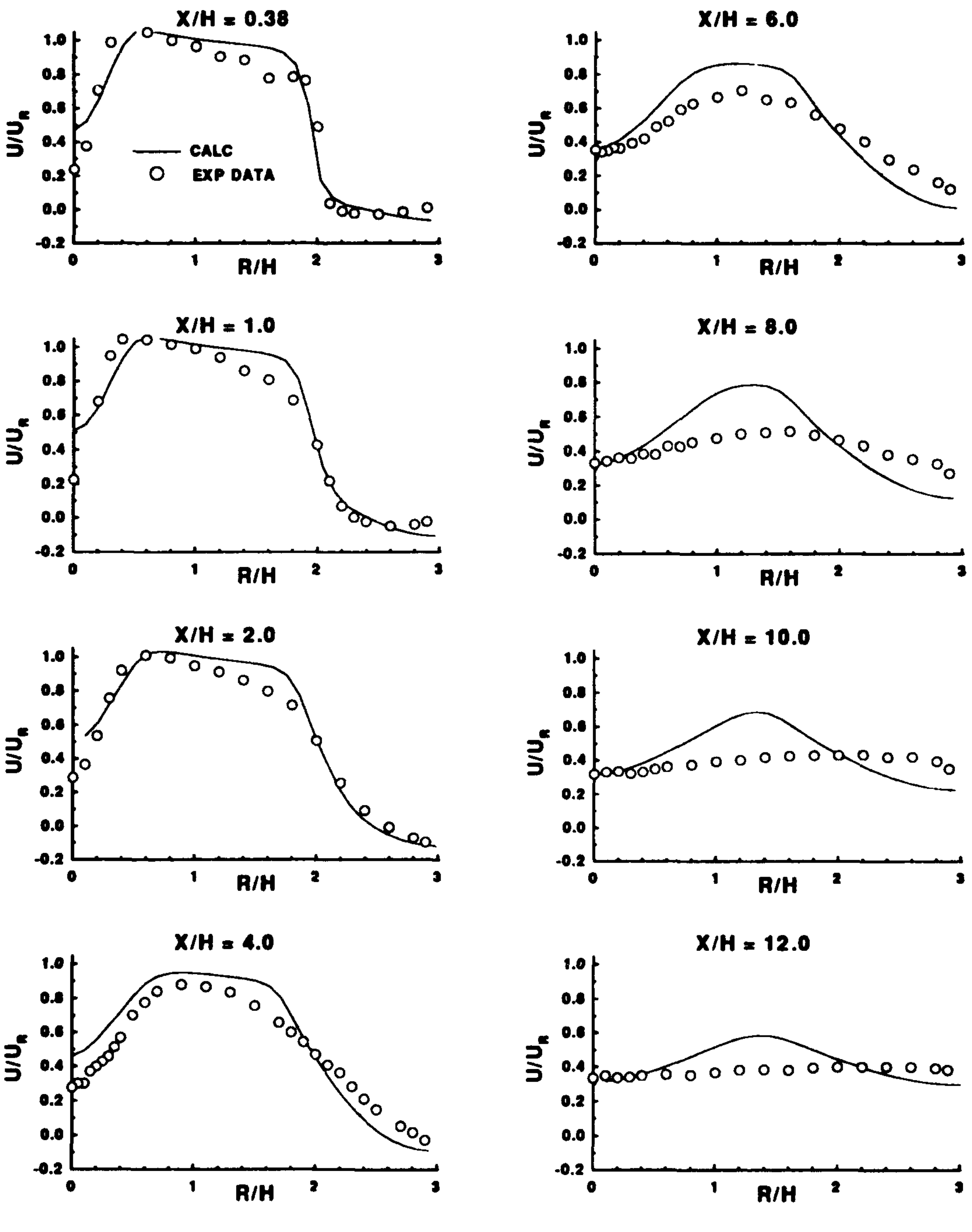

Figure 29: Axial Velocity Profiles for $S = 0.3$.

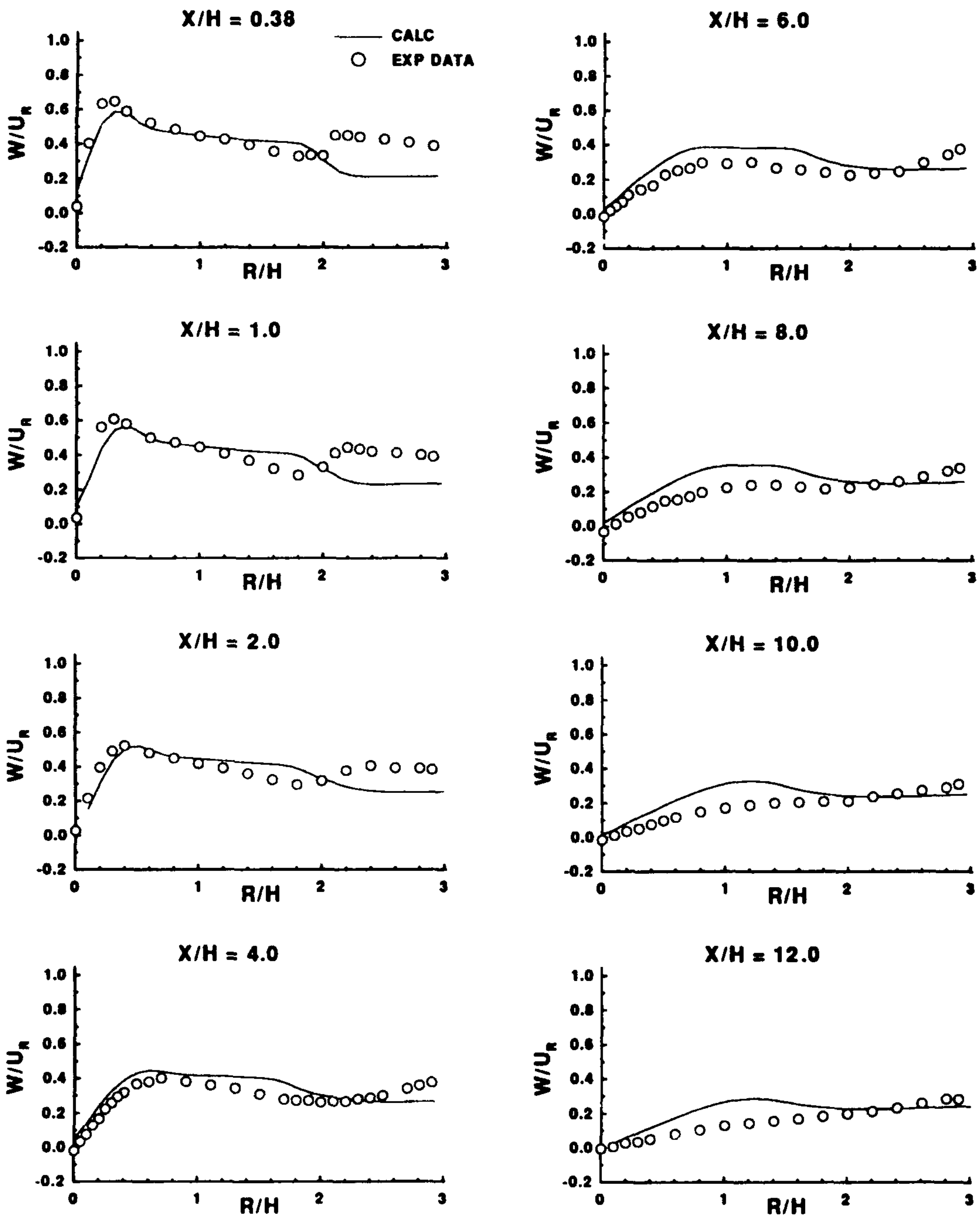

Figure 30: Swirl Velocity Profiles for $S = 0.3$.

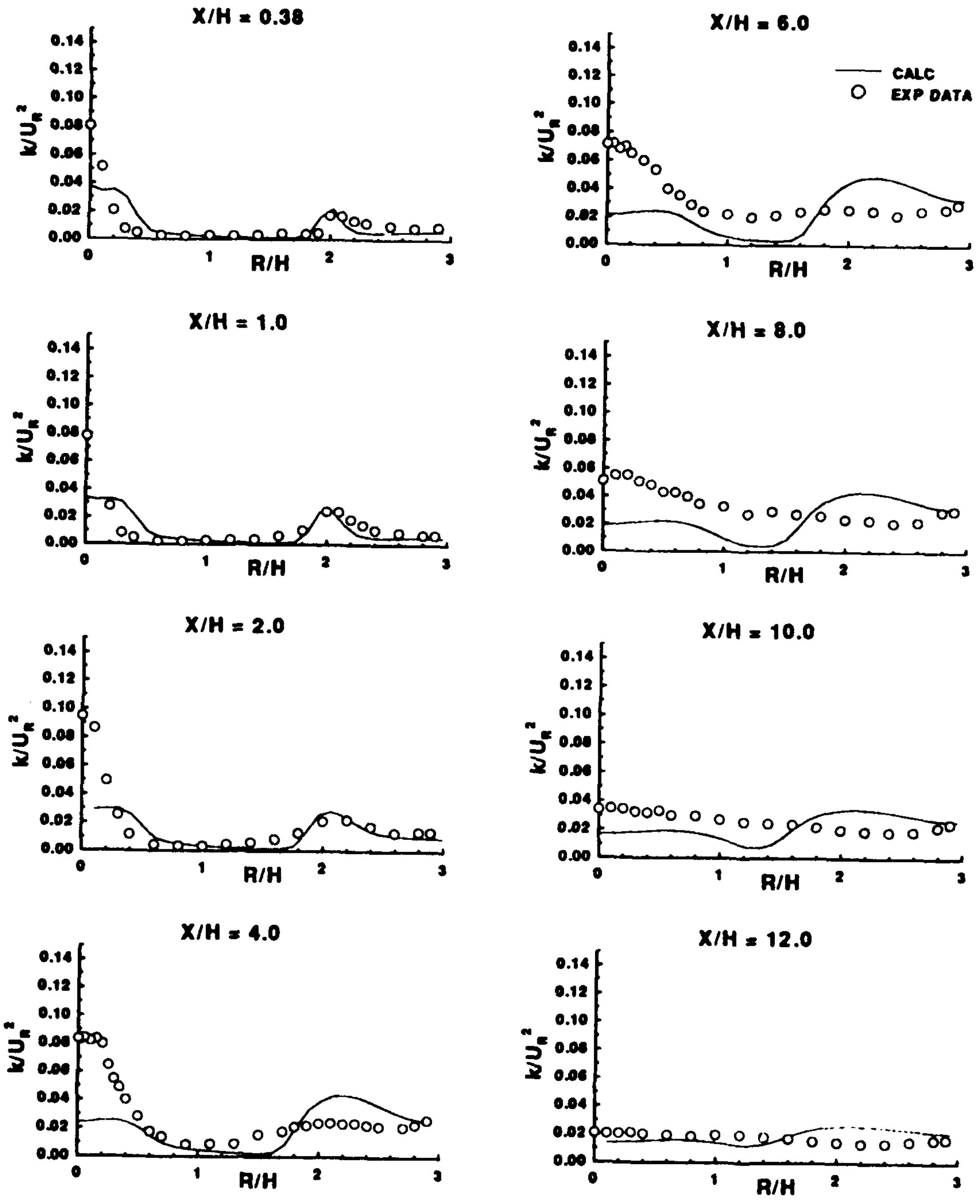

Figure 31: Turbulent Kinetic Energy Profiles $S = 0.3$.

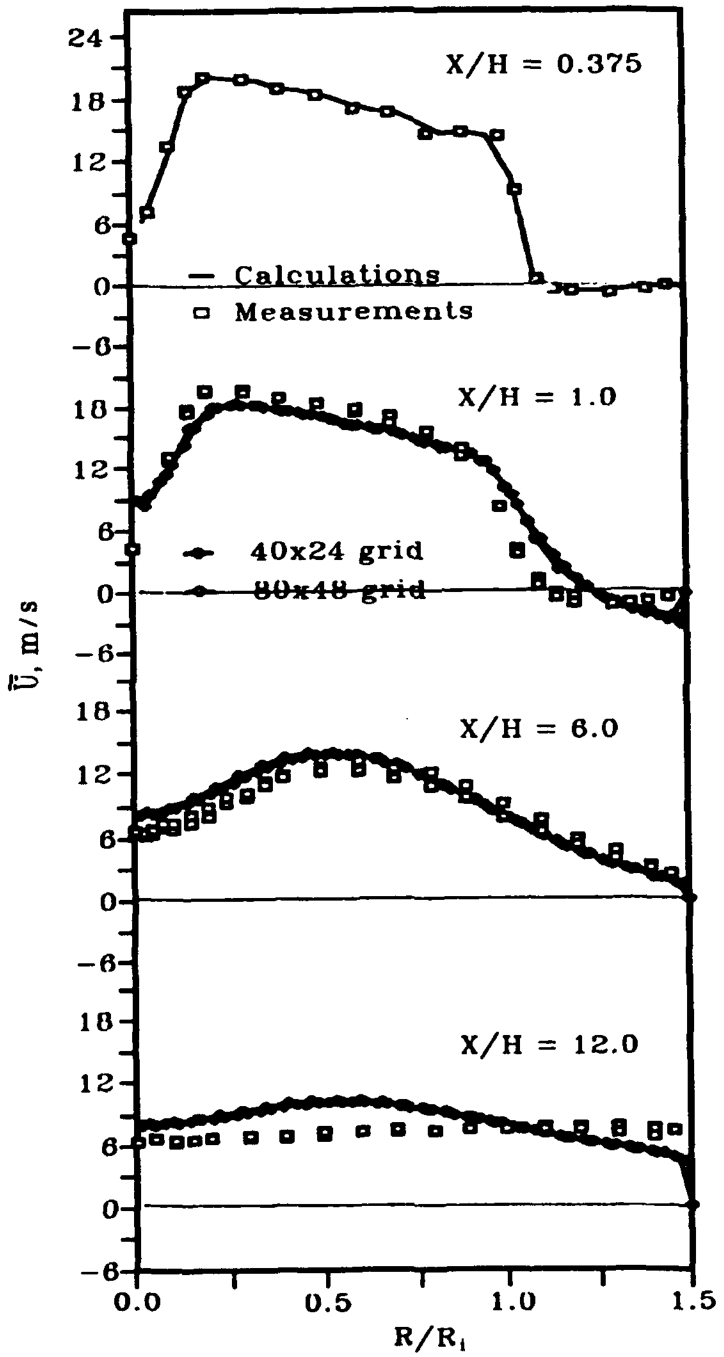

Figure 32: Axial Velocity Profile Comparisons $S = 0.3$ Calculated by Vanka, et al..

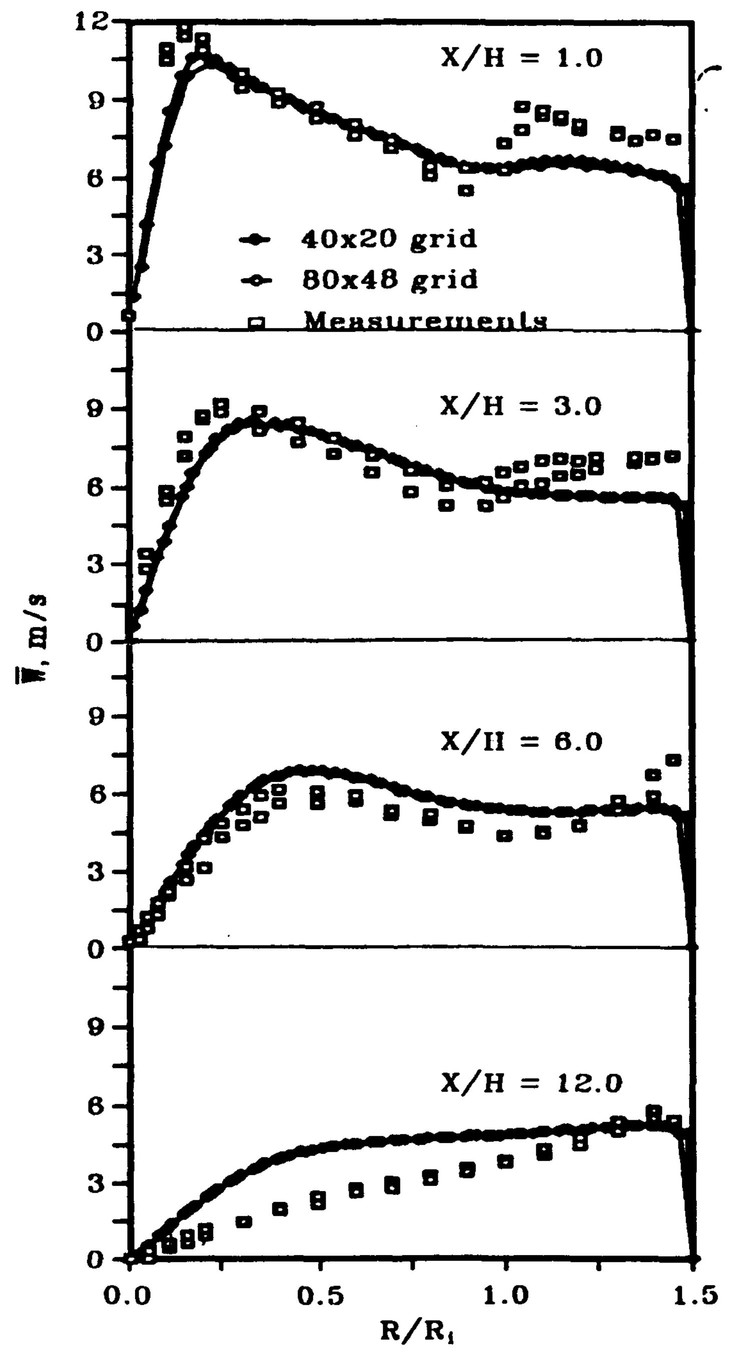

Figure 33: Swirl Velocity Profile Comparisons $S = 0.3$ Calculated by Vanka, et al..

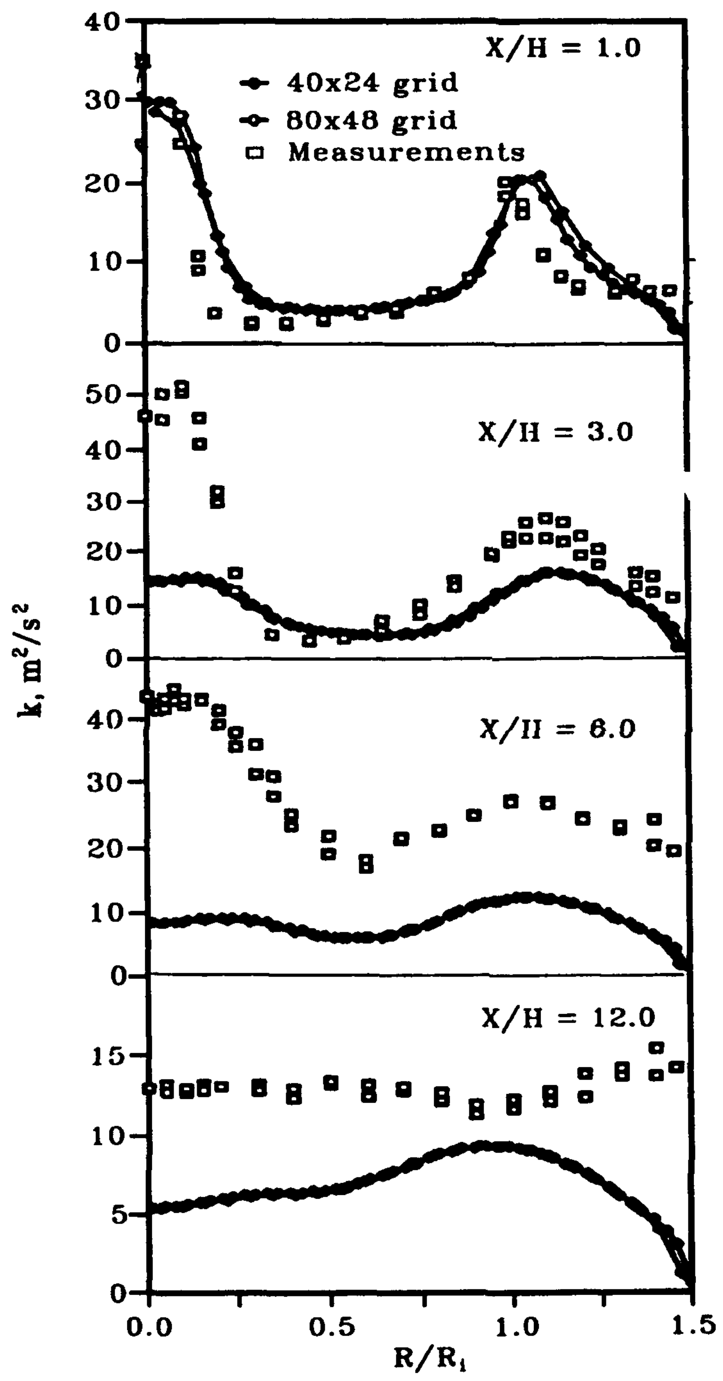

Figure 34: Turbulent Kinetic Energy Profiles $S = 0.3$ Calculated by Vanka, et al..

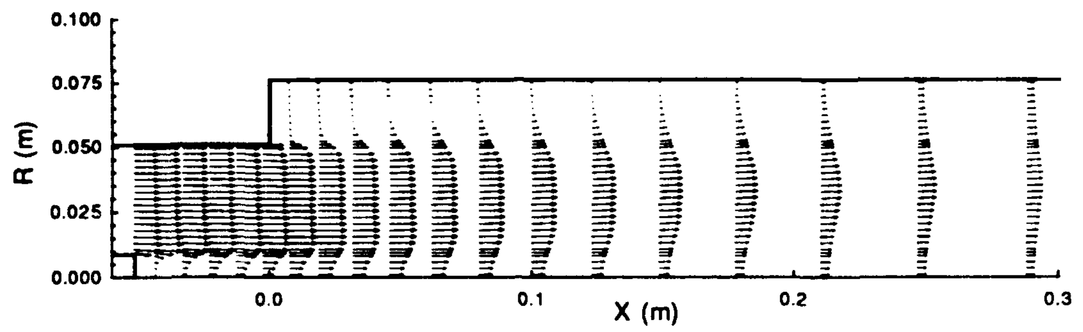

Figure 35: Calculated Axial-Radial Velocity Vectors (U-V) for $S = 0.5$.

6.2.3 High Swirl: $S = 0.5$

This case is the the same as the previous case ($S = 0.3$) except for that the swirl number is now set to 0.5. The computational domain and boundary conditions are the same except for the inflow velocity angles listed below.

$$\begin{aligned} u/|\vec{V}| &= 0.8084 \\ v/|\vec{V}| &= 0.0 \\ w/|\vec{V}| &= 0.589 \end{aligned}$$

Results for this calculation are shown in Figure 35 through 38. The length of the corner recirculation zone is about 3.5 step heights which is approximately equal to the experimental value of 3.2. A CTRZ is apparent in the exerimental data for this case, but did not appear in the calculation. However, the velocity profiles indicate that a CTRZ is close to forming. A small recirculation bubble is shown just downstream of the hub. Figures 39 and 40 show profiles of measured and computed mean axial velocities (u) and swirl velocities (w) at eight axial locations. Like the $S = 0.3$ case, the k–ϵ model under predicts the turbulent kinetic energy near the center line of the dump combustor (Figure 41).

6.2.4 Calculations with Modified k–ϵ Constants for $S = 0.0$.

Figures 42 through 46 show the calculated results for the no-swirl case with slightly modified k–ϵ constants. The modified constants are: $C_\mu = 0.07$ and $C_{\epsilon 2} = 1.887$.

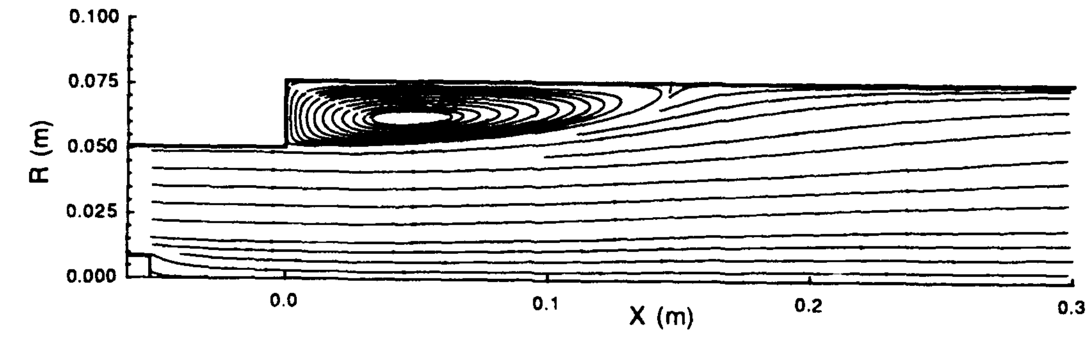

Figure 36: Calculated U-V Streamlines for $S = 0.5$.

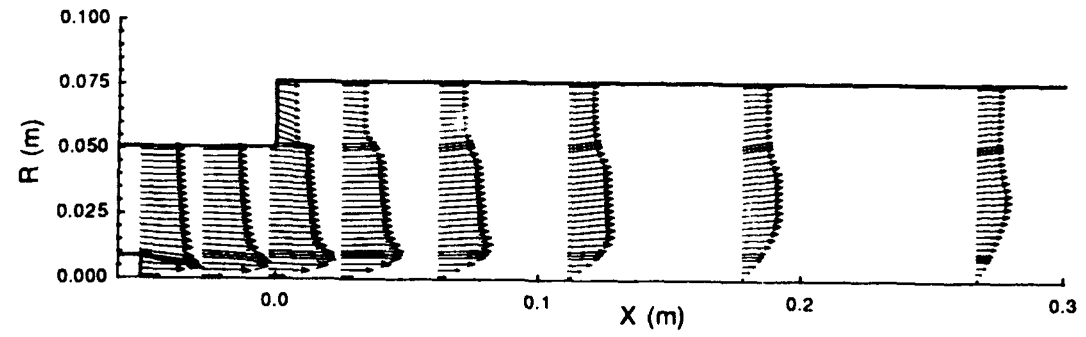

Figure 37: Calculated Swirl-Radial Velocity Vectors (W-V) for $S = 0.5$.

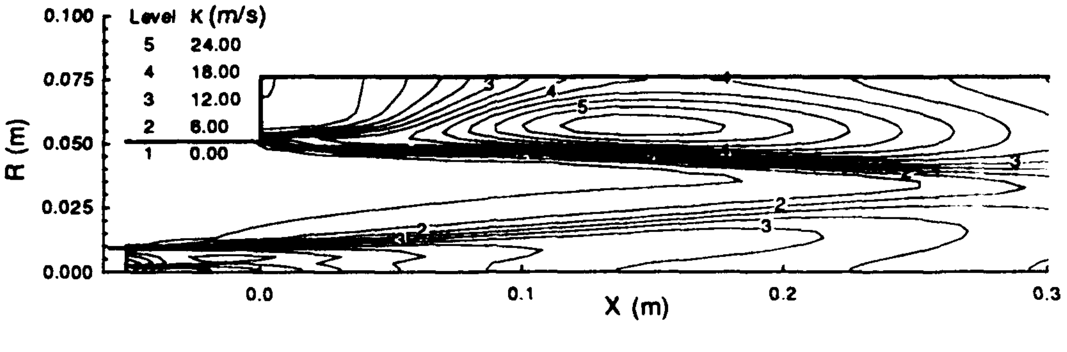

Figure 38: Calculated Turbulent Kinetic Energy Contours for $S = 0.5$.

Lowering the values of these constants result in lower values of the eddy viscosity. The calculated streamlines shown in Figure 43 show the primary corner recirculating region is predicted very well, about 7.5 step heights compared to 8 step heights observed experimetally. This calculation shows that the k–ϵ turbulence model constants have a significant influence on the recirculating flow. Small changes in the turbulence model result in significant changes in the flow field.

6.2.5 Calculations with Modified k–ϵ Constants for $S = 0.5$.

The $S = 0.5$ case computed with the standard k–ϵ constants shows the largest discrepancy with the experimental data, since the computed results show no vortex breakdown. This case was recalculated with modified turbulence model constants: $C_\mu = 0.07$ and $C_{\epsilon 2} = 1.90$. The calculated results are shown in Figures 47 through 50. The length of the CRZ region agrees very well with the experimental results. The extent, location, and size of the CRTZ is also very well simulated. It is important to note that the CTRZ extends upsteam of the dump plane. Comparison of calculated velocity profiles (axial and swirl) and turbulent kinetic energy profiles with measured values in Figures 51 through 53 show that the calculations with the modified turbulence model constants agree favorably with experimental results.

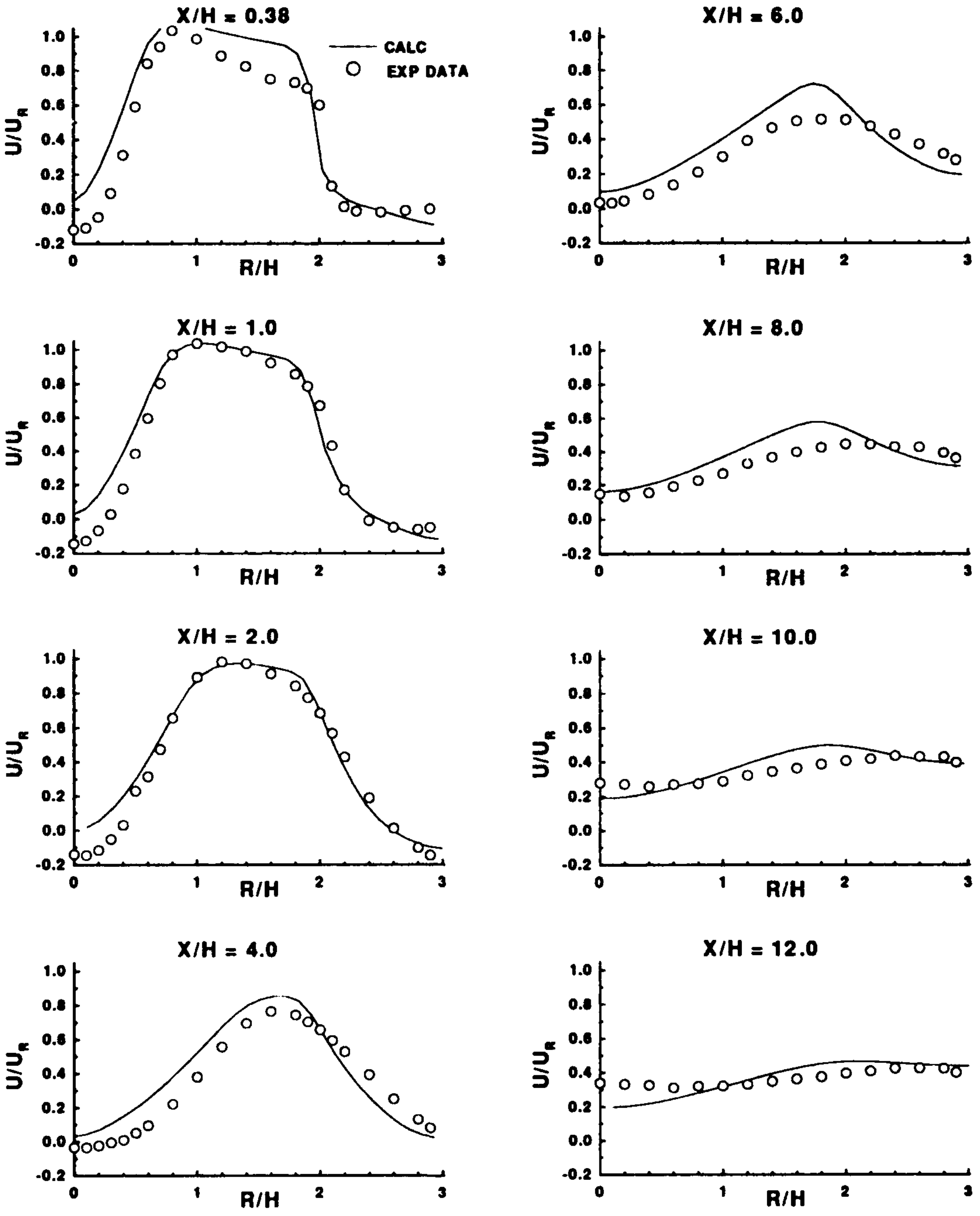

Figure 39: Axial Velocity Profiles for $S = 0.5$.

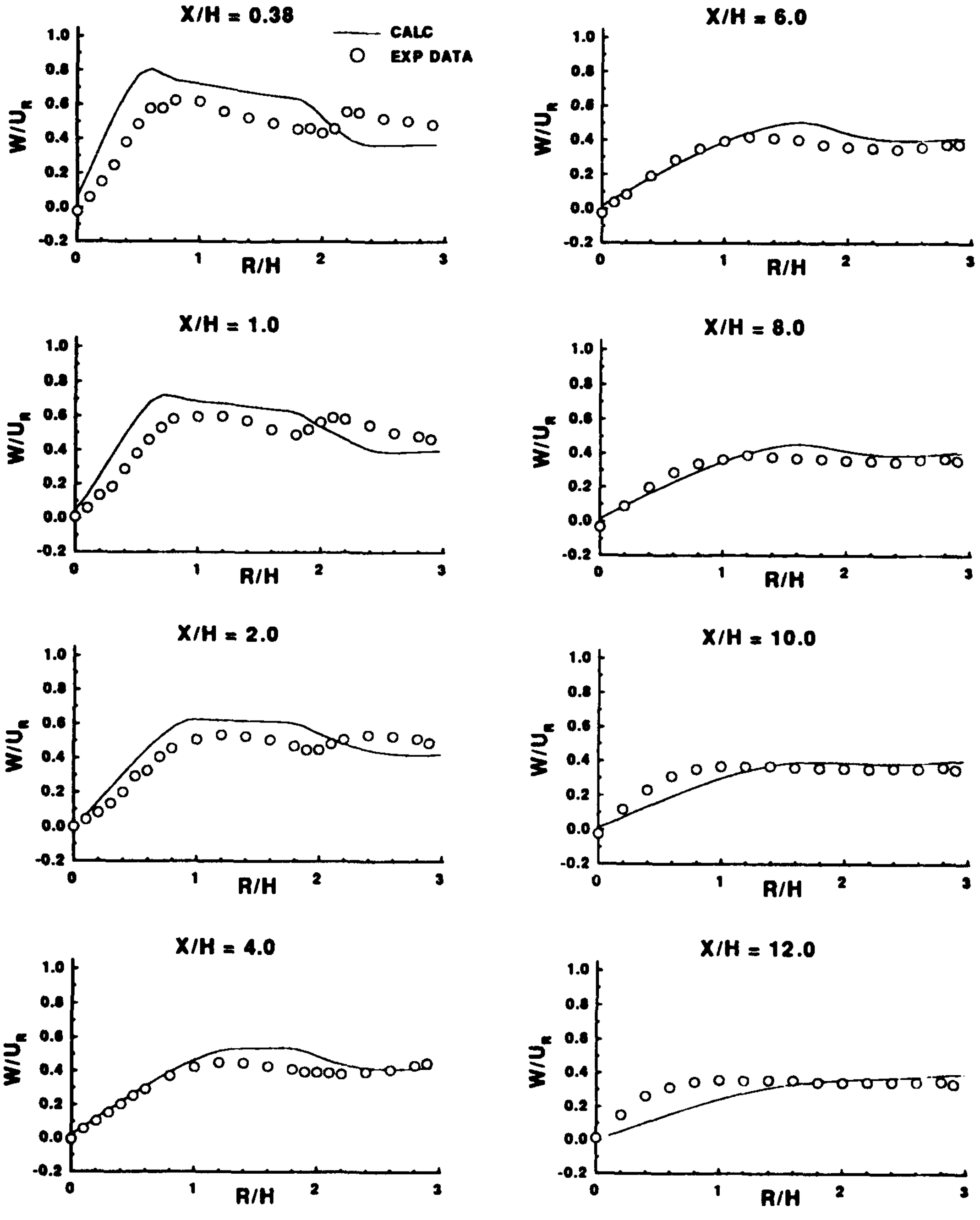

Figure 40: Swirl Velocity Profiles for $S = 0.5$.

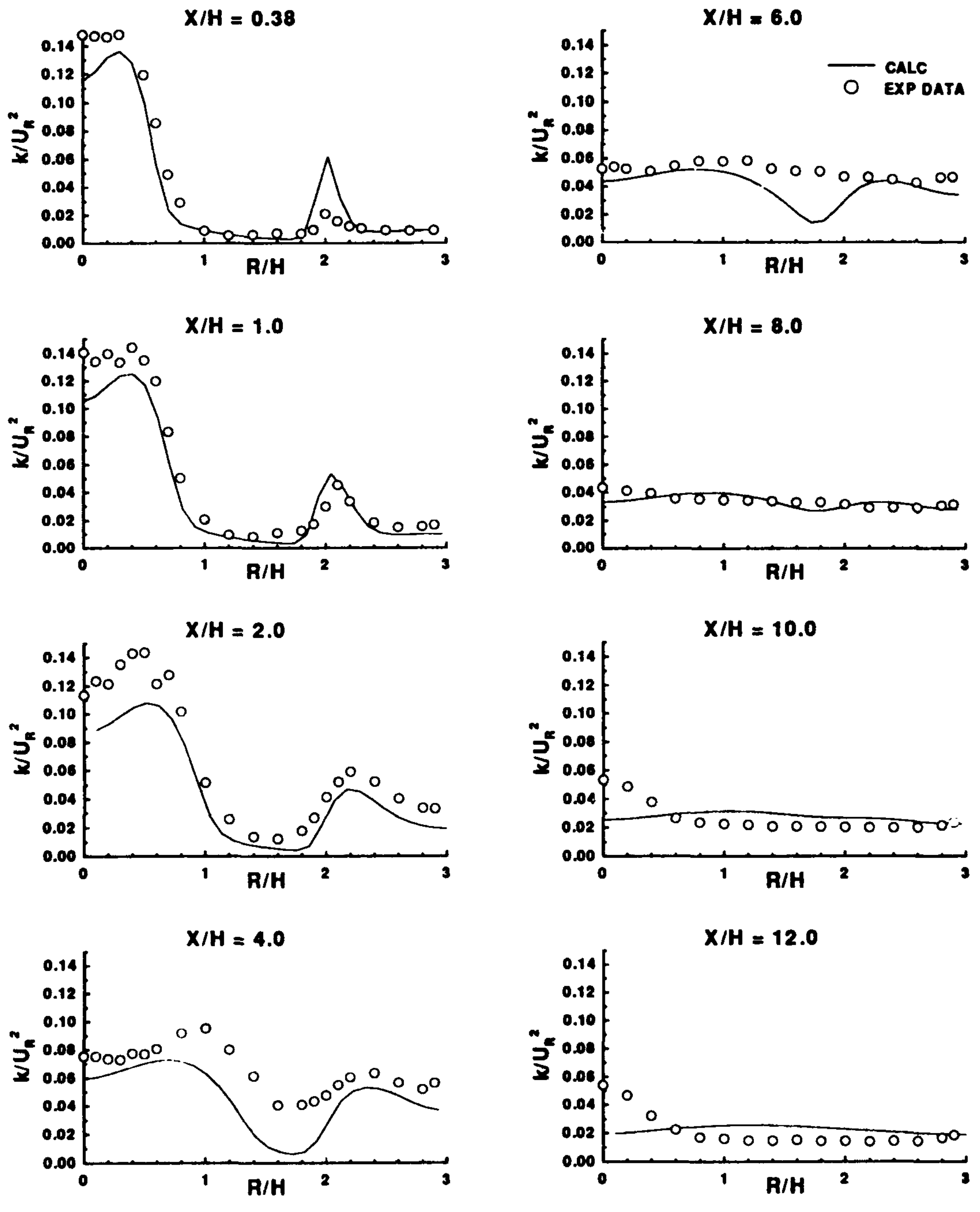

Figure 41: Turbulent Kinetic Energy Profiles for $S = 0.5$.

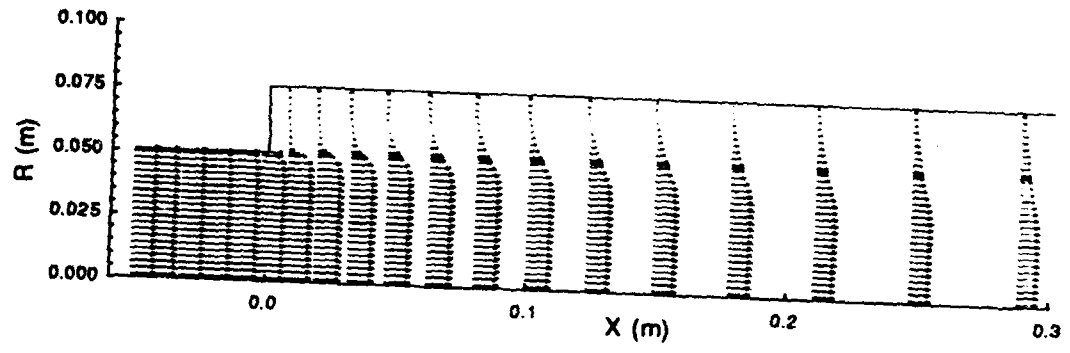

Figure 42: Calculated Axial-Radial Velocity Vectors (U-V) with $C_\mu = 0.07$ and $C_{\epsilon 2} = 1.887$ for $S = 0.0$.

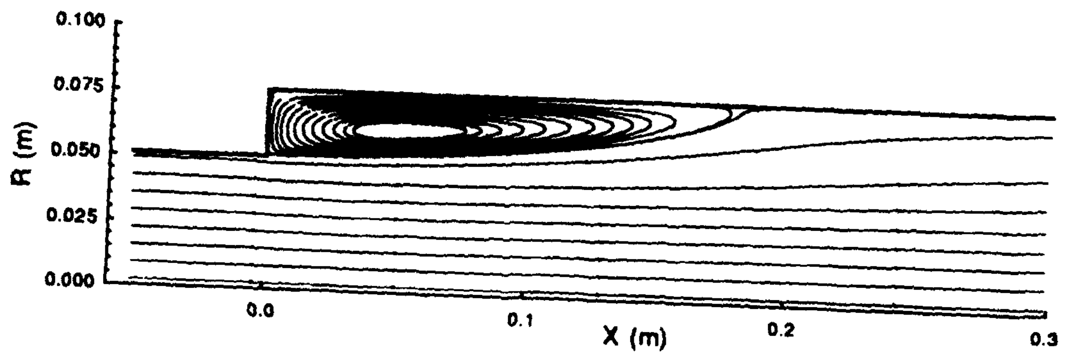

Figure 43: Calculated U-V Streamlines with $C_\mu = 0.07$ and $C_{\epsilon 2} = 1.887$ for $S = 0.0$.

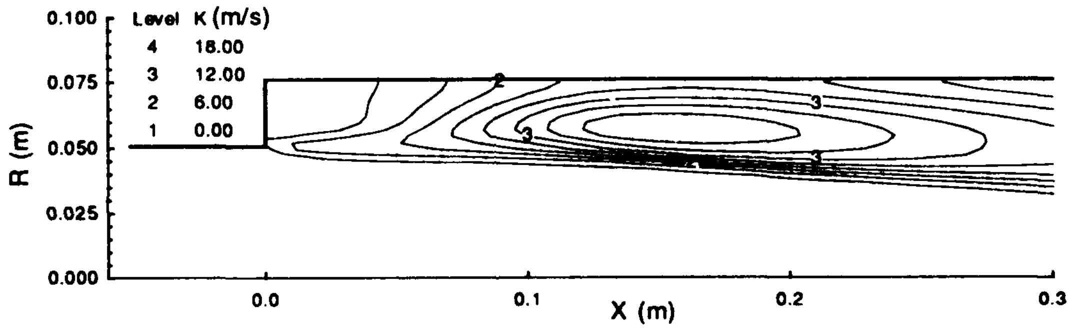

Figure 44: Calculated Turbulent Kinetic Energy Contours with $C_\mu = 0.07$ and $C_{\epsilon 2} = 1.887$ for $S = 0.0$.

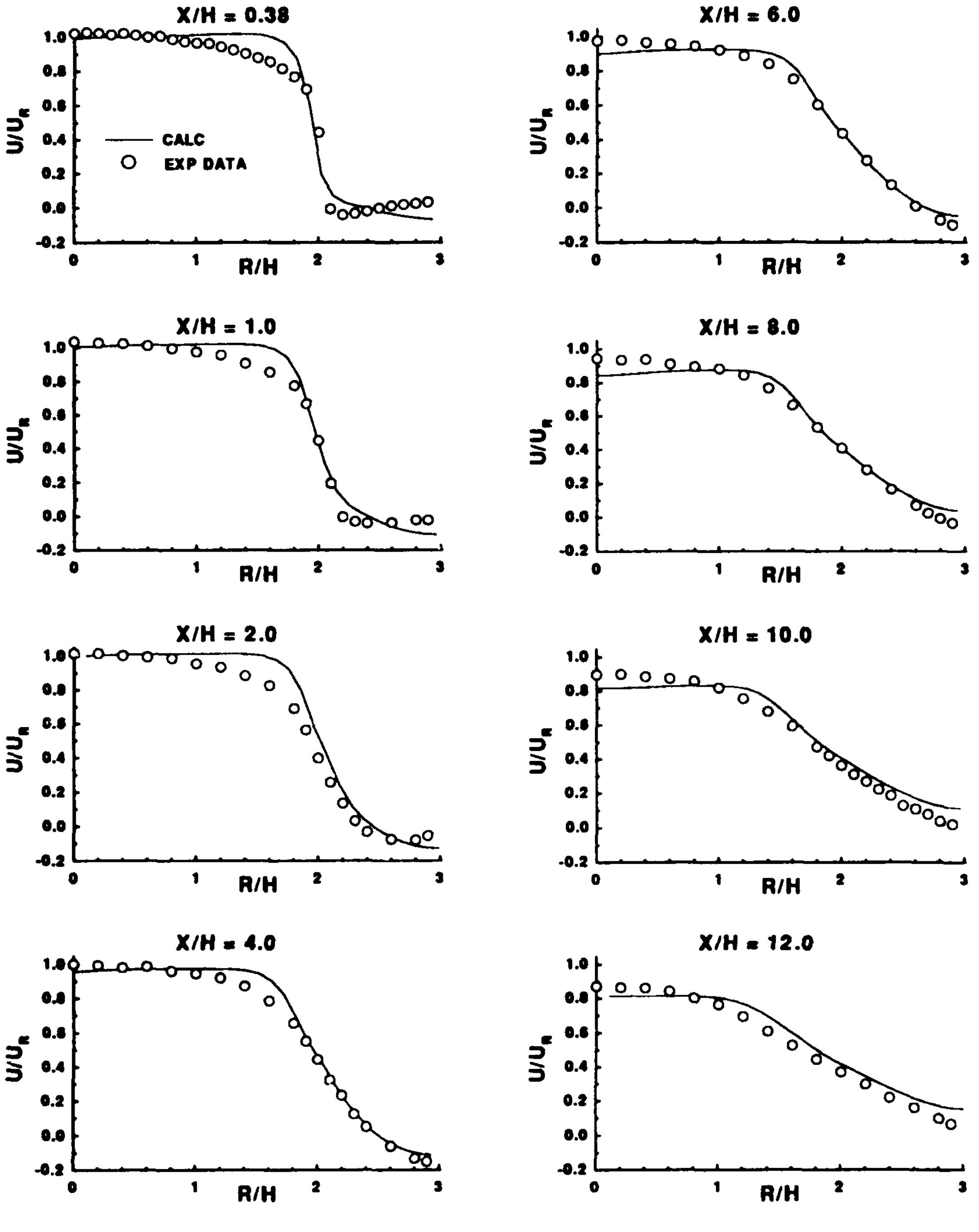

Figure 45: Axial Velocity Profiles. Calculations with $C_\mu = 0.07$ and $C_{\epsilon 2} = 1.887$ for $S = 0.0$.

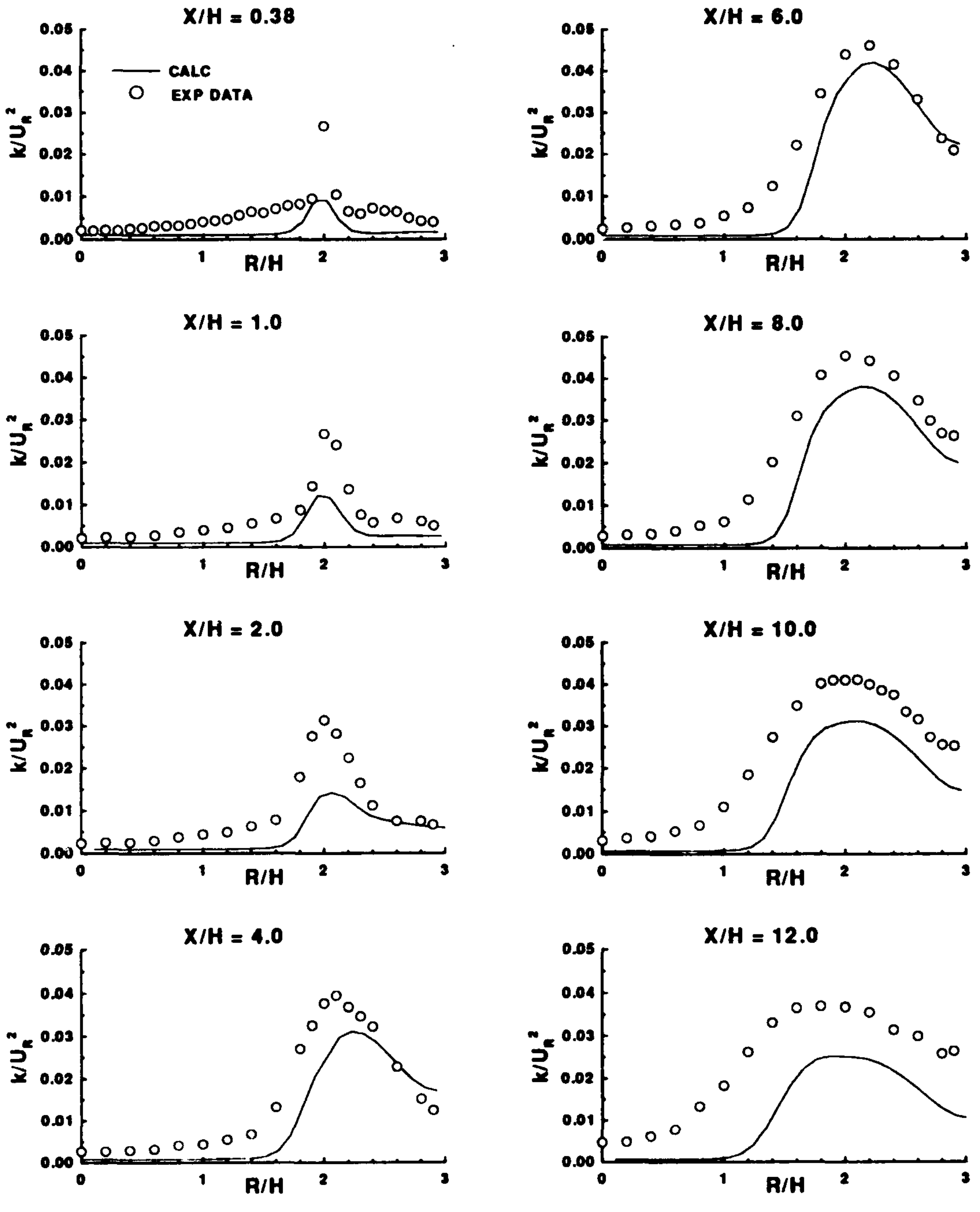

Figure 46: Turbulent Kinetic Energy Profiles. Calculations with $C_\mu = 0.07$ and $C_{\epsilon 2} = 1.887$ for $S = 0.0$.

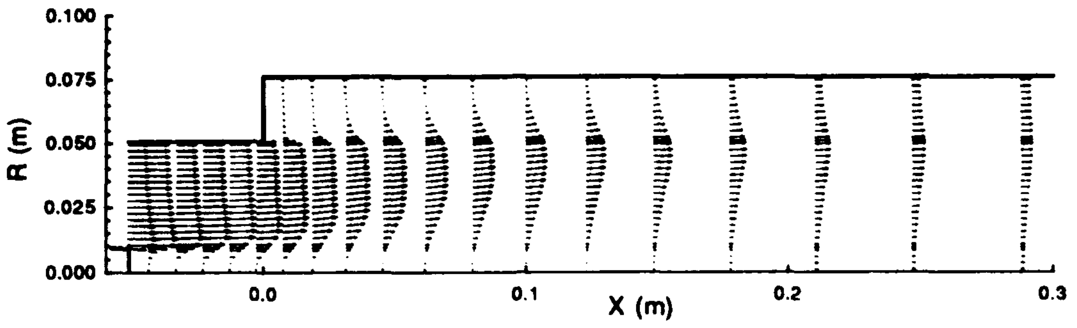

Figure 47: Calculated Axial-Radial Velocity Vectors (U-V) with $C_\mu = 0.07$ and $C_{\epsilon 2} = 1.90$ for $S = 0.5$.

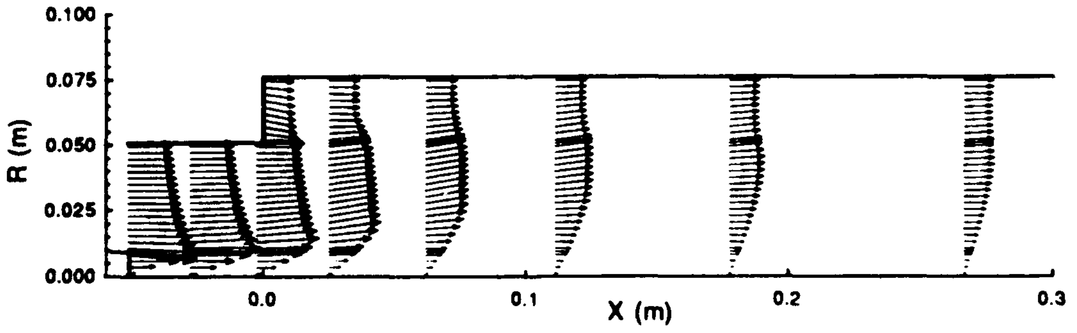

Figure 48: Calculated Swirl-Radial Velocity Vectors (W-V) with $C_\mu = 0.07$ and $C_{\epsilon 2} = 1.90$ for $S = 0.5$.

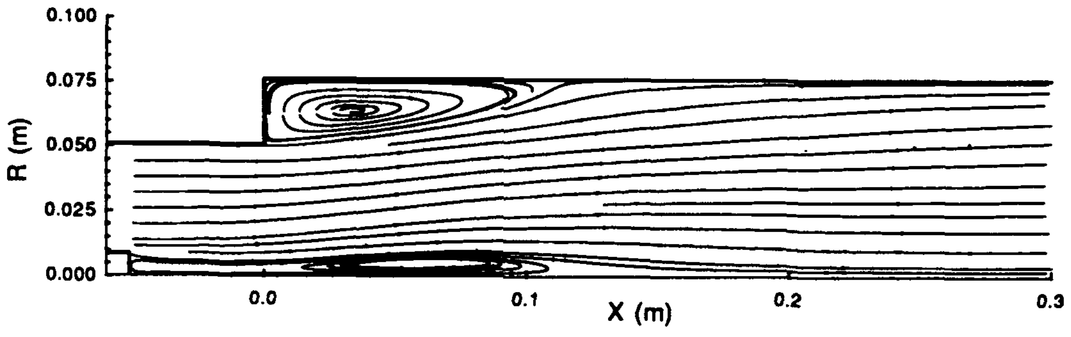

Figure 49: Calculated U-V Streamlines with $C_\mu = 0.07$ and $C_{\epsilon 2} = 1.90$ for $S = 0.5$.

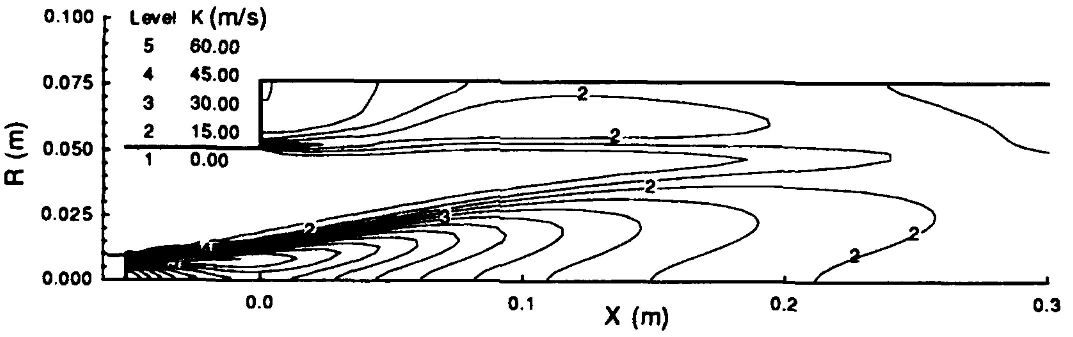

Figure 50: Calculated Turbulent Kinetic Energy Contours with $C_\mu = 0.07$ and $C_{\epsilon 2} = 1.90$ for $S = 0.5$.

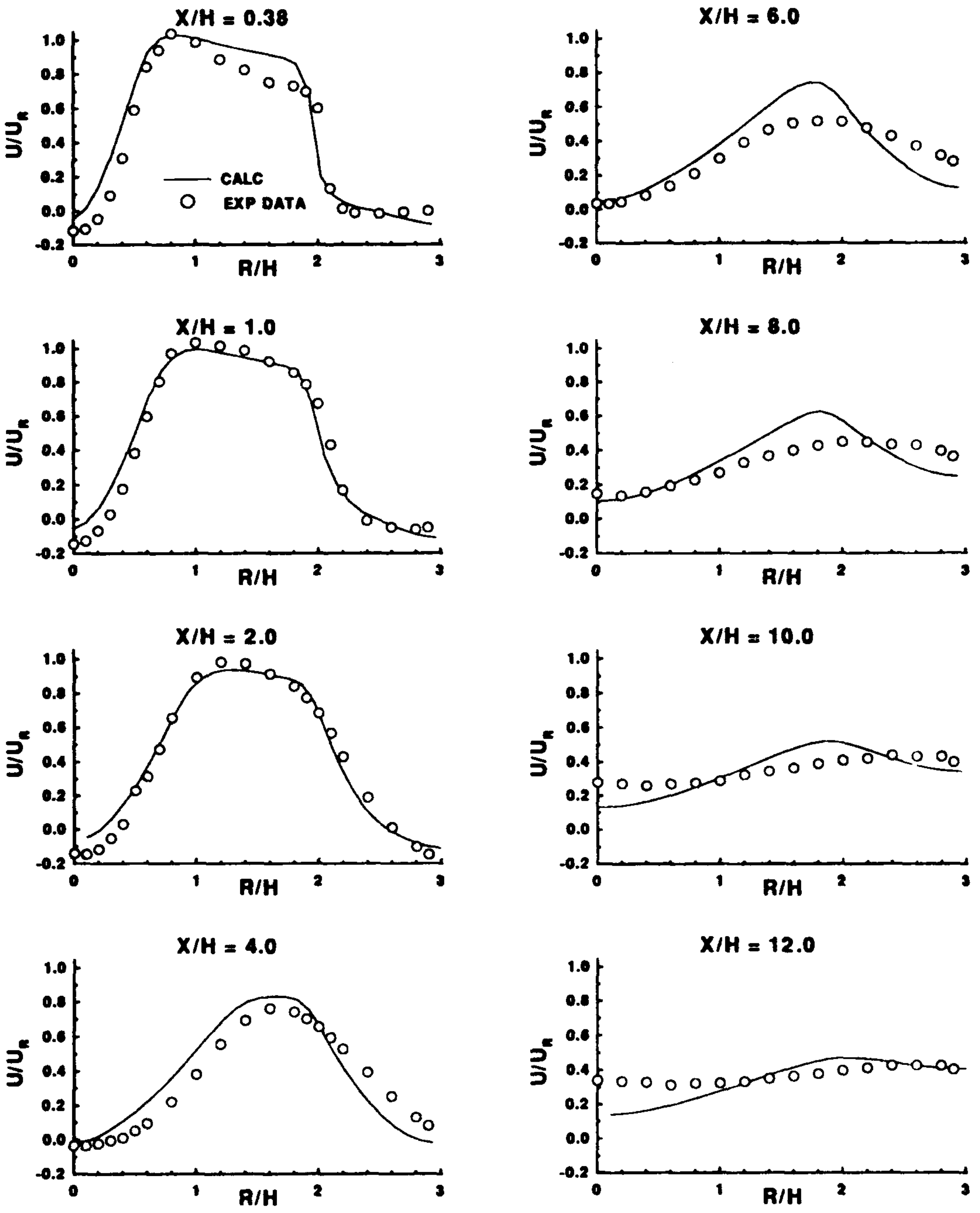

Figure 51: Axial Velocity Profiles. Calculations with $C_\mu = 0.07$ and $C_{\epsilon 2} = 1.90$ for $S = 0.5$.

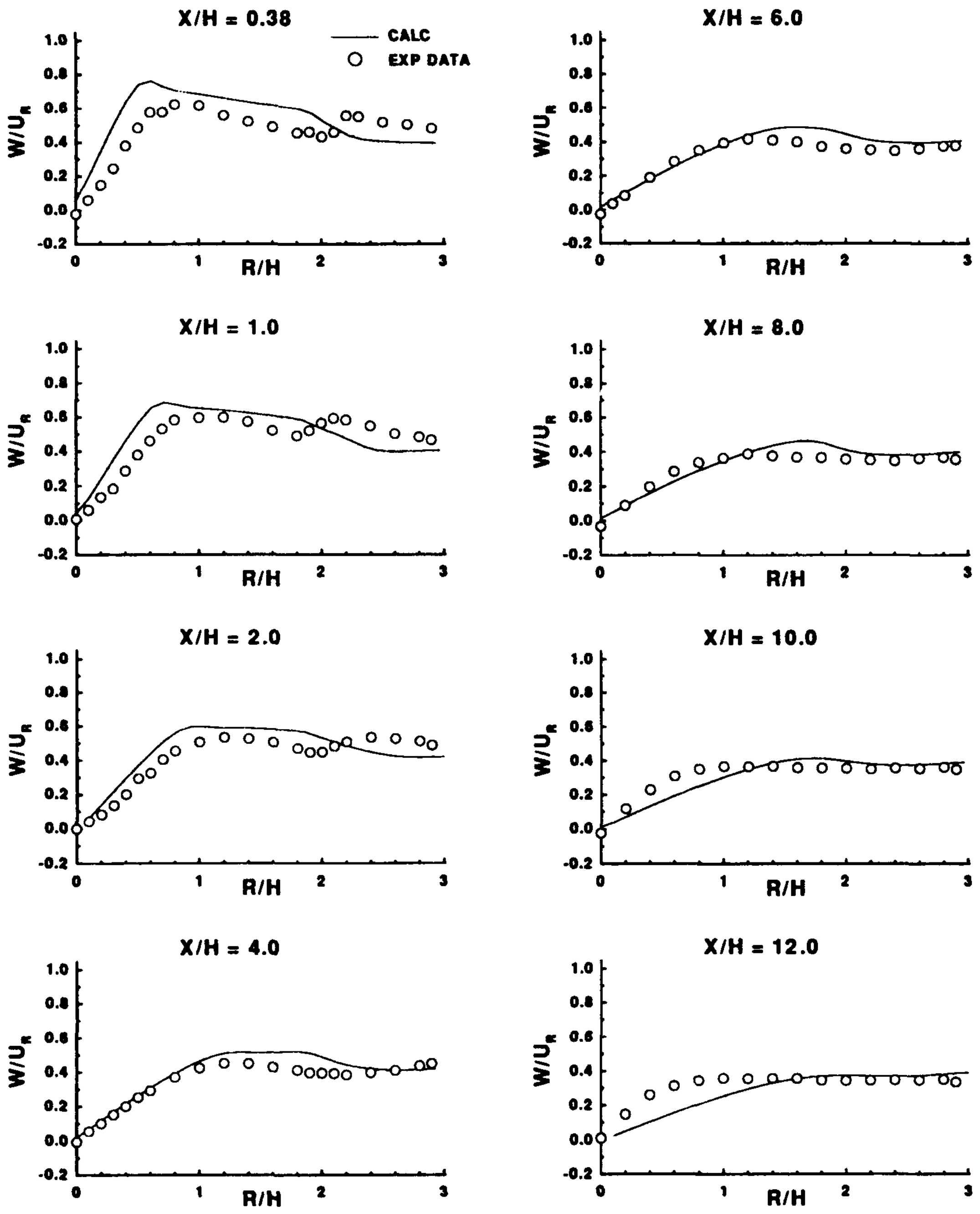

Figure 52: Swirl Velocity Profiles. Calculations with $C_\mu = 0.07$ and $C_{\epsilon 2} = 1.90$ for $S = 0.5$.

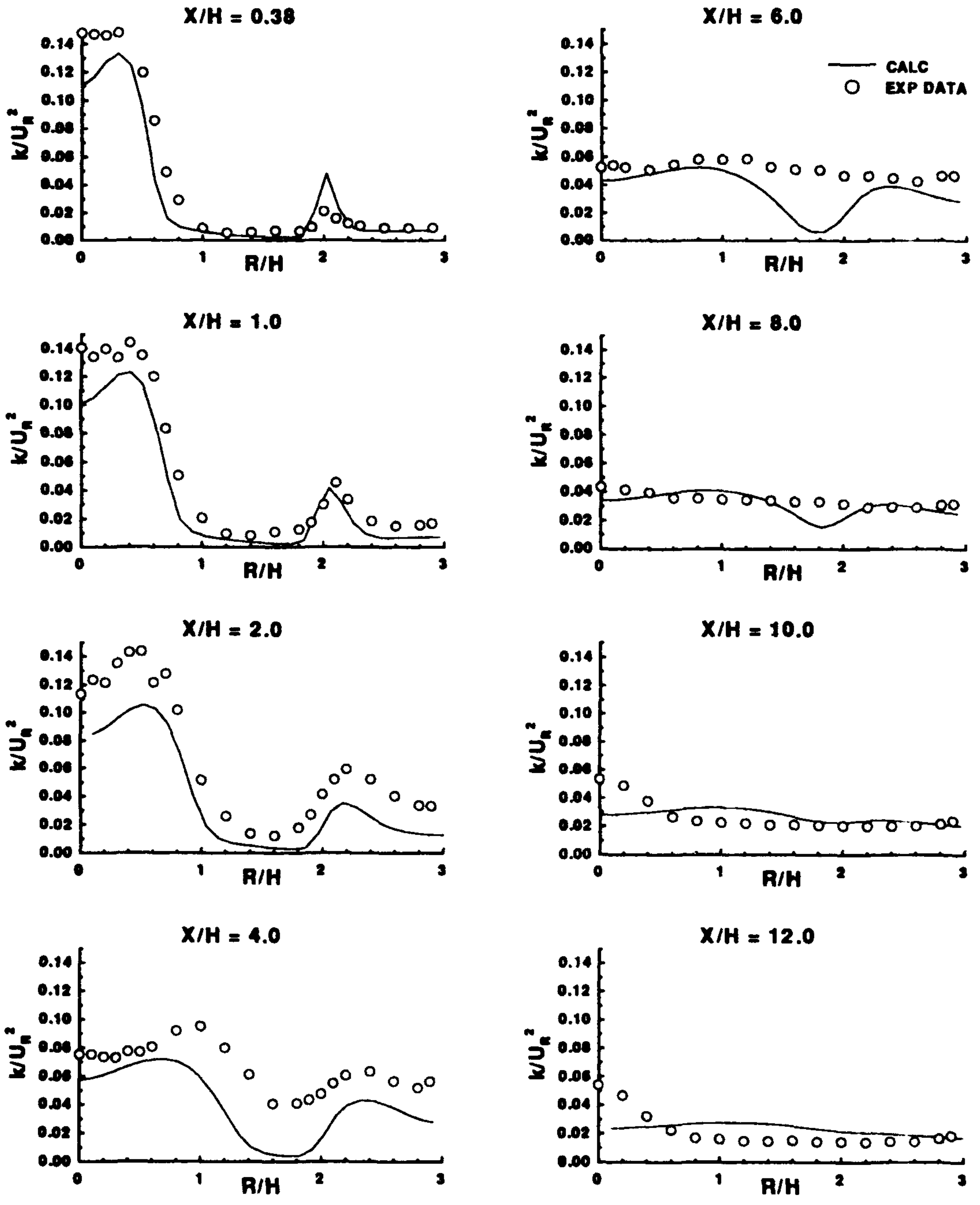

Figure 53: Turbulent Kinetic Energy Profiles. Calculations with $C_\mu = 0.07$ and $C_{\epsilon 2} = 1.90$ for $S = 0.5$.

6.3 Coupled Dump Combustor/CD-Nozzle Calculations

To demonstrate the capability of DUMPSTER to calculate the flow in a closely coupled dump combustor and CD nozzle and to study the influence of the CD nozzle on the combustor flow field, four calculations were performed on the configurations shown in Figure 54. The only differences between the calculations were the lengths of the combustor region. The combustor/nozzle geometry is similar to the cases above in that the inlet and step height are the same. Three-zone meshes that were used are shown in Figure 54. The dimensions of each mesh zone is shown in the table below. The combustor length (L) is the distance in meters from the dump plane to the beginning of the nozzle.

Case No:	1	2	3	4
Combustor Length (L)	$0.1m$	$0.2m$	$0.3m$	$0.4m$
Zone 3	50x10	65x10	80x10	85x10
Zone 2	60x17	75x17	90x17	95x17
Zone 1	60x4	75x4	90x4	95x4

The *inflow boundary conditions* are listed below for an inlet swirl number of 0.5.

$$\begin{aligned} P_T &= 101500.0\,Pa \\ T_T &= 300.6\,^{\circ}K \\ u/|\vec{V}| &= 0.8084 \\ v/|\vec{V}| &= 0.0 \\ w/|\vec{V}| &= 0.589 \\ k &= 0.5\,m^2/s^2 \\ \epsilon &= 23.0\,m^2/s^3 \end{aligned}$$

The outflow boundary condition is supersonic outflow. The k-ϵ-turbulence model constants were set to the modified values of the last case.

The results of the calculations are presented in Figures 55 through 59. Comparison of these four cases illustrate the effects of the nozzle position upon the combustor flow field. As the combustor region is shortened the nozzle effects the size and extent of the CTRZ and the CRZ. As the combustor length is decreased the CD nozzle has a stronger influence upon the combustor flowfield. The CRZ increases in length as the combustor length is shortened, and the CTRZ reduces in length and finally vanishes.

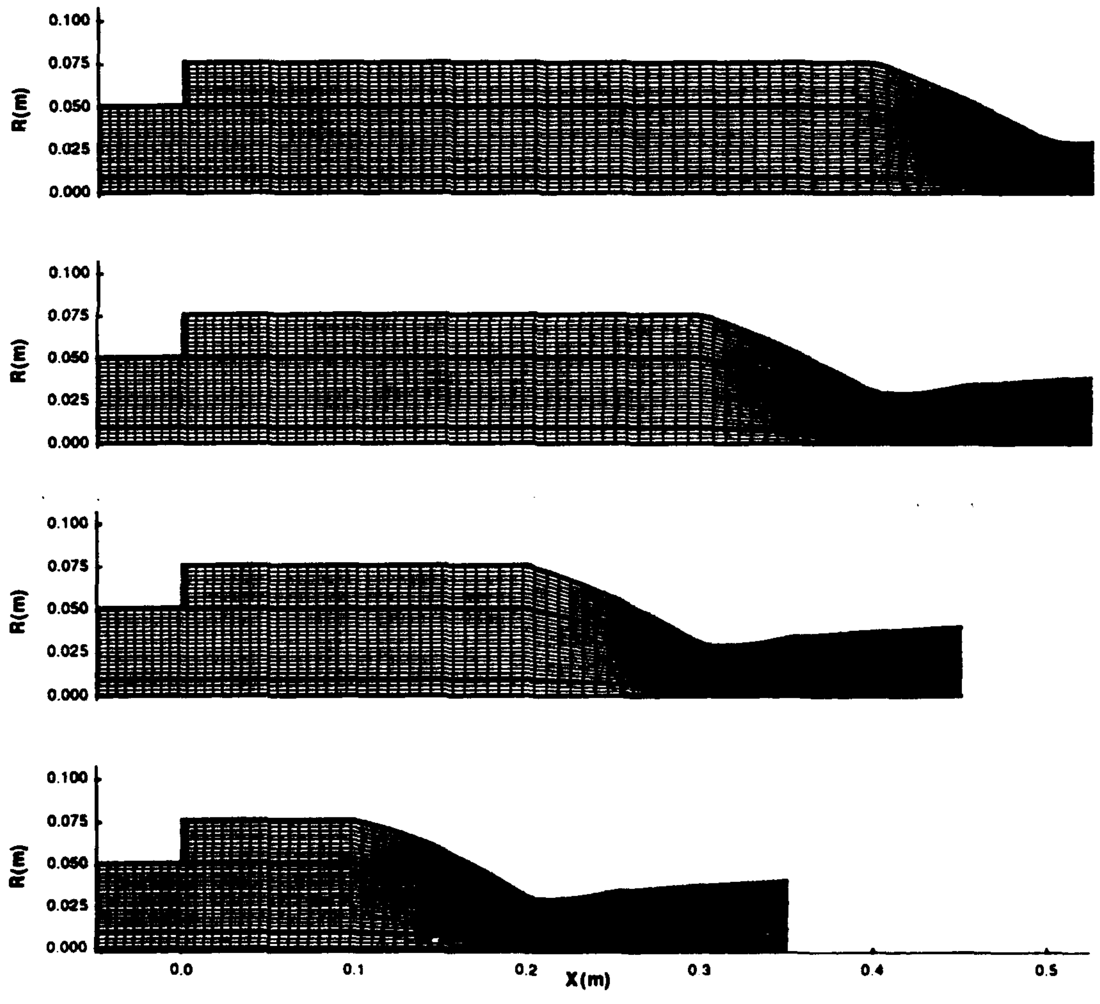

Figure 54: Computational Meshes for the Dump Combustor with Coupled CD Nozzle.

The CTRZ changes continuously with no sudden change in shape as the nozzle is moved closer to the dump plane.

Figures 60 to 62 show the calculated profiles of axial and swirl velocities and the turbulent kinetic energy at four stations along the axis for all four combustor lengths (L). The calculations were run 3000 steps with the Steger-Warming flux function to "rough out" the initial conditions, converging the solution three orders of magnitude. Then each case was restarted using the Harten-Yee flux function and run 6000 steps converging another three orders of magnitude. The overall run time for these cases was approximatly 8 CPU hours on an IBM RS/6000 model 320 workstation.

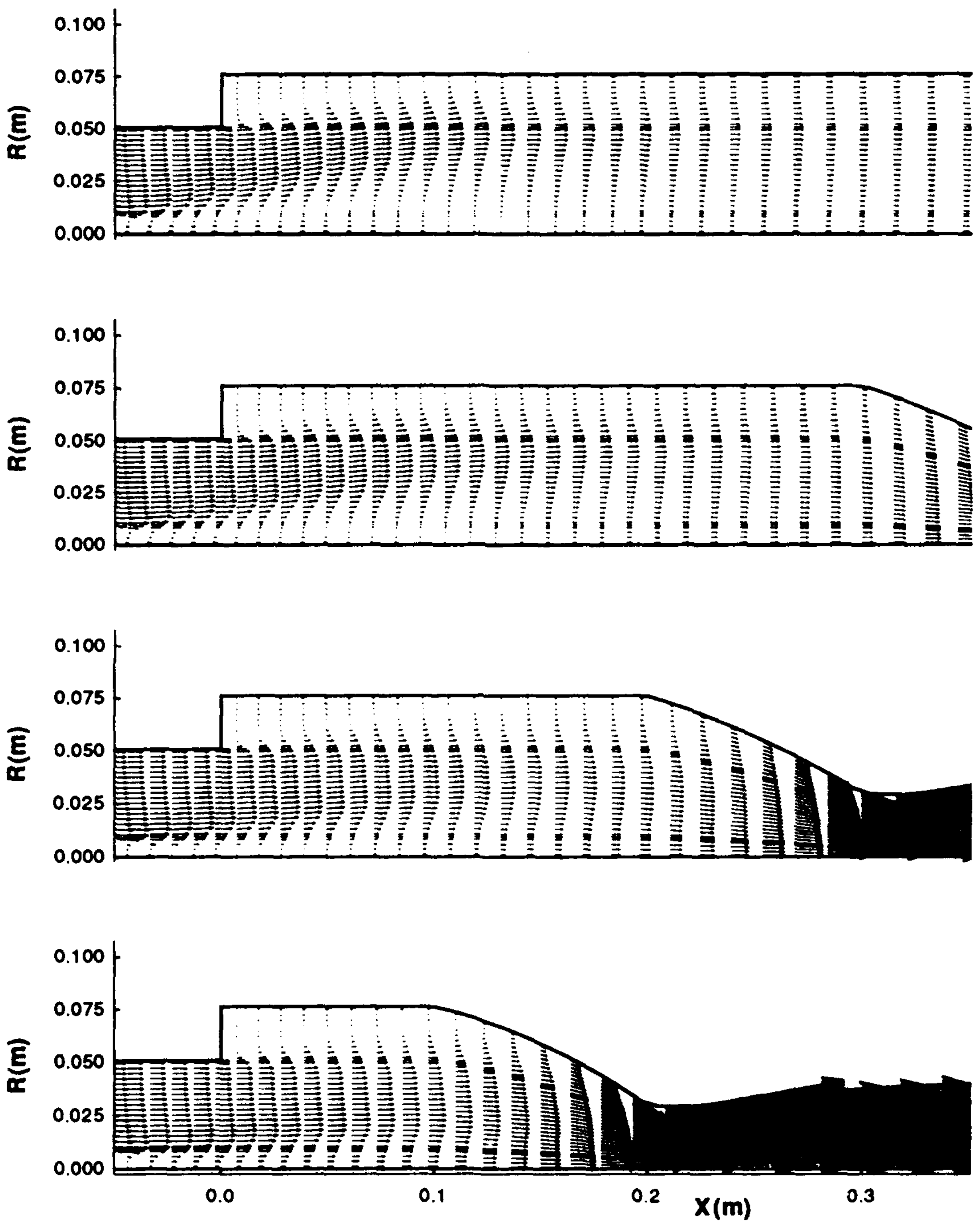

Figure 55: Calculated Axial-Radial Velocity Vectors (U-V) for Dump Combustor and Closely Coupled CD Nozzle.

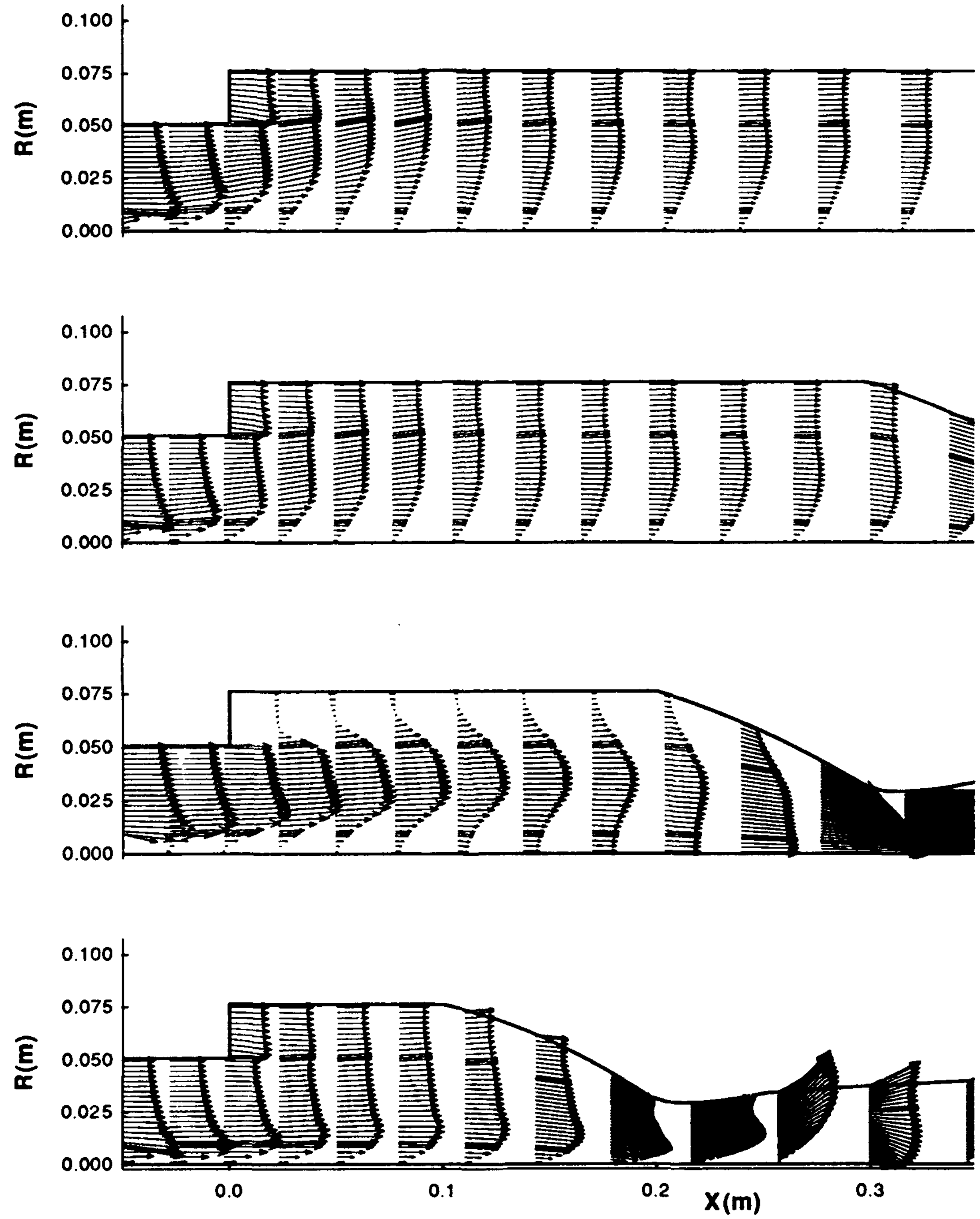

Figure 56: Calculated Swirl-Radial Velocity Vectors (W-V) for Dump Combustor and Closely Coupled CD Nozzle.

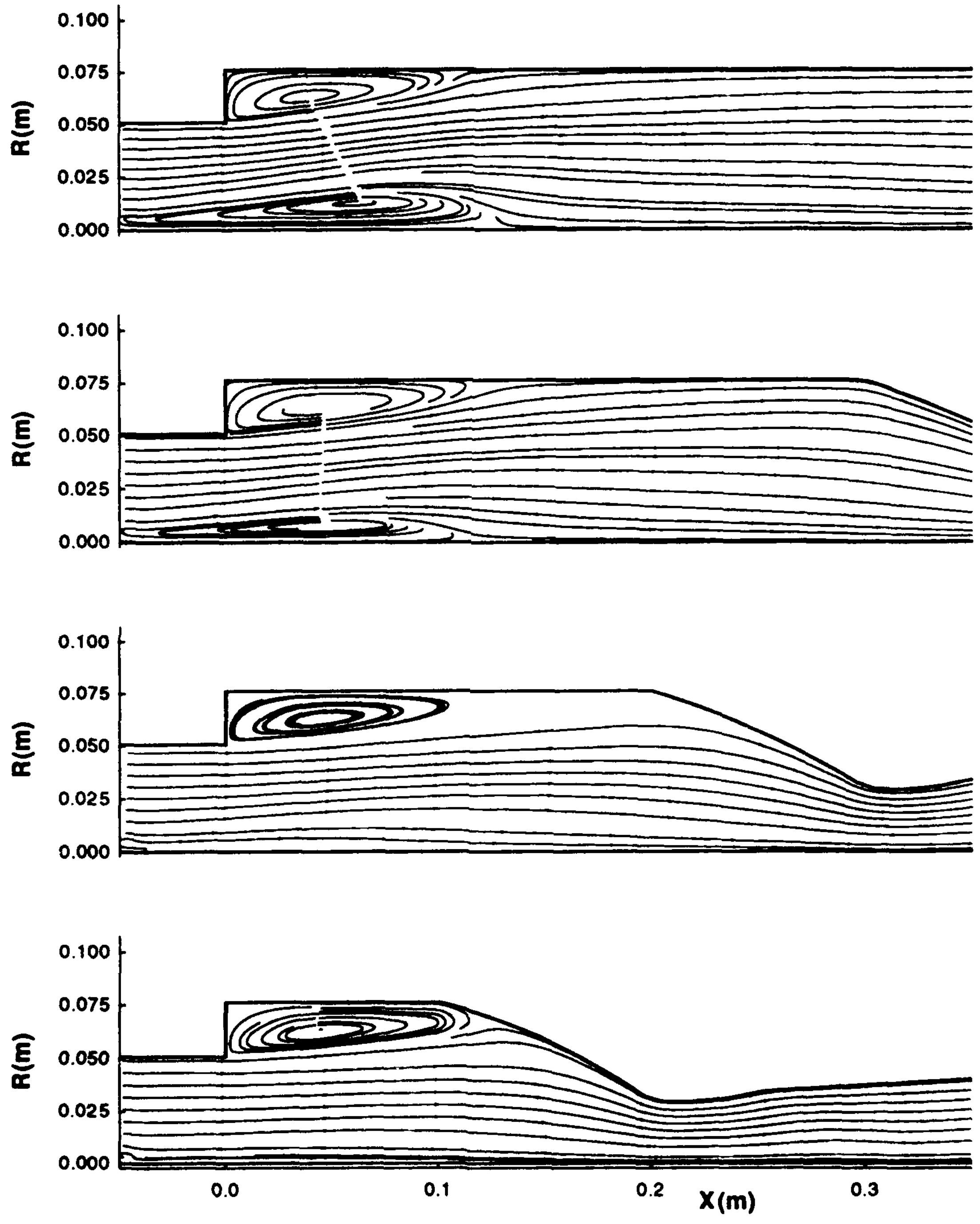

Figure 57: Calculated U-V Streamlines for Dump Combustor and Closely Coupled CD Nozzle.

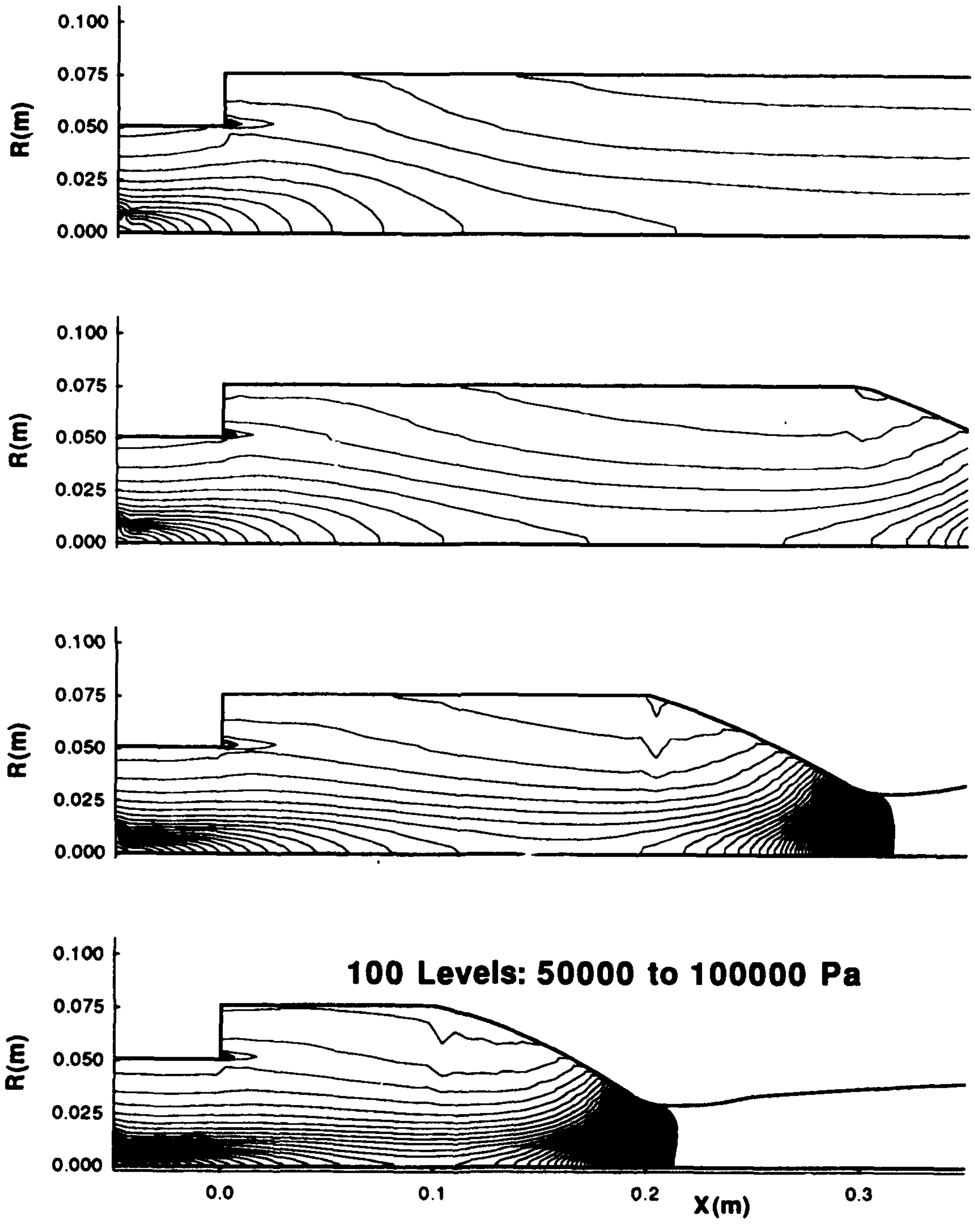

Figure 58: Calculated Pressure Contours with for Dump Combustor and Closely Coupled CD Nozzle.

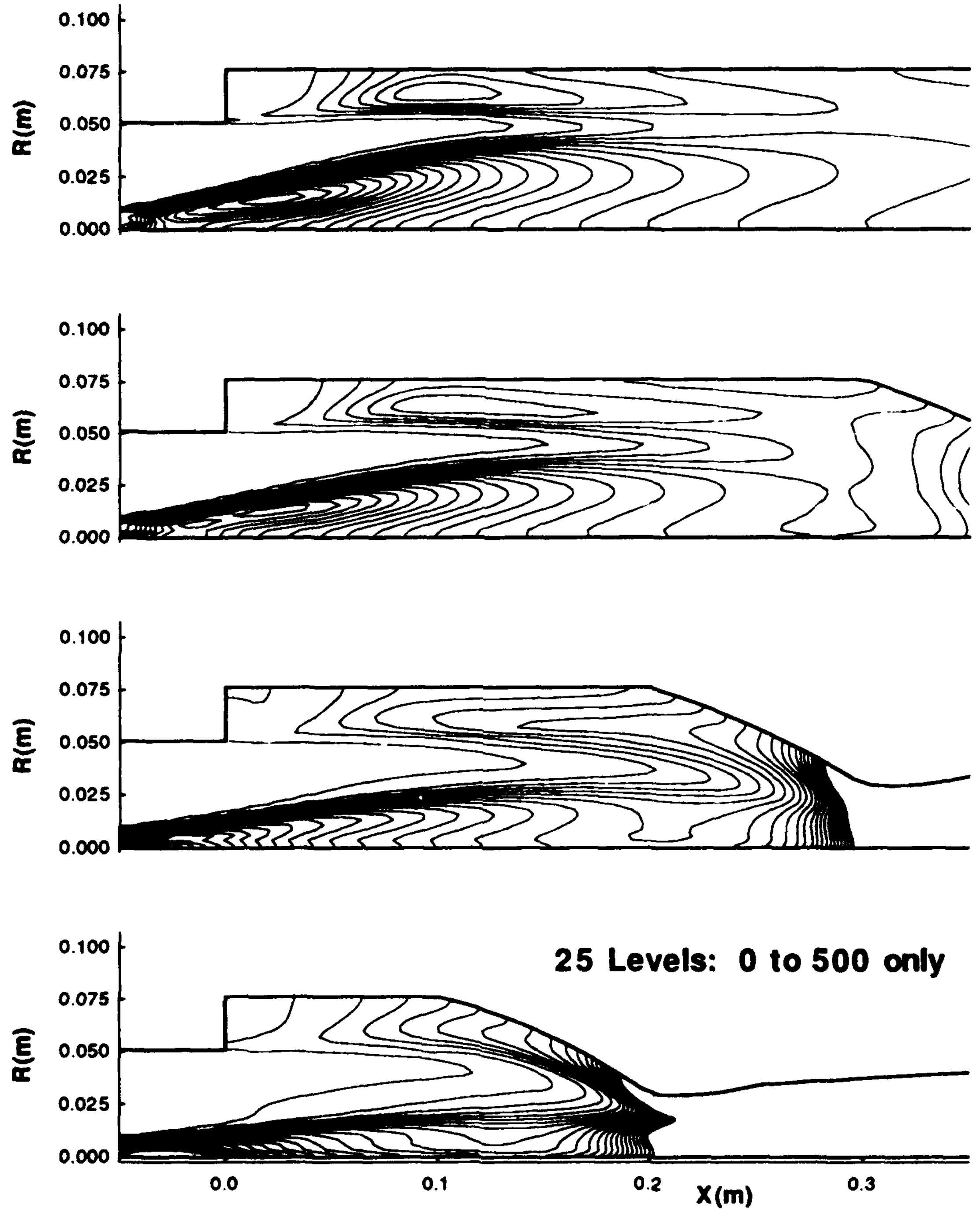

Figure 59: Calculated Turbulent Kinetic Energy Contours for Dump Combustor and Closely Coupled CD Nozzle. $S = 0.5$.

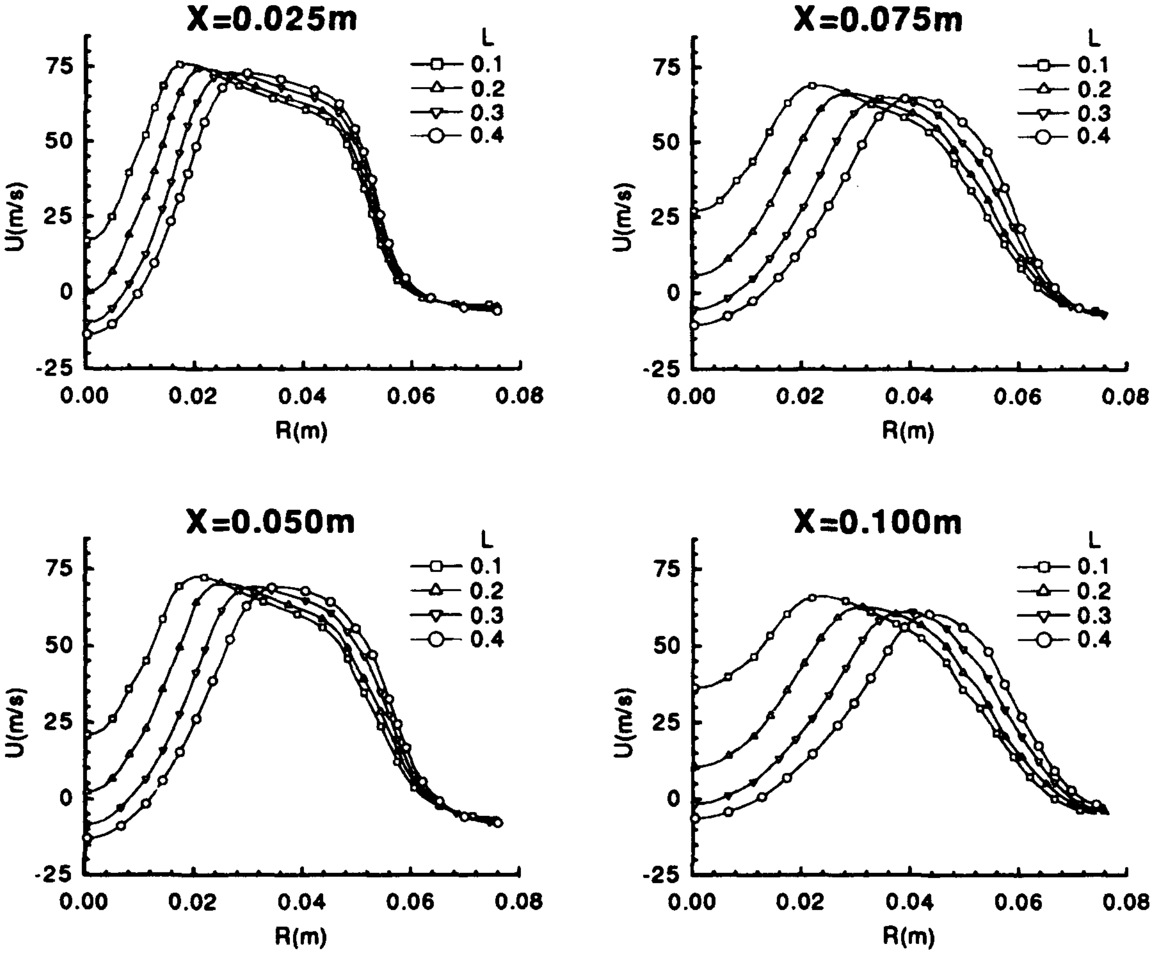

Figure 60: Axial Velocity Profiles for Dump Combustor and Closely Coupled CD Nozzle. $S = 0.5$

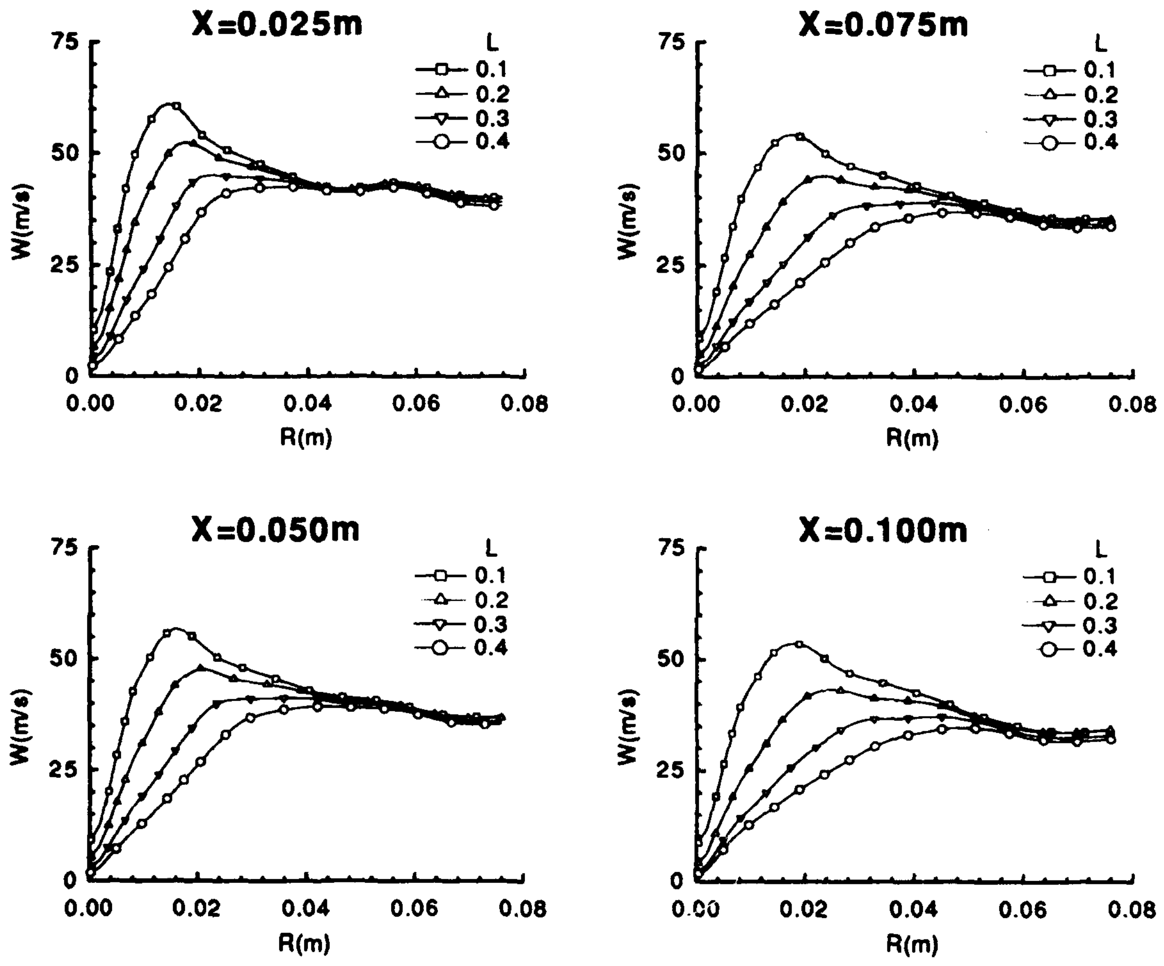

Figure 61: Swirl Velocity Profiles for Dump Combustor and Closely Coupled CD Nozzle. $S = 0.5$

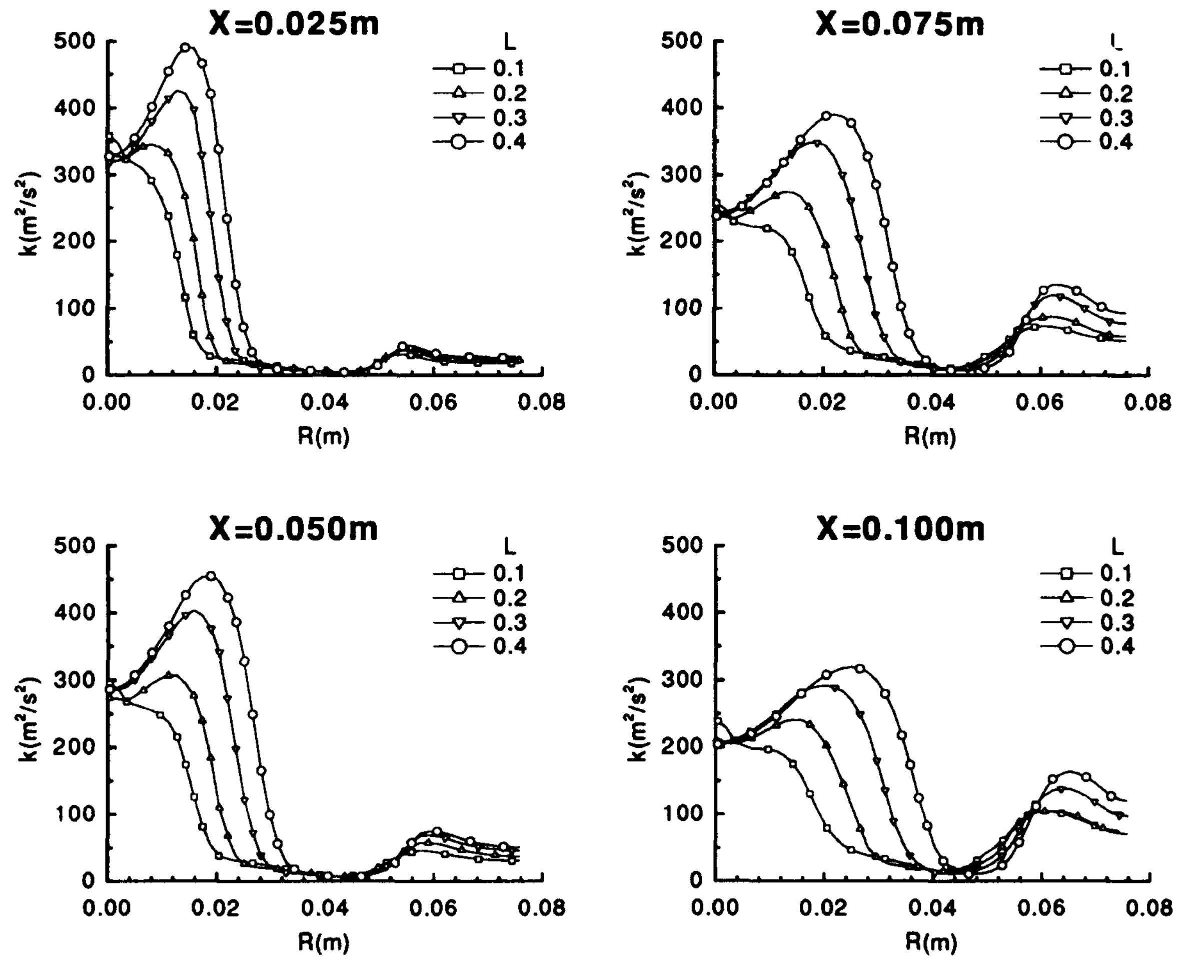

Figure 62: Turbulent Kinetic Energy Profiles for Dump Combustor and Closely Coupled CD Nozzle. $S = 0.5$

7 CONCLUSIONS AND RECOMMENDATIONS

1. The DUMPSTER computer code was developed and applied to nonreacting axisymmetric dump combustor flow field with closely coupled CD nozzle.

2. The code uses a new implicit solution (LU-SGS) procedure which allows the use of low-dissipation flux functions by Harten-Yee and Roe. Low dissipation is advantageous, because it introduces less aphysical diffusion which can reduce the accuracy of the simulation. The LGS solution procedure was also developed. The LGS was dependent upon a dissipative flux function for convergence, and therefore, discarded in favor of the LU-SGS method.

3. DUMPSTER has been used to simulate the flow field of several a non-reacting dump combustor flow field. The computed results agree resonably with both the measurements and with another flow analysis that had been applied to the same problem.

4. DUMPSTER has been documented and delivered in source code form for use in the AFAPL. The code uses an extended NAMELIST input format to simplify setting up problems. The code tracks the transport of trace species and has options for both the Baldwin-Lomax algebraic turbulence model and the k–ϵ two-equation model. The computational mesh is constructed of one or more zones (or blocks) of mesh cells, making mesh generation and specification easy.

5. The calculations indicate that the standard k–ϵ turbulence model is not adequate for accurately simulating swirling dump combustor flow fields. It does provide a resonably good simulation if the constants are adjusted, but this would not be satisfactory for *predicting* dump combustor flow fields.

References

[1] Lilley, D.G.,"Flowfield Modeling in Pracical combustors: A review," AIAA *Journal of Energy*, Vol. 4, No. 4, July-August 1979, pp. 193- 210.

[2] Syred, N. and Beer, J.M.,"Combustion in Swirling Flows: A Review," *AIAA Journal*, Vol. 15, No. 8, August 1977, pp. 1063-1078.

[3] Lilley, D.G.,"Prospects for Computer Modeling in Ramjet Combustors," AIAA Paper No. 80-1189, June 30, 1980.

[4] Escudier, M.P. and Keller, J.J.,"Recirculation in Swirling Flow; A Manifestation of Vortex Breakdown," *AIAA Journal*, Vol. 23, No. 1, January 1985, pp. 111-116.

[5] Lilley, D.G.,"Investigations of Flowfields Found in Typical Combustor Geometries," NASA Report No. N84-20544, 1985.

[6] Yoon, H.K. and Lilley, D.G.,"Five-Hole Pitot Probe Time-Mean Velocity Measurements in Confined Swirling Flows," AIAA Paper No. 83-0315, January 1983.

[7] Lilley, D.G.,"Swirl Flows in Combustion: A Review," *AIAA Journal*, Vol. 15, No. 8, August 1977, pp. 1063-1078.

[8] Lilley, D.G.,"A Computer code for Swirling Turbulent Axisymmetric Recirculating Flows in Practical Combustor Geometries," NASA CR 3442, February 1982.

[9] Abujelala, M.T., Doctorate Thesis, Oklahoma State University, 1984.

[10] Ramos, J.I. and Somer, H.T.,"Swirling flow in a Research Combustor," *AIAA Journal*, Vol. 23, No. 2, February 1985, pp. 241-274.

[11] Rhode, D.L. and Stowers, S.T.,"Turbulence Model Assessment for the Confiend Mixing of Co-Swirling Concentric Jets," AIAA paper No. 85-1269, July 1985.

[12] Gosman, A.D. and Pun, W.M.,"Calculation of Recirculating Flows," Report No. HTS-74-12, Dept. of Mech. Engrg., Imperial College, London, England, 1974.

[13] Peery, K.M. and Forester, C.K.,"Numerical Simulation of Multistream Nozzle Flows," *AIAA Journal*, Vol. 18, No. 9, September 1980, pp 1088- 1094.

[14] Cline, M.C.,"Computation of Two-Dimensional Viscous Nozzle Flows," *AIAA Journal*, Vol. 14, No. 3, March 1976, pp. 295-296.

[15] Drummond, J.P.,"Numerical Study of a Ramjet Dump Combustor Flowfield," *AIAA Journal*, Vol. 23, No. 4, April 1985, pp. 604-611.

[16] MacCormack, R.W.,"Current Status of Numerical Solutions of the Navier-Stokes Equations," AIAA Paper No. 850032, January 1985.

[17] Thomas, J.L. and Walters, R.W.,"Upwind Relaxation Algorithms for the Navier-Stokes Equations," AIAA Paper No. 85-1501, July 1985.

[18] Charkravarthy, S.R.,"Relaxation Methods for Unfactored Implicit Schemes," AIAA Paper No. 84-0165, January 1984.

[19] Beam, R.M. and Warming, R.F., "An Implicit Factored Scheme for the Compressible Navier-Stokes Equations," *AIAA Journal*, Vol. 16, No. 4, 1978, pp. 393-402.

[20] Yoon, S., and Jameson, A., "An LU-SSOR Scheme for the Euler and Navier-Stokes Equations," AIAA Paper No. 87-0600, 1987.

[21] Baldwin, B.S. and Lomax, H.,"Thin Layer Approximation and Algebraic Model for Separated Turbulent Flows," AIAA Paper No. 78-257, January 1978.

[22] Launder, B.E. and Spalding, D.B.,"The Numerical Computation of Turbulent Flows," *Computer Methods in Applied Mechanics and Engineering,*, Vol. 3, 1974, pp. 269-289.

[23] Anderson, D. A. , Tannehill, J. C. , and Pletcher, R. H. *Computational Fluid Mechanics and Heat Transfer*, McGraw-Hill, 1984.

[24] Srinivasan, S., Tannehill, J.C., and Weilmuenster, K.J., "Simplified Curve Fits for the Thermodynamic Properties of Equilibrium Air," ISU-ERI-Ames-86041, Iowa State University, 1986.

[25] Vinokur, M., and Lui, Y., "Equilibrium Gas Flow Computations II: An Analysis of Numerical Formulations of Conservation Laws," AIAA Paper No. 88-0127, 1988.

[26] Srinivasan, S., and Tannehill, J.C., "Simplified Curve Fits for the Transport Properties of Equilibrium Air," NASA CR-178411, 1987.

[27] Mathews, D.C., Childs,M.E, and Paynter, G.C. "Use of Cole's Universal Wake Function for Compressible Turbulent Boundary Layers," AIAA *Journal of Aircraft*, March 1970.

[28] Steger, J.L. and Warming, R.F., "Flux Vector Splitting of the Inviscid Gasdynamic Equations with Applications to Finite-Difference Methods," *Journal of Comp. Phys.*, Vol. 40, pp. 263-293, 1981.

[29] Yee, H.C., Warming, R.F., Harten, A., "Implicit Total Variation Diminishing (TVD) Schemes for Steady State Calculations," J. of Comp. Phys., Vol 57, pp. 327-360, 1985.

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

[31] Pulliam,T.H. and Steger,J.L.,"On Implicit Finite Difference Simulations of Three Dimensional Flows," AIAA Paper No. 78-10, Jan. 1978.

[32] Anderson, W.K., Thomas, J.L., and Van Leer, B., "A Comparison of Finite Volume Flux Vector Splitting for the Euler Equations," AIAA Paper No. 85-0122, 1985.

[33] van Leer, B., "Flux-Vector Splitting for the Euler Equations," ICASE Report 82-30, Sept., 1982.

[34] Harten, A., Lax, P.D., and Van Leer, B., "On Upstream Differencing and Godunov-Type Schemes for Hyperbolic Conservation Laws," SIAM Review, Vol. 25, No. 1, Jan. 1983, pp. 35-61.

[35] Engquist, B., and Osher, S., "One-Sided Difference Approximations for Nonlinear Conservation Laws," Mathematics of Computation, Vol. 36, No. 154, April 1981, pp. 321-351.

[36] Osher, S., and Solomon, F., "Upwind Difference Schemes for Hyperbolic Systems of Conservation Laws," Mathematics of Computation, Vol. 38, No. 158, April 1982, pp. 339-374.

[37] Harten, A., "On a Class of High Resolution Total-Variation-Stable Schemes," NYU Report, Oct., 1982; SIAM J. Num. Anal, Vol. 21, 1984, pp 1-23.

[38] Peery, K.M. and Imlay, S.T., "Blunt-Body Flow Simulations," AIAA Paper No. 88-2904, July 1988.

[39] Peery, K.M. and Imlay, S.T., "An Efficient Implicit Method for Solving Viscous Multi-Stream Nozzle/Afterbody Flow Fields," AIAA Paper No. 86-1380, June 1986.

[40] Users Manual for TECPLOT Version 4.0, Amtec Engineering, Inc., June 1990.

[41] Schlichting, H., "Boundary-Layer Theory," McGraw-Hill Book Company, Seventh Edition, p 339.

[42] Nejad, A.S., Vanka, S.P., Favaloro, S.C., Samimy, M., Langenfeld, C., "Application of Laser Velocimeter for Characterisation of Confined Swirling Flow," The American Society of Mechanical Engineering Paper No. 88-GT-159.

[43] Vanka, S.P., Krazinski, J.L., and Nejad, A.S.,"An Efficient Computational Tool for Ramjet Combustor Research," AIAA Paper No. 88-0060, January 1988.

[44] Lesehziner, M.A. and Rodi, W.,"Computation of Strongly Swirling Axisymmetric Free Jets," *AIAA Journal*, Vol. 22, No. 12, 1984, pp. 1742-1747.

[45] Goldberg, V.C., Gorski, J.J. and Chakravarthy, S.R.,"Transonic Turbulent Flow Computations for Axisymmetric Afterbodies," AIAA Paper No. 85-1639, July 1985.

[46] Imlay, S.T., Kao, T.J., McMaster, D., and MacCormack, R.W.,"Solution of the Navier-Stokes Equations for Flow within a 2D Thrust Reversing Nozzle," AIAA Paper No. 84-0344, January 1984.

[47] Imlay, S.T., "Numerical Solution of 2-D Thrust Reversing and Thrust Vectoring Nozzle Flowfields," AIAA Paper No. 86-0203, Jan. 1986.

[48] Johnson, D.A., "Predictions of Transonic Separated Flow with an Eddy-Viscosity/Reynolds-Shear-Stress Closure Model," AIAA Paper 85-1683, July 1985.

U.S. Government Printing Office 548-076