SECA-FR-93-18

HEAT TRANSFER IN ROCKET ENGINE COMBUSTION CHAMBERS AND REGENERATIVELY COOLED NOZZLES

Contract No. NAS8-38961
Final Report
November 16,1993

Prepared for:

National Aeronautics & Space Administration
George C. Marshall Space Flight Center
Marshall Space Flight Center, AL 35812

SECA, Inc.
Suite 202
3313 Bob Wallace Avenue
Huntsville, AL 35805

FOREWORD

This document presents the results of the research performed by SECA, Inc. under contract NAS8-38961 for NASA, Marshall Space Flight Center. The NASA Contracting Officer's Representative for this study is Mr. Gil Wilhold. The interests of Mr. Kevin Tucker and Dr. Ten-See Wang are also gratefully acknowledged. Part of this study was conducted by Mississippi State University under a subcontract to SECA.

SUMMARY

A conjugate heat transfer computational fluid dynamics (CFD) model to describe regenerative cooling in the main combustion chamber and nozzle and in the injector faceplate region for a launch vehicle class liquid rocket engine was developed. An injector model for sprays which treats the fluid as a variable density, single-phase media was formulated, incorporated into a version of the FDNS code, and used to simulate the injector flow typical of that in the SSME. Various chamber related heat transfer analyses were made to verify the predictive capability of the conjugate heat transfer analysis provided by the FDNS code. The density based version of the FDNS code with the real fluid property models developed in this study was successful in predicting the streamtube combustion of individual injector elements.

TABLE OF CONTENTS

1.0 INTRODUCTION

A conjugate heat transfer computational fluid dynamics (CFD) model to describe regenerative cooling of the main combustion chamber and nozzle for a launch vehicle class liquid rocket engine was developed in the Phase I research program. The Space Shuttle Main Engine (SSME) was used to illustrate this detailed heat transfer analysis (Ref. 1). This Phase II study augmented these heat transfer models by developing additional component models, improving the regenerative cooling wall models, developing a method for coupling all of the unit models into an overall engine heat transfer model, and simulating critical engine operating conditions parametrically. The end product of the investigation was a CFD design tool for predicting structural heat transfer caused by hot combustion gases as they interact with a regenerative cooling system.

The principal objectives of this study are: to predict conjugate heat transfer to critical segments of the main combustion chamber (MCC) using unit models of local phenomena, to incorporate the predictions of unit models into an overall rocket engine heating analysis, to use the overall model to assess the effect of specific physical phenomena on the heat transfer process, and to establish the effects of normal, abnormal, and transient operating conditions on engine thermal response. The need for making submodel analyses is evident by considering the geometric complexity of operational engines like the SSME shown in Figs. 1-3.

The first serious effort to develop unit models to represent flow from individual injectors into the combustion chamber of a rocket motor was presented by Combs (Ref. 2). This work used empirical data from hot wax tests to characterize the atomization process for specific types of injector elements to serve as

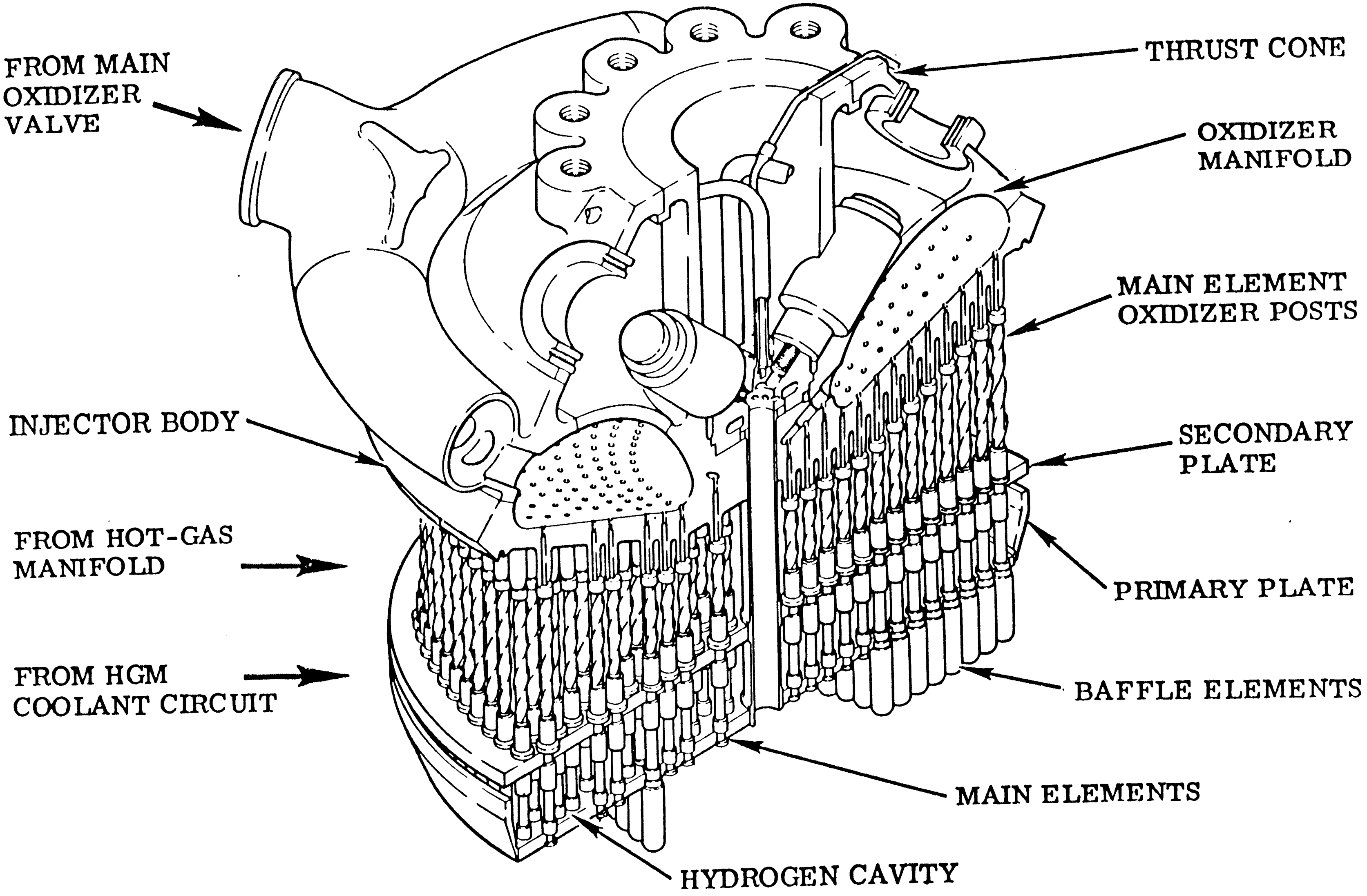


Fig. 1 The SSME Main Injector Assembly

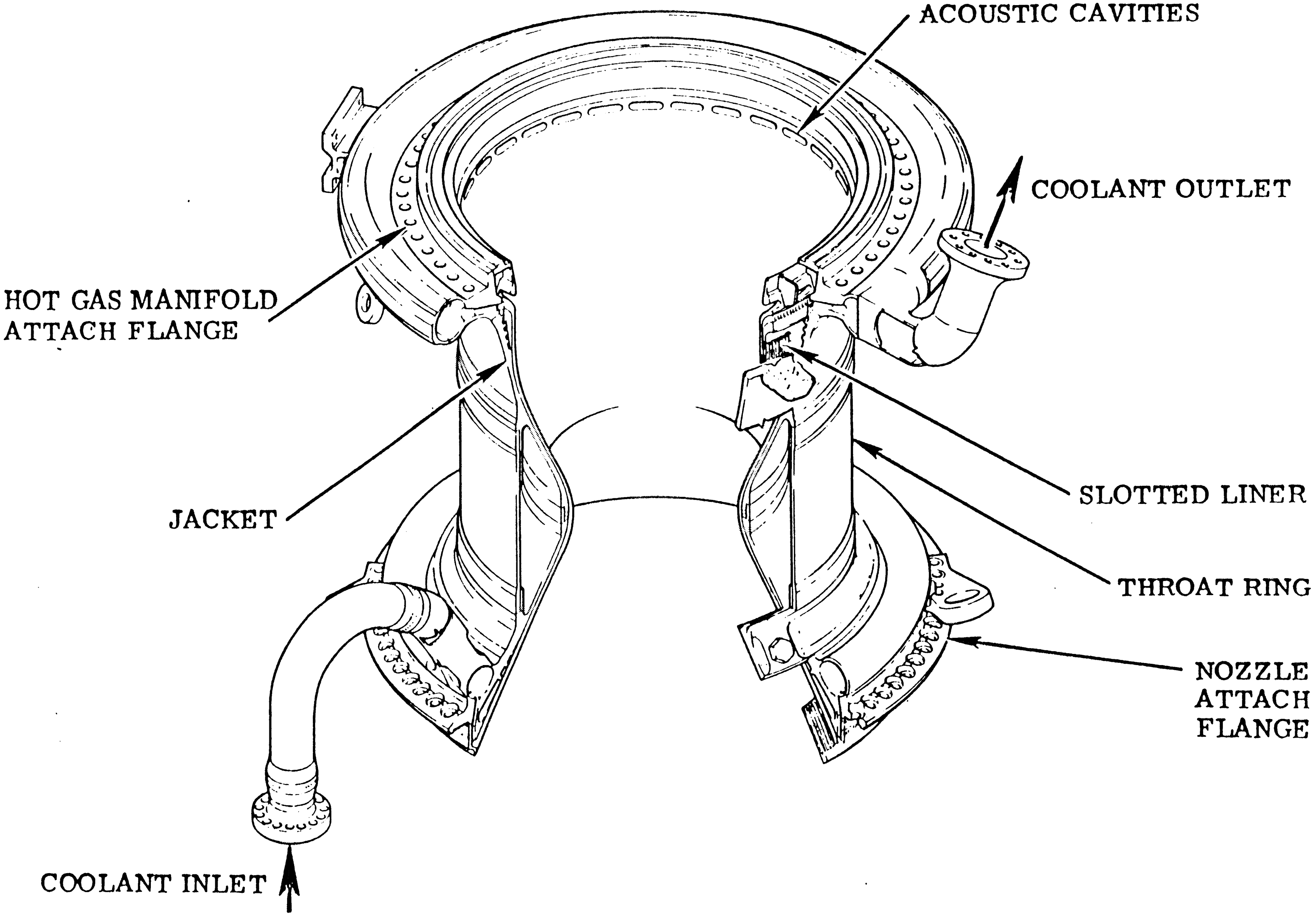


Fig. 2 The SSME MCC

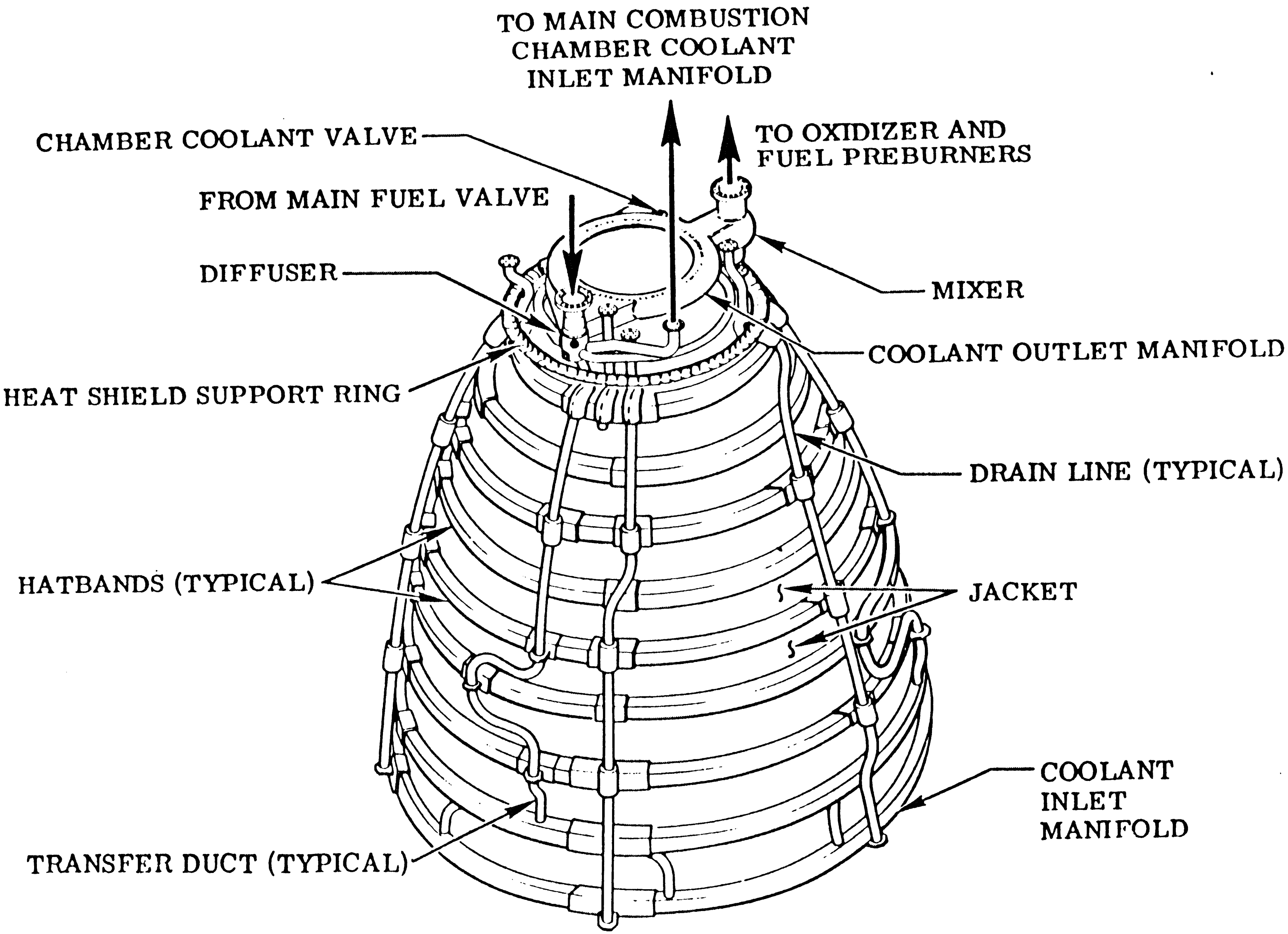


Fig. 3   The SSME Nozzle

boundary conditions for a droplet vaporization/combustion analysis. This work was synthesized into the LIST code. Data for impinging jets was originally included in the code. The output of the LIST code was designed to be the input for a 3-dimensional flowfield code for the entire combustor so that accurate performance and chamber heat transfer analyses could be performed. The concept of the LIST analysis is excellent; however, very few analyses using this methodology have been made. Before simply reviving this code and applying it to current problems, two areas of improvement must be considered. Two generations of improvement in empirical characterization of injectors have been made: water/nitrogen sprays and cryogenic sprays have been measured and correlations of these data may be used to replace the hot wax data base. It has not yet been demonstrated that the hot wax data correlations need improvement. Furthermore, complete reliance on empirical data is no longer necessary, since computational fluid dynamic (CFD) analyses for the propellant flows through the powerhead have now been made so that local mixture ratios (by using local flowrates) into the main combustion chamber are now predictable. Such predictions made with SECA's porosity model are shown in Fig. 4. The second improvement is that the LIST analysis assumes the drops to be surrounded by hot gases at the local equilibrium gas temperature. This assumption is necessary because an upstream boundary condition is needed to start the calculation. If part of the fuel and oxidizer streams are assumed to evaporate, mix, and burn immediately at each point across the faceplate, the upstream boundary condition is specified, albeit at an unrealistically high temperature. In reality, the drops feed a streamtube which probably contains a small recirculation zone such that the feedback of heat provides the required heat of vaporization, or in the supercritical case the energy to heat-up the drops. The backflow of heat from the recirculation provides a much milder environment for the drops. It seems pointless to perform the

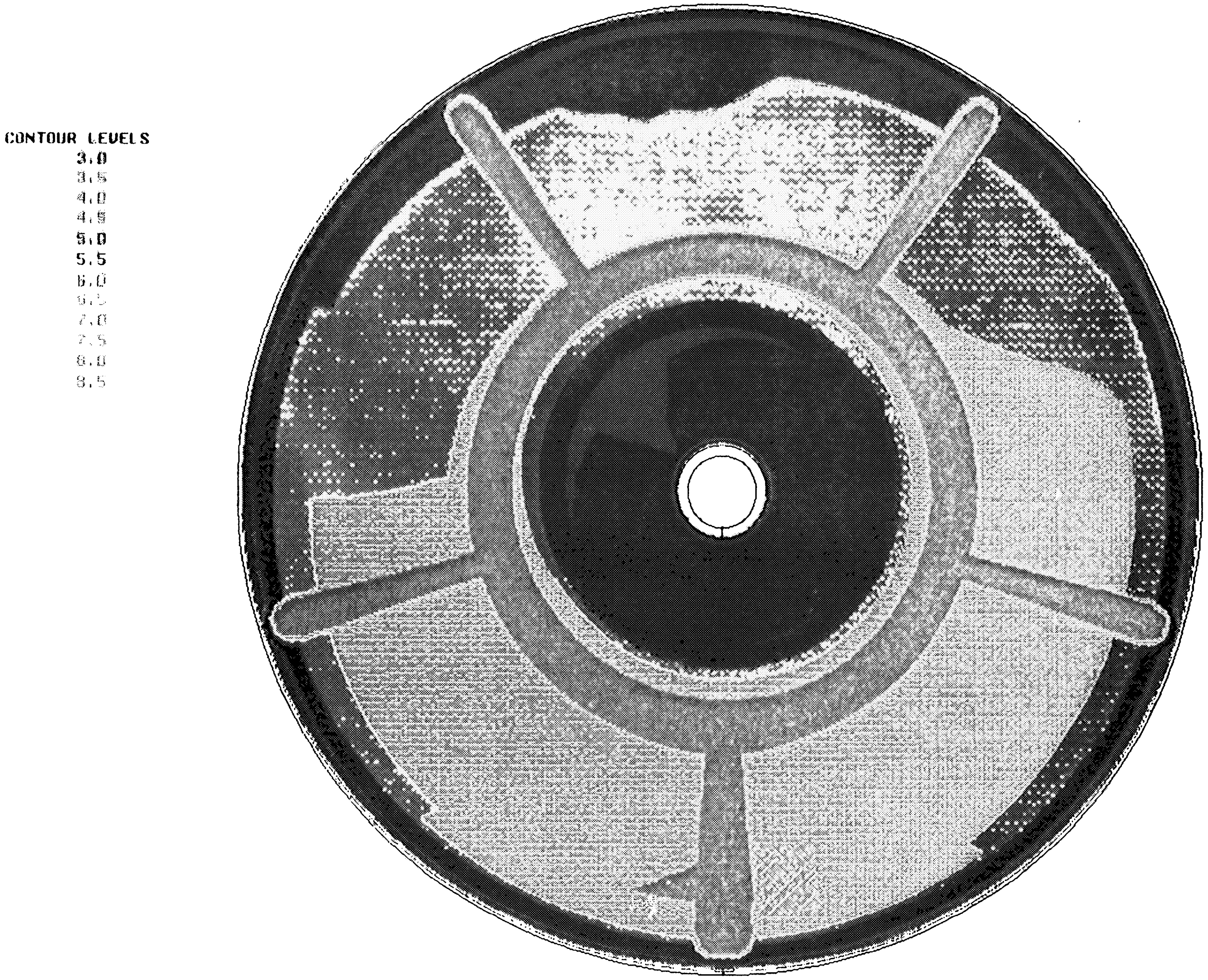


Figure 4 The O/F Ratio Contours of the Main Injector Assembly at 104% Power Level

two-phase calculation to provide a more accurate solution, and immediately compromise that solution by estimating too hot an environment for the drops near the faceplate. Furthermore, this is the very environment which is the necessary result of the analysis for predicting the heat transfer to the injector faceplate. The crux of this investigation was to accurately describe these recirculation zones so that realistic ignition of the inlet flow and the attendant heat transfer into the injector faceplate could be predicted.

2.0 TECHNICAL APPROACH

2.1 Overview

There are 525 main injector elements, 75 baffle injector elements, the acoustic cavity and the porous injector plates in the Space Shuttle Main Engine (SSME) main combustion chamber, plus a large number of regenerative cooling tubes in the nozzle. A prohibitively large number of grid points would be required to simulate the engine hardware, propellant flows, and hot gas flows in sufficient detail to accurately evaluate the conjugate heat transfer within the entire engine. Therefore, a two-pronged approach was employed in performing the rocket engine heating analysis. Unit models for the main injectors and the baffle injectors for the SSME were developed to investigate local phenomena in sufficient detail to identify critical heating phenomena. An overall engine heat transfer analysis was then performed by incorporating results from the unit models into a global model to describe the entire main combustion chamber and nozzle flowfield with the attendant wall boundary conditions. The interplay between the models so developed is required to understand the entire problem. Such a methodology is necessary because the unit models alone can never satisfactorily address the heating problems due to the elliptic nature of the fluid flow.

The conjugate heat transfer model was incorporated into the FDNS code during the Phase I investigation. The FDNS code is a CFD Navier-Stokes equation solver that now includes provision for calculating the heat transfer within bounding structures, thereby being capable of describing temperatures at structural node points for regenerative cooling systems. This code will be used to perform the conjugate heat transfer analyses required in this study. Although the Phase I studies were limited to axisymmetric investigations of the SSME, the FDNS code also treats three-dimensional two-phase flows. The current state of the FDNS code is described in Ref. 3. The resulting conjugate heat transfer model will not apply only to the SSME, rather the SSME will serve as an example to focus the model development.

2.2 Spray Model

Before beginning the detailed development of the unit models, the rational for the treatment of the injected sprays will be presented. As the fluids leave the injector and enter the combustion chamber, several phenomena occur sequentially. First, the mechanical action of the injector element creates a "spray". This is not a liquid-droplet/gas-carrier flow typical of a hydrocarbon fuel spray injection. The hydrogen in the warm exhaust gas stream is well above its critical point and will not form a spray. The steam in the warm exhaust gas stream is slightly above its critical point, consequently it will condense and form a spray as it is cooled by the oxygen. The oxygen is below its critical temperature but well above its critical pressure, hence it is a dense gas. Thus, oxygen and water globules should be formed which are of a size and dispersion which is controlled by the ambient warm gas flow and the injector geometry. This "break-up" region is so near the injector that negligible heat transfer occurs between the globule and carrier gas.

The second flow region is characterized by the carrier stream and fluid globules attempting to reach a thermal equilibrium. This region merges into the third region, a combusted state which is stabilized by the sonic throat of the nozzle.

The igniter causes the ignition, but the steady flow is not a flame spread from the ignition source, rather it is a more uniform combustion stabilized by the upstream conduction of heat. The third region begins within inches of the injector face. The initial globule dispersion and subsequent heat-up would not require analysis if the local O/F and mass fluxes could be defined (or assumed) on a plane across the chamber at a small distance downstream of the injector face in the post-combustion region. Such distributions can be inferred from cold-flow tests which are performed as part of the design procedure by Rocketdyne and other engine manufacturers.

Most previous attempts to describe the near-injector flow have been based on a liquid spray flame model which includes a droplet vaporization/flame submodel (Ref. 4). The supercritical nature of oxygen globules is not well represented by such models in two respects: (1) there is no surface tension to stabilize the droplets, hence the atomization would appear like very high local Weber number sprays which tend to disintegrate the droplets; and (2) there is no requirement for supplying heat of vaporization to the globule, so that thermal equilibration of the gas carrier/globule mixture should occur fairly rapidly.

Since the oxygen is injected under supercritical conditions, the "spray" will actually be a variable density turbulent jet. Faeth (Refs. 5 & 6) developed spray models for dense liquid jets discharging into air which he termed "locally homogeneous" spray models. These models treat the liquid as a turbulent jet, neglect the identification of drop sizes, and treat the two-phase mixture as a binary mixture of air and fuel which exhibits no velocity or thermal lag between the phases. Faeth's model treats turbulent momentum exchange with a k-ϵ turbulence model and mass transport with a specified probability distribution function (PDF) to represent the spray. Exactly the same type calculation can be made with the FDNS code to represent the oxygen spray. Faeth assumed that the sprays which he studied were unbounded, this is not a good assumption for main injector or baffle injector elements. The injector elements feed a streamtube whose cross-sectional area is determined by considering the entire injector face layout. This streamtube model will allow recirculation of burned gases to surround the cold injector flow jets and act as a flame holder. This is qualitatively the same flow phenomena which has been observed to stabilize large scale spray flames (Ref. 7). The necessity of using a PDF mixing model to describe the oxygen jet, rather than simply a Schmidt number has not been established. The studies which are now being conducted by Bachalo of Aerometrics (Ref. 8) and by Eskridge of MSFC (Ref. 9) will be most useful for describing mass transfer in cryogenic jet sprays. Preliminary data from Ref. 8 already suggests that the globules of oxygen will be very small in size and confined to a more narrow jet than non-cryogenic fluid flowing through the same injector.

Since the velocity of the drops and that of their immediate environment is relatively slow near the injector face and since the atomization process has not been modeled to the extent that drop sizes can be predicted, the thermal and velocity equilibrium assumptions between the phases are considered reasonable, therefore the physics of Faeth's homogeneous spray model were used for unit model development. As better thermodynamic analyses, such as those presented herein, and as more complete experimental data on cryogenic sprays become available, droplet tracking submodels can be developed and incorporated in the unit models offered in this study.

3.0 UNIT INJECTOR AND COMBUSTION CHAMBER ANALYSES

3.1 General Approach

An injector unit model starts with predicted oxidizer flow from the core of a single coaxial injector, hot turbine exhaust gases, consisting of hydrogen and steam, flow from the annulus, and hydrogen bleed flow from the porous primary injector plate. An intermediate calculation then establishes the flow field in the region between the face plate and an established flame in the chamber. In general, this is a very complex calculation because it is three-dimensional and mathematically elliptic. However, if each injector is simulated by assuming that it feeds a streamtube and if the composition and temperature of the ambient environment of the injected jet were specified, the analysis could be accomplished on an injector-by-injector basis. The streamtube assumption is obvious and requires no discussion. Specifying the ambient conditions is difficult. An analytical and experimental furnace study (Ref. 7) has provided further insight to this problem.

The furnace studied was fed by a single large-scale coaxial jet, and flow was controlled with an exit nozzle. Conceptually, this is analogous to a streamtube analysis for a single coaxial injector element. The recirculated, ambient gases surrounding the coaxial jet were measured to be entirely combustion products. This suggests that the coaxial element can be simulated by assuming that the ambient gases will be drawn from a stagnant atmosphere of combustion products admixed with H_2 bleed gases. The assumed recirculating combustion gases would provide the steady state ignition source, and the flame would stand-off the injector face by a distance which is determined by the calculation. No arbitrary ignition source would be required because the analysis would behave elliptically. Notice also that the proper and predictable thermal level of all the bounding streams would be accurately accounted for.

Consideration was also given to using the results of empirical correlations of cold flow data taken on various rocket motor injector configurations (Ref. 10). These data confirm the basic structure of a single primary jet resulting from merging of the two concentric jets. However, ambient gas composition and the axial extent of the effective primary jet are not provided by these correlations. Therefore, no way of using such correlations can be presently determined. Consequently, the unit injectors were modeled numerically and are presented below.

3.2 Main Injector Element

The analysis of the flow field and conjugate heat transfer of a typical main injector element was accomplished using the FDNS code. The analysis was performed in order to determine the effects of flow losses and heat transfer on the thermodynamic properties of the fluids as they traverse the long narrow elements and enter the combustion chamber. The results of this analysis will serve as input data for analyzing the combustion process in the chamber itself. The analysis includes the flow of oxygen from the LOX dome to the combustion chamber, the flow of hot turbine exhaust gases from the exhaust manifold to the chamber, the heat transfer from the environment surrounding the element, and the transfer of heat from the hot exhaust gases to the cold oxygen through the LOX post wall.

3.2.1 Geometry

A typical element was used because of the existence of many variations in configuration. There are 525 main injector elements in each engine, arranged in 13 rows. The elements vary in length from row to row and vary in configuration from engine to engine. For instance, some elements contain a LOX post consisting of a single tube extending from the LOX dome to the primary injector plate while others consist of two separate pieces. The accompanying retainers, filters and sleeves vary in configuration to accommodate the different posts. The cross sectional area of the holes in the retainer that feed hot turbine exhaust gases into the element vary from modification to modification. Consequently, no single configuration exists that would be typical of all elements.

As a result of this large variation in configurations, a composite main injector element has been synthesized based mainly on modifications M83 through M99 and is shown in Figure 5. This element has been sized to conform to several reference lengths provided by drawing RS009122. The downstream, or hot side, of the primary injector plate is located 9.538 inches from the reference plane "-A-" on the LOX dome as shown on the drawing. Reference plane -A- is defined on drawing RS009138. The primary and secondary injector plates have reference thicknesses of 0.25 inches and are a reference distance of 2.25 inches apart. The variations in element length are manifested in variations in the length of the spiral-shaped spoiler and are dictated by the row-to-row variation in the distance from the LOX dome to the hot side of the primary injector plate.

The single-piece version of the LOX post was chosen and modeled from drawing RS009207. Choice of the single or two-piece versions is not consequential to this analysis. The filter and retainer are modeled from drawing RS009133, the sleeve from RS009131 and the nut from RS009132. The injector plates are modeled from drawings RS009140 and RS009141.

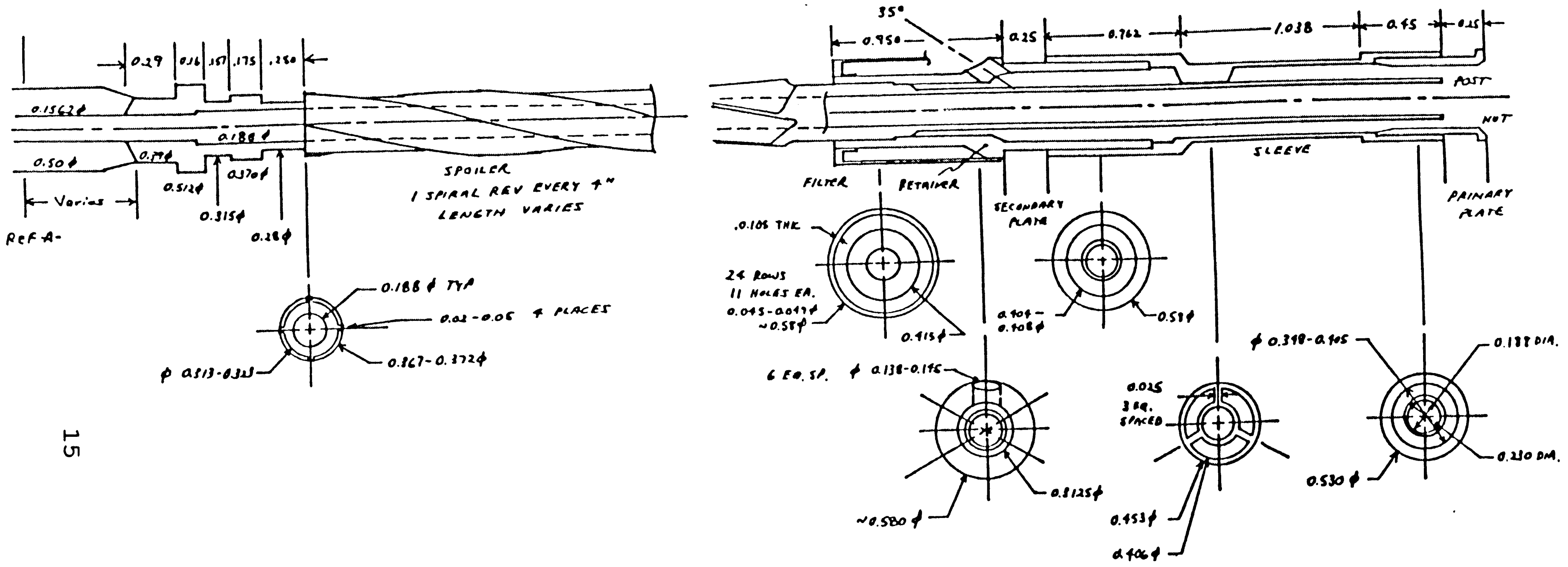


Fig. 5. Main Injector Element Configuration

The grid generator for the geometry of the main injector element has been coded based on the above criteria and is presented as Appendix A to this report. The grid code has been written to facilitate changing dimensions in the event more than one configuration must be analyzed. The code also allows the user to isolate and analyze specific portions of the element. For instance, the user can analyze the portion of the element between the LOX dome and the secondary plate, taking care to employ appropriate boundary conditions.

Several simplifications were included in the model. The retainer has 6 equally spaced holes where hot turbine exhaust gases enter the element. These holes have a diameter of 0.138 to 0.145 inches and are angled at 35 degrees to the element axis. In order to reduce the number of grid points required, these holes were modeled as a slot with the same total cross sectional area.

The filter covering the retainer was not included. The filter contains 24 rows of 11 holes each, each hole with a diameter of 0.045 to 0.049 inches, resulting in a flow area much larger than the flow area in the retainer. The filter is, therefore, not consequential to this heat transfer analysis and was omitted.

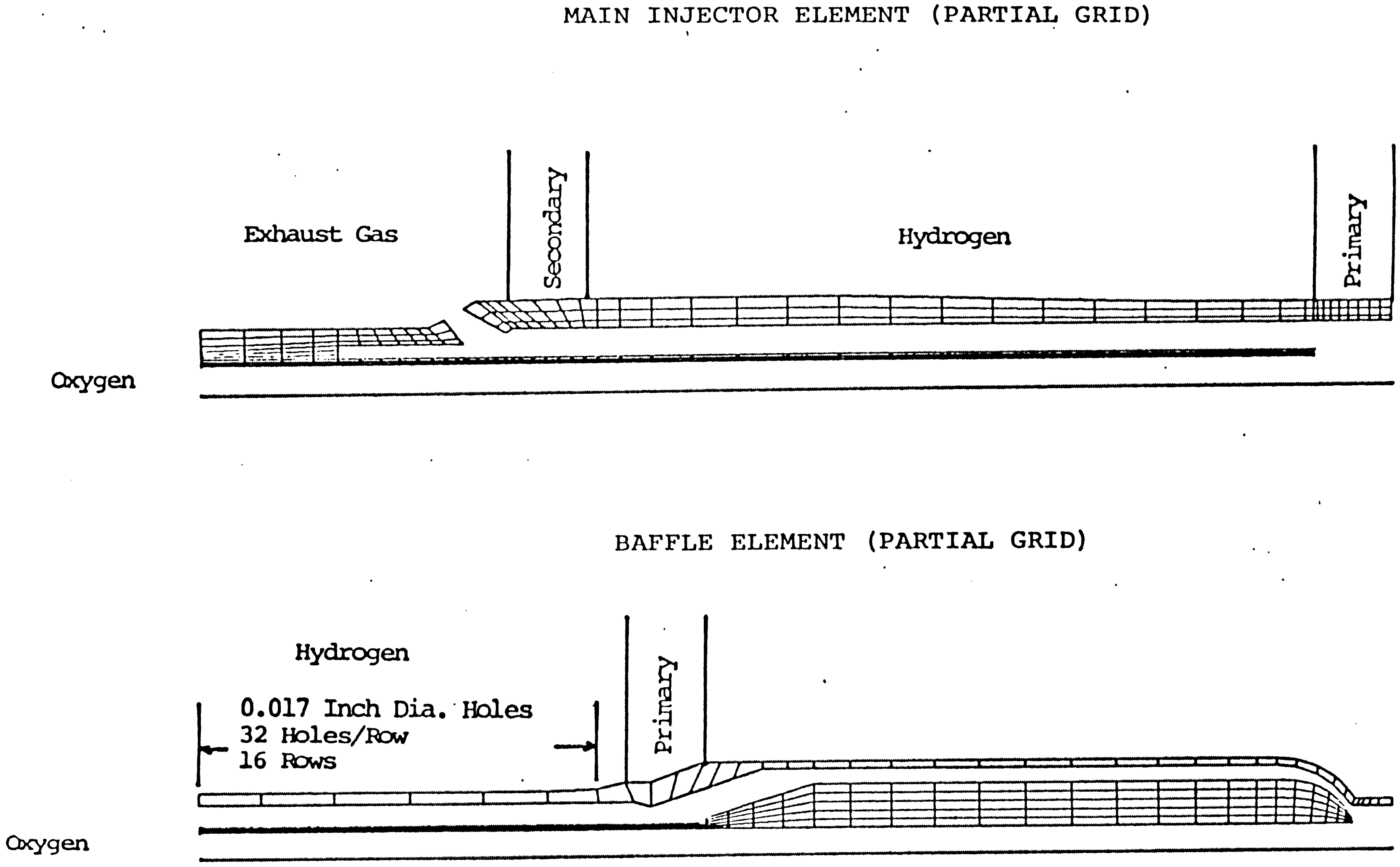
The LOX posts have a spiral-shaped spoiler located in the hot gas area between the LOX dome and the secondary injector plate. Although modeling this spiral section is not difficult, doing so results in a very complicated grid. Also, the heat transfer analysis does not account for the effects of spiraled ridges. Therefore, the spiral shape of the spoiler was not modeled.

The geometry of the element was modeled in 3-D assuming a plane of symmetry parallel to the element axis. However, the resulting grid required too many nodes to allow a reasonably quick solution to the flow field. Therefore, the geometry was remodelled axisymmetrically and is presented in Figure 6. For clarity, only grid points in the element structures are shown. The actual grid contains 10,962 nodes. There are 13 nodes across the LOX post oxygen flow path, 6 across the LOX post wall, 13 across the hot exhaust gas flow path, 13 across the outer wall and 261 along the axis. The 6 holes in the retainer have been modeled as a single slot with the equivalent flow area of the holes. There are 11 nodes across the slot and 13 nodes along the slot.

3.2.2 Fluid Properties

The flow of oxygen from the LOX dome to the combustion chamber involves the transition from a liquid to a vapor at pressures and temperatures well above the critical conditions. Also, the steam in the hot exhaust gas could drop below its critical temperature when mixing with cold oxygen and condense. These operating conditions, along with fluid critical conditions and compressibility factors, are summarized in Table 1. Since the compressibilities differ substantially from unity, ideal gas models, even with temperature dependent heat capacities, are not an accurate representation of these fluids. Therefore, a real fluid thermodynamic model which accurately describes all states of the fluids is required.

Oxygen and hydrogen properties over the complete gas/liquid regime are described in NBS reports Ref. 11 and 12, respectively. Oxygen properties have been accurately correlated and modeled with appropriate generalized equations of state (Refs. 13 and 14). SECA used these sources to develop a code to generate



MAIN INJECTOR ELEMENT (PARTIAL GRID)

BAFFLE ELEMENT (PARTIAL GRID)

Fig. 6. Grid Systems For Coaxial Injectors

Table 1. Precombustion Flow Conditions

	H_2	H_2O	O_2
T_c (°R)	59.357	1165.16	278.237
P_c (lbm/ft^3)	187.51	3208.2	731.4
Temperature Range (°R)	500 - 650 1600 - 1800	1600 - 1800	200 - 400
Pressure Range (psia)	3200 - 3500	3200 - 3500	3200 - 3500
Reduced Temperature Range	8.5 - 11.0 25.4 - 30.5	1.3 - 1.5	0.7 - 1.4
Reduced Pressure Range	17.1 - 18.7	1.0 - 1.1	4.4 - 4.8
Approximate Compressibility Factor	1.8	0.8	1.8

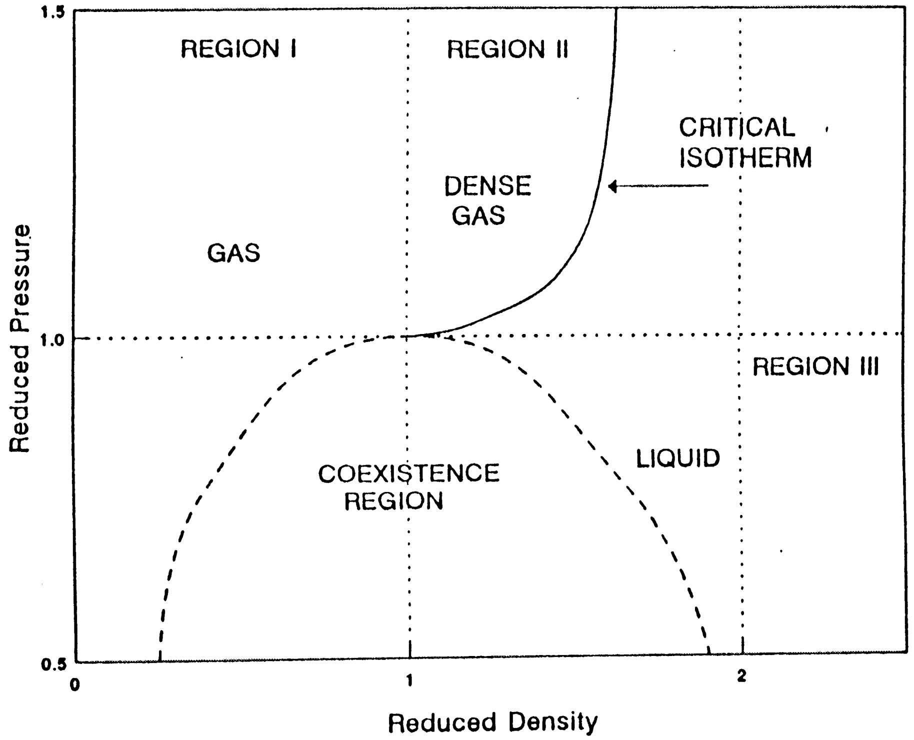
tables of oxygen fluid properties, Ref. 15. This code was used as the basis for developing a real fluid properties model for the FDNS code. The model was expanded to include hydrogen and water properties. Water properties were obtained form Ref. 16.

Thermodynamic properties of pressure (P) and enthalpy (h) are given in terms of density (ρ) and temperature (T) using the HBMS equations, Ref. 14. The entire liquid-vapor regime for each fluid is divided into 4 regions, as shown in Figure 7. Region 1 consists of 2 parts; one part is for subcritical temperatures and densities less than the saturated vapor density at that temperature, and the second part is for supercritical temperatures and subcritical densities. Region 2, the "dense gas" region, models supercritical temperatures and densities. Region 3 models the liquid regime and region 4 is the liquid-vapor two-phase region.

The form of the enthalpy calculated using The HBMS equations is a deviation from the ideal gas enthalpy. The actual enthalpy is obtained by adding the ideal gas enthalpy, calculated using standard CEC thermodynamic data, to the defect value.

In order to determine which region is appropriate for subcritical temperatures, the liquid and vapor saturation lines must be modeled to obtain saturation densities and enthalpies. Vapor pressure was modeled using NBS high-order polynomials in temperature for low temperatures and Riedel's equation (Ref. 17) for temperatures near the critical point. The enthalpy of vaporization was estimated using Clapeyron's equation but was corrected for near-critical temperatures using the empirical equation (Ref. 17)

$$\Delta H_v = \Delta H_{v_{ref}} \left(\frac{1-t}{1-t_{ref}} \right)^{0.38}$$



For each region

$$\frac{P}{P_{crit}} = \sum_{j=1}^{4} t^{j-2} \sum_{i=1}^{6} B_{ij} \rho^{i-2} + A(t)$$

$$\frac{H-H_o}{RT} = Z_c\int_a^b [\frac{p}{t} - (\frac{\delta p}{\delta t})_\rho] \rho^{-2} d\rho + Z_c \frac{p}{\rho t} + C(t)$$

Also , P_v, ρ_L, H_o are functions of t and ρ_v is a function of t and P_v

Fig. 7. Generic Liquid - Vapor Regime

t and t_{ref} (=0.9) are reduced temperatures. Liquid saturation densities are estimated by HBMS's correction equations (Ref. 14) using two reference values on the saturation curve from NBS data.

Thermal conductivity and viscosity are given by NBS correlations (Ref. 11). Experimental reference values of viscosity and thermal conductivity are input. Temperature corrections are described by:

$$\mu^o/\mu_c = t{**}(0.71+0.29/t)$$

from Ref. 17. μ^o is the low pressure viscosity, and μ_c is the viscosity of the critical point; this value is estimated by forcing the equation to fit the reference value. Reported pressure corrections to the low pressure viscosity values (μ^o) are extremely complex even though the corrections are small. Therefore, a simple linear correlation was developed, namely:

μ/μ^o = (1+0.0447p) for H_2O, and

μ/μ^o = (1+0.0058p) for H_2

where p is reduced pressure.

For thermal conductivity, temperature corrections are:

$$\lambda^o/\lambda_{ref} = (T/T_{ref})^N$$

N = 1.4544 for H_2O, and
N = 0.740 for H_2
λ^o is the low pressure thermal conductivity, "ref" denotes reference values, and T is temperature.

Pressure corrections are:

$$\lambda/\lambda^{\circ} = (1+0.1842p) \text{ for } H_2O,$$

and are assumed negligible in the NBS tables for H_2. Oxygen properties from NBS 384 (Ref. 11), hydrogen properties from NBS 617 (Ref. 12), and steam properties from steam tables (Ref. 16) are well fitted by the predictive code.

The original version of the fluid properties model has been modified several times in order to make it more compatible with the FDNS code. The empirical relations for saturation conditions near the critical point were inaccurate enough to create oscillations in FDNS results as oxygen and water properties neared critical values. This problem was resolved by replacing the empirical relations with tabulated data from NBS and steam tables. These sources present few data points near the critical point, therefore additional data points were created by cubic spline fitting the existing points.

Highly accurate saturation properties for oxygen have been obtained by taking NBS data, creating a large population using cubic spline fitting, then applying a least-squares fit using Chebyshev polynomials to produce high-order polynomials for the properties. This method producing very accurate properties but will not be incorporated into the model until the same procedure is applied to hydrogen and water data.

Extensive check-out of the resulting fluid properties model was conducted to insure their accuracy. Numerous calculations were made at many different temperatures and densities in each of the four regions and compared to NBS and steam table properties. The model was then incorporated into the FDNS code.

3.2.3 External Wall Temperature

The conjugate heat transfer analysis requires a specified wall temperature distribution along the outside of each element. White (Ref. 18) presents an algorithm for computing a Nusselt number for banks of staggered and in-line cylinders in cross flow. The Nusselt numbers for the flow of hot exhaust gas between the LOX dome and the secondary plate, and the flow of coolant hydrogen between the injector plates, can be approximated by averaging the staggered and in-line algorithms presented in the reference. These Nusselt numbers, along with specified hot exhaust gas and hydrogen temperatures, can then be used to determine the conductance on the external wall of the elements. Once the wall conductance has been determined, a steady-state, quasi-one dimensional heat transfer analysis can be employed to approximate the external wall temperature distribution.

The steady-state heat transfer rate is

$$q = U(T_a - T_i)$$

$$U = 1/(t_w/k_w + 1/h)$$

where t_w is the wall thickness, k_w is the wall conductivity, h is the external wall conductance determined above, T_a is the temperature of the hot exhaust gas (1600 R) or coolant hydrogen (465 R), and T_i is the wall temperature at a node just inside the surface of the element, as computed by FDNS. But this steady-state heat transfer rate q is also equal to

$$q = h(T_a - T_w)$$

where T_w is the external wall temperature. Then

$$h(T_a - T_w) = U(T_a - T_i)$$

or

$$T_w = T_a - U(T_a - T_i)/h \ .$$

After each flow field and heat transfer iteration in the FDNS code, the above analysis can be used to approximate the wall temperature at each external node and this updated temperature is then used by FDNS to perform the conjugate heat transfer analysis.

3.2.4 Fluid Flow and Heat Transfer Analysis

The flow field and conjugate heat transfer analysis of the main injector element, from the fluid sources through the primary injector plate, was very difficult to compute using the pressure based FDNS code (Ref. 2). The FDNS code was not been able to compute through the two orders of magnitude density variation encountered in the region where the hot exhaust gases and cold oxygen mix. The analysis invariably drove the pressure to either the maximum or minimum allowed in the code.

An alternative approach to obtaining exit plane flow properties was investigated. This alternative would provide the flow field analysis, using FDNS, of the two separate fluid streams only up to the exit plane of the LOX post. This plane is 0.25 inches upstream of the injector element exit plane. The results of the modified analysis would provide the pressures, temperatures, and velocities of the two streams as they enter the mixing region. This data would be used in any of several codes such as GENMIX to compute the mixing of the fluids up to the element exit plane. GENMIX uses a Gaussian probability distribution function (pdf) for the species concentrations. The "tails" of the pdf are clipped to prevent the distribution function from going to infinite extremals. The code uses a two-equation turbulence model but assumes constant density. The results of the mixing analysis would then be used to complete the heat transfer analysis as planned previously.

Unfortunately, the FDNS code was not been able to compute even the two separate flows. An examination of the possible causes for this problem revealed several candidates which are discussed below.

The HBMS fluid properties model discussed above, used to obtain "dense gas" properties for oxygen and hydrogen, has equations for pressure and enthalpy, both being functions of density and temperature. The FDNS code computes enthalpy from the energy conservation equation, then, assuming enthalpy is a function of temperature only, updates the temperature. Next, the code computes pressure from the pressure correction equation, then, assuming constant temperature, updates the density. This incompatibility between the HBMS model and FDNS was driving the temperature and density too hard, resulting in the observed pressure discrepancy. Underrelaxing of the temperature and density corrections merely postponed the problem.

In addition, the pressure correction equation is derived from the continuity equation. The density change in the continuity equation is replaced by a pressure change derived by differentiating the ideal equation of state assuming constant temperature. This substitution, along with a simplified momentum equation, results in a pressure correction equation assuming constant temperature. The use of the ideal gas relationship in this derivation and the assumption of constant temperature may contribute to the problems using the HBMS model in FDNS.

Another problem source is the Dp/Dt term in the energy equation. This term results when the temporal term in the energy equation is cast in terms of enthalpy instead of internal energy. If assumptions similar to those above are employed in evaluating this term, this term could contribute to incompatibilities between the HBMS model and FDNS.

In an effort to circumvent each of the above problems, the FDNS code was modified to compute density and temperature from conservation equations instead of pressure and enthalpy. This allows the HBMS equations to be used directly to obtain pressure and enthalpy, thereby eliminating the incompatibility between the HBMS model and the FDNS code.

The density is calculated by modifying the species mass fraction conservation equations to compute species densities. The fluid density is the sum of the species densities. The same hybrid algorithm (Ref. 19) used for species mass fractions is employed for species densities. This algorithm uses central differencing if the Peclet number is less than 2 and uses first order upwinding otherwise.

The energy equation in the FDNS code is cast in terms of enthalpy, resulting in the Dp/Dt term. The temporal term in the energy equation was recast in terms of internal energy, thereby eliminating the DP/Dt term. Next, the internal-energy form of the equation was recast in terms of temperature assuming the specific heats are locally constant. In the neighborhood of each computational grid point, where small changes in temperature occur during each iteration, this is a good approximation. The specific heats are updated each iteration. This form of the equation is solved using the same algorithm options currently available in the FDNS code.

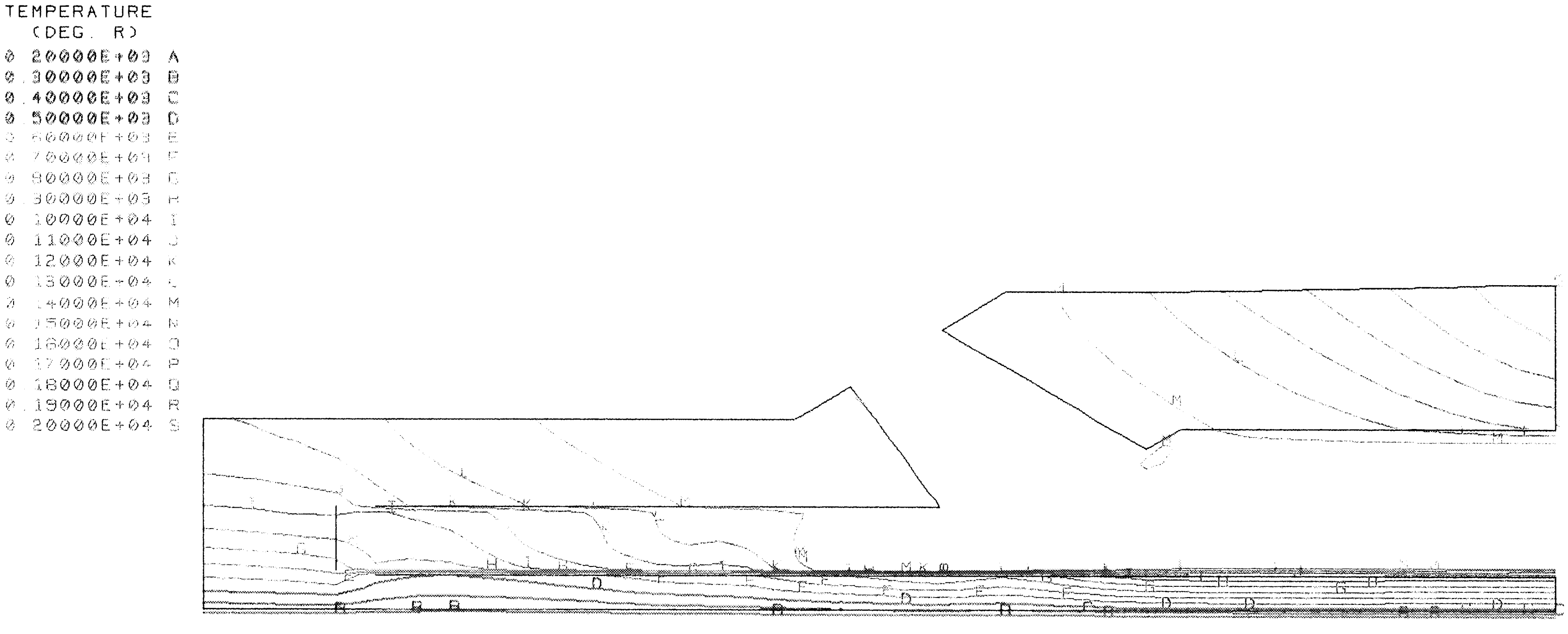
These modifications were incorporated into the FDNS code, and this modified version of FDNS was used to compute the flow field and heat transfer in the main injector element.

The properties of the oxygen as it entered the LOX posts were approximated at 200 R and 3450 PSIA, resulting in a liquid with a density of 68.0665 LBM/FT3. The flowrate was computed by dividing the total flowrate, 877.62 LBM/SEC, by 600. The entrance velocity was obtained by dividing the flowrate by the density and the cross sectional area, resulting in a velocity of 111.50 FPS and a Mach Number of 0.1489.

The hot exhaust gas properties were approximated at a temperature of 1500 R and a pressure of 3527 PSIA, yielding a density of 0.7292 LBM/FT3. The flowrate of 241.3 LBM/SEC had a O/F of 0.8012, resulting in mass fractions of 0.4992 for hydrogen and 0.5008 for steam. The flowrate was 0.45962 LBM/SEC for each of the 525 elements, and when divided by the density and the cross sectional area of the slots, resulted in a inflow velocity of 954.77 FPS and a Mach Number of 0.1776.

The skin temperatures resulting from the wall conductance analysis resulted in temperatures from 975 R near the LOX dome, 1250 R in the exhaust manifold near the secondary plate, and 911 R between the plates.

The results, presented in Figures 8 through 12, show the velocity vectors, temperatures and oxygen mass fraction in the exit region of the element as the flow enters the combustion chamber. The oxygen, entering the LOX post as a liquid at 200 degrees Rankine, reaches temperatures of 240 R (still a liquid) along the element axis and 304 R (dense gas) at the wall in the exit plane of the LOX post, prior to mixing with the hot exhaust gases. As oxygen mixes with the hot gases, the oxygen partial pressure drops below critical pressure but the temperature stays well above critical, and the fluid remains a gas. The exhaust



TEMPERATURE - HOT GAS INLET TO MAIN INJECTOR ELEMENT

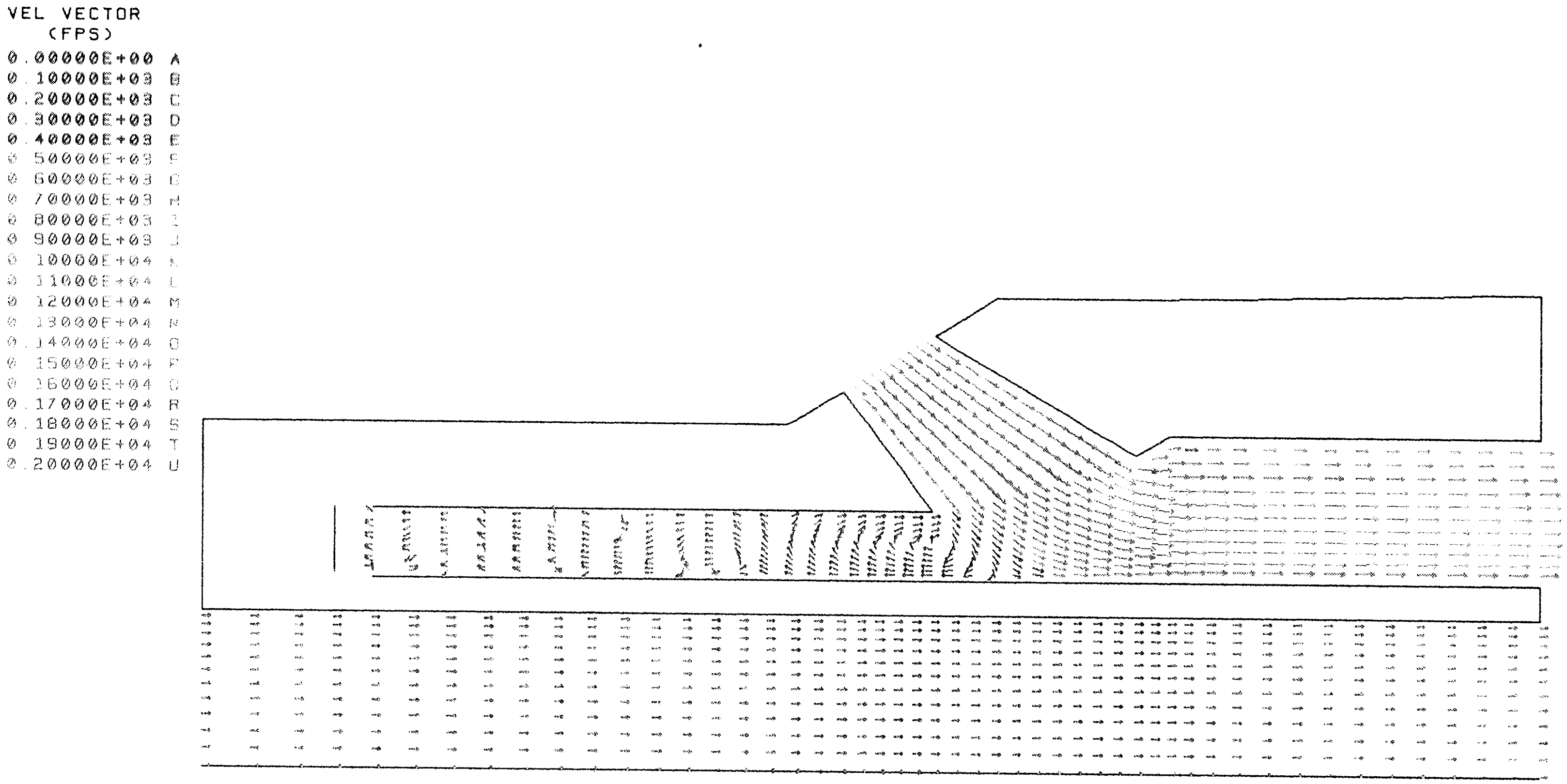


Fig. 9. VELOCITY VECTORS IN MAIN INJECTOR ELEMENT

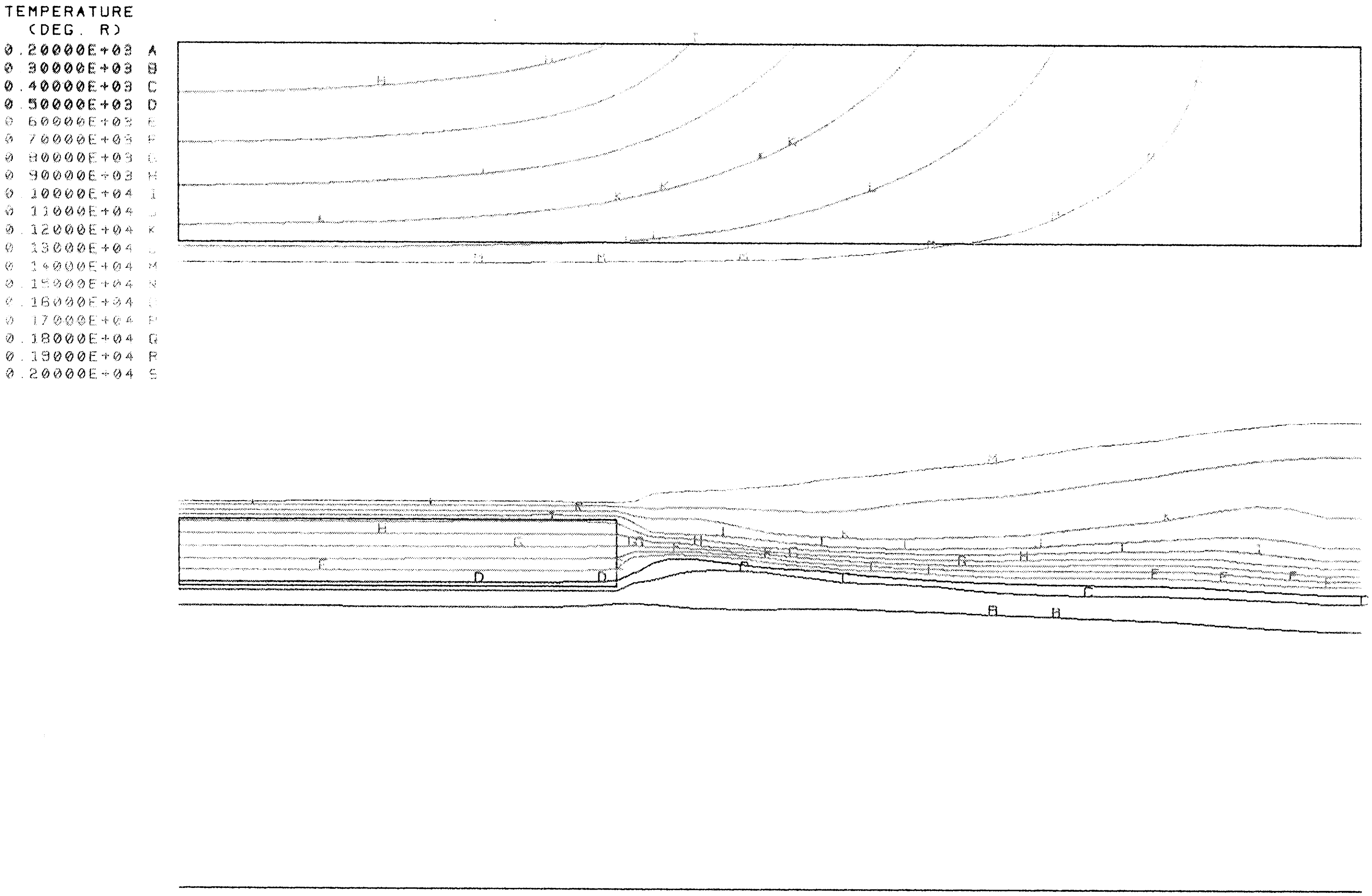


Fig. 10. TEMPERATURE IN MAIN INJECTOR ELEMENT EXIT

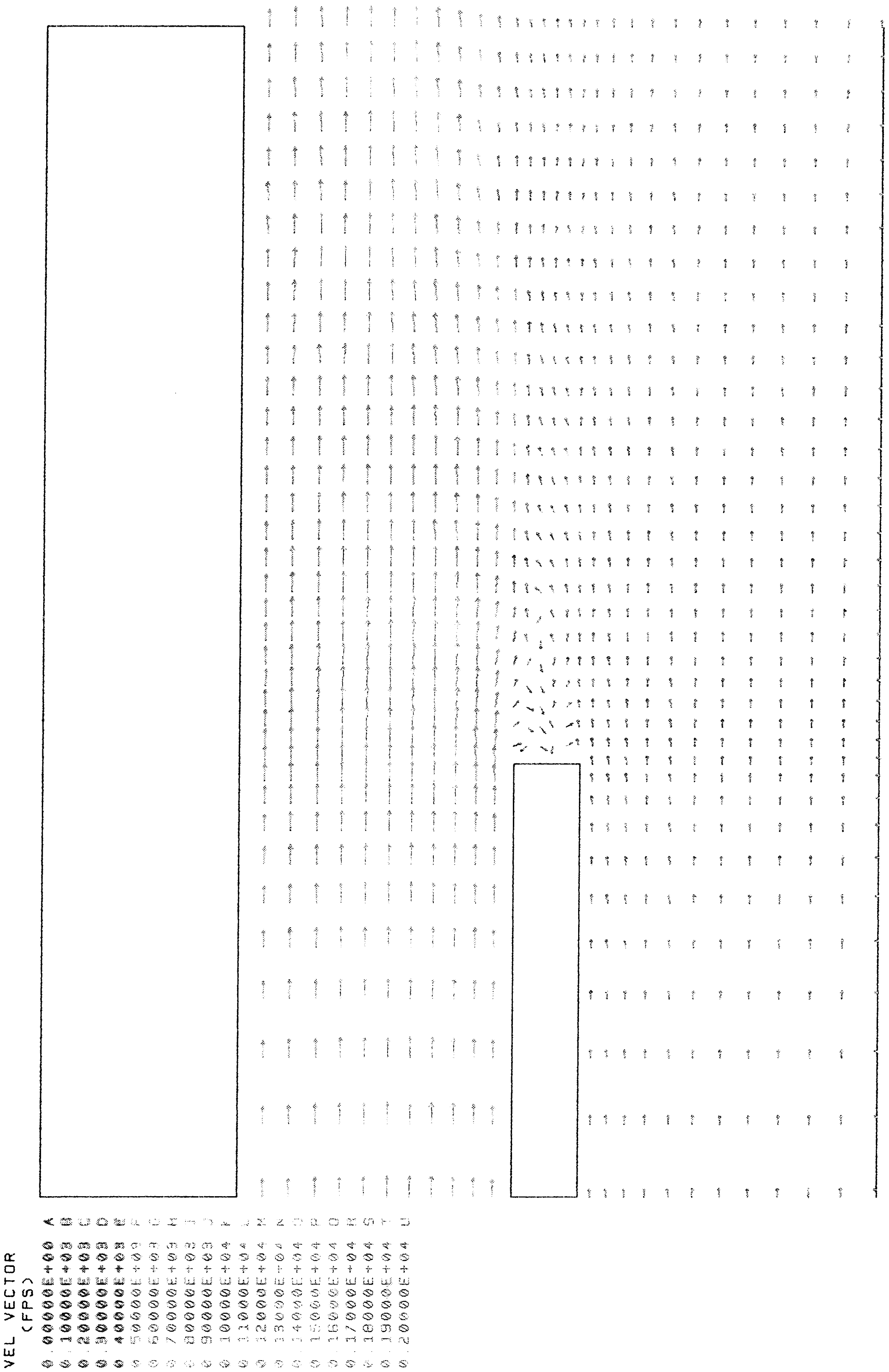


Fig. 11. VELOCITY VECTORS IN MAIN INJECTOR ELEMENT

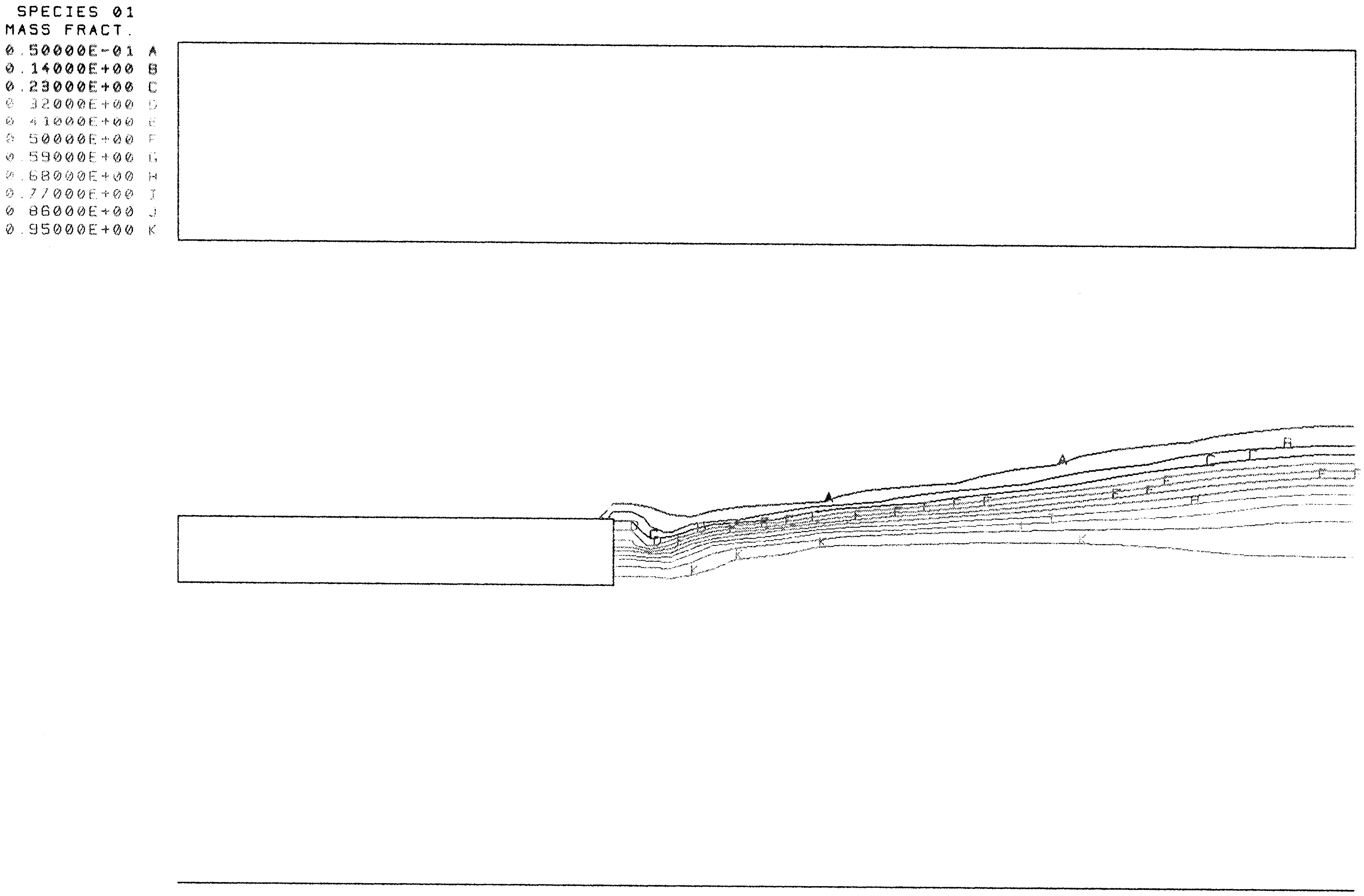


Fig. 12. OXYGEN MASS FRACTION IN MAIN INJECTOR ELEMENT EXIT

gases have been cooled from 1500 R to around 1440 R. As the hot gases mix with the cold oxygen, the hydrogen remains a "dense gas" but some of the steam condenses due to the high critical conditions of water. The results of this analysis provide a good approximation of the real fluid properties as they enter the main combustion chamber.

3.3 Baffle Injector Element

The analysis of the flow field and conjugate heat transfer of a typical baffle injector element was accomplished using the FDNS code. The analysis was performed in order to determine the effects of flow losses and heat transfer on the thermodynamic properties of the fluids as they traverse the long narrow elements and enter the combustion chamber. The results of this analysis will serve as input data for analyzing the combustion process in the chamber itself. The analysis includes the flow of oxygen from the LOX dome to the combustion chamber, the flow of coolant hydrogen gases from the hydrogen manifold to the chamber, the heat transfer from the environment surrounding the element, and the transfer of heat from the coolant hydrogen to the cold oxygen through the LOX post wall.

3.3.1 Geometry

There are 75 baffle elements located along 7 equally spaced radials in rows 7 through 13 and the entirety of row 6. The wide variety of configurations discussed for the main elements applies to the baffle elements as well. A composite baffle element has been synthesized based primarily on modifications M83 through M99 and is shown in Figure 13. The geometry has been included in the grid code discussed above. In addition to the reference lengths described previously, the tip of the LOX posts in all baffle elements are a reference distance of 11.590 inches from reference

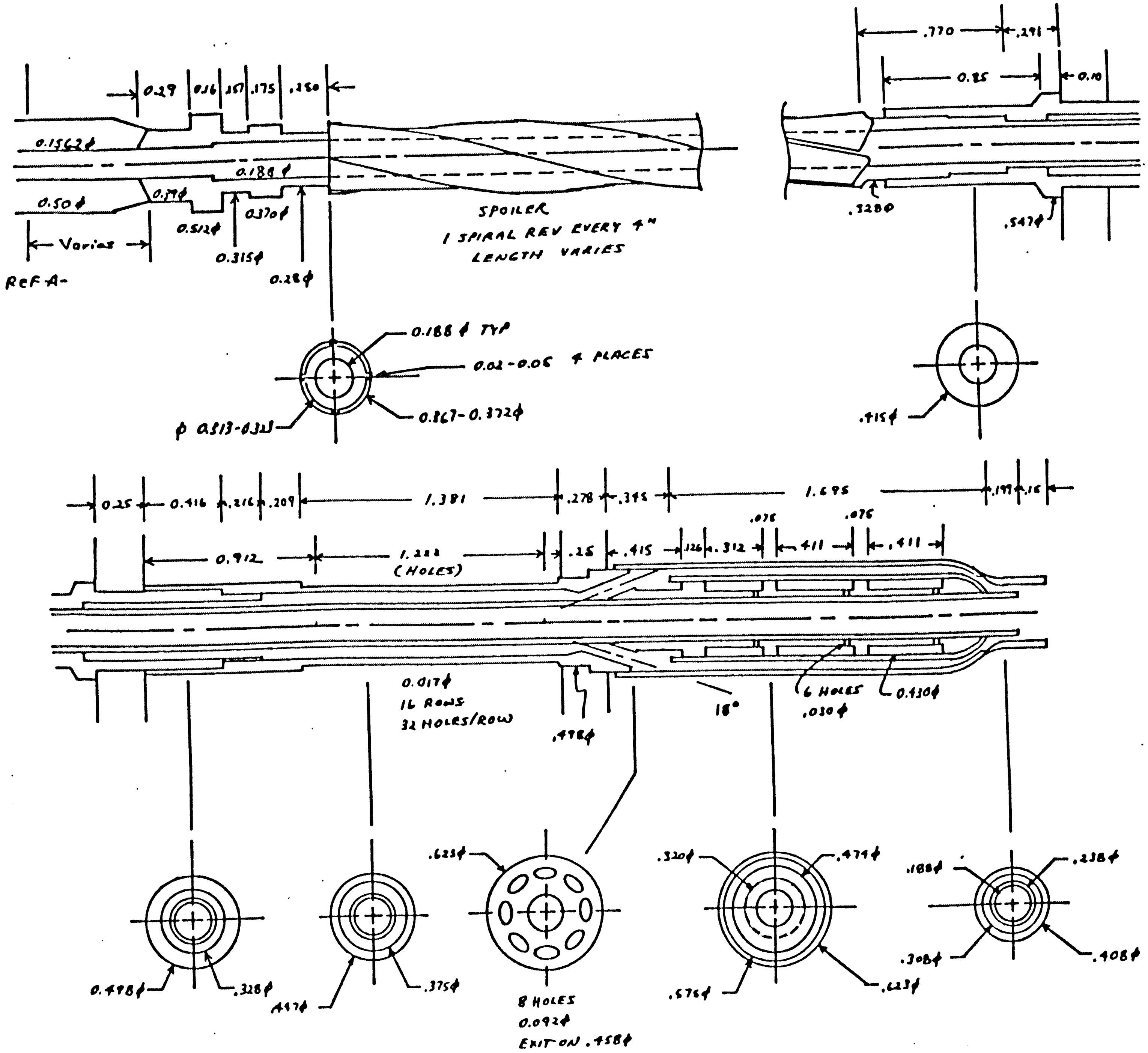


Fig. 13. Baffle Injector Element Configuration

plane -A- as shown on drawing RS009122. This reference is consistent with those used for the main elements, allowing for clustering of main and baffle elements from different rows consistently.

The baffle elements are more complicated than the main elements. The baffle elements protrude approximately 2.5 inches into the main combustion chamber, therefore the model must extend into the MCC. This extension consists of the LOX post surrounded by a jacket and a core (RS009226). The core has 8 equally spaced holes that pass hydrogen coolant into the area between the core and the jacket. These holes are modeled as slots of equal cross sectional area. The air spaces in the core are ignored. Some of the baffle elements have a tip extension (R0019527) attached to the jacket. This tip has been included in the model.

The retainer (RS009134) does not have holes in it, so hot exhaust gases do not enter the elements. The sleeve between the injector plates (RS009226) is perforated with 16 rows of holes with 32 holes per row, through which hydrogen coolant enters the element. This sleeve was modeled as a slot with the same flow area as the sum of the holes.

As with the main injector element, the baffle element was originally modeled 3-D, but because of the large number of nodes required, the element was modeled axisymmetrically. The grid for the baffle element contains 10,404 nodes and is represented in Figure 6. There are 13 nodes across the oxygen flow path, 6 nodes across the LOX post wall, 13 nodes across the hydrogen flow path, 7 nodes across the outside wall of the element, and 306 nodes along the axis.

3.3.2 Fluid Properties

The fluid properties model discussed above was used in the analysis of the baffle element.

3.3.3 External Wall Temperature

The external wall temperature analysis discussed above was also used in the baffle element analysis to obtain skin temperatures along the element.

3.3.4 Fluid Flow and Heat Transfer Analysis

The analysis of the flow field and heat transfer for the baffle element was completed using the modified FDNS code discussed above.

The properties of the oxygen as it entered the LOX posts for the baffle elements were the same as for the injector elements. Coolant hydrogen properties were approximated at a temperature of 465 R and a pressure of 3580 PSIA, yielding a density of 1.2298 LBM/FT3. An hydrogen flowrate of 19.3 LBM/SEC, divided by the density and the cross sectional area of the 75 sleeves, resulted in a inflow velocity of 17.558 FPS.

The wall conductance analysis resulted in skin temperatures of 866 R in the exhaust manifold near the LOX dome, 941 R in the manifold near the secondary plate, 465 R in the hydrogen coolant manifold, and was estimated at 1500 R in the chamber along the baffle. This relatively cool temperature was assumed due to the cool hydrogen flowing through the porous primary plate.

The results of the analysis are presented in Figures 14 through 18. The oxygen remains a liquid as it emerges from the

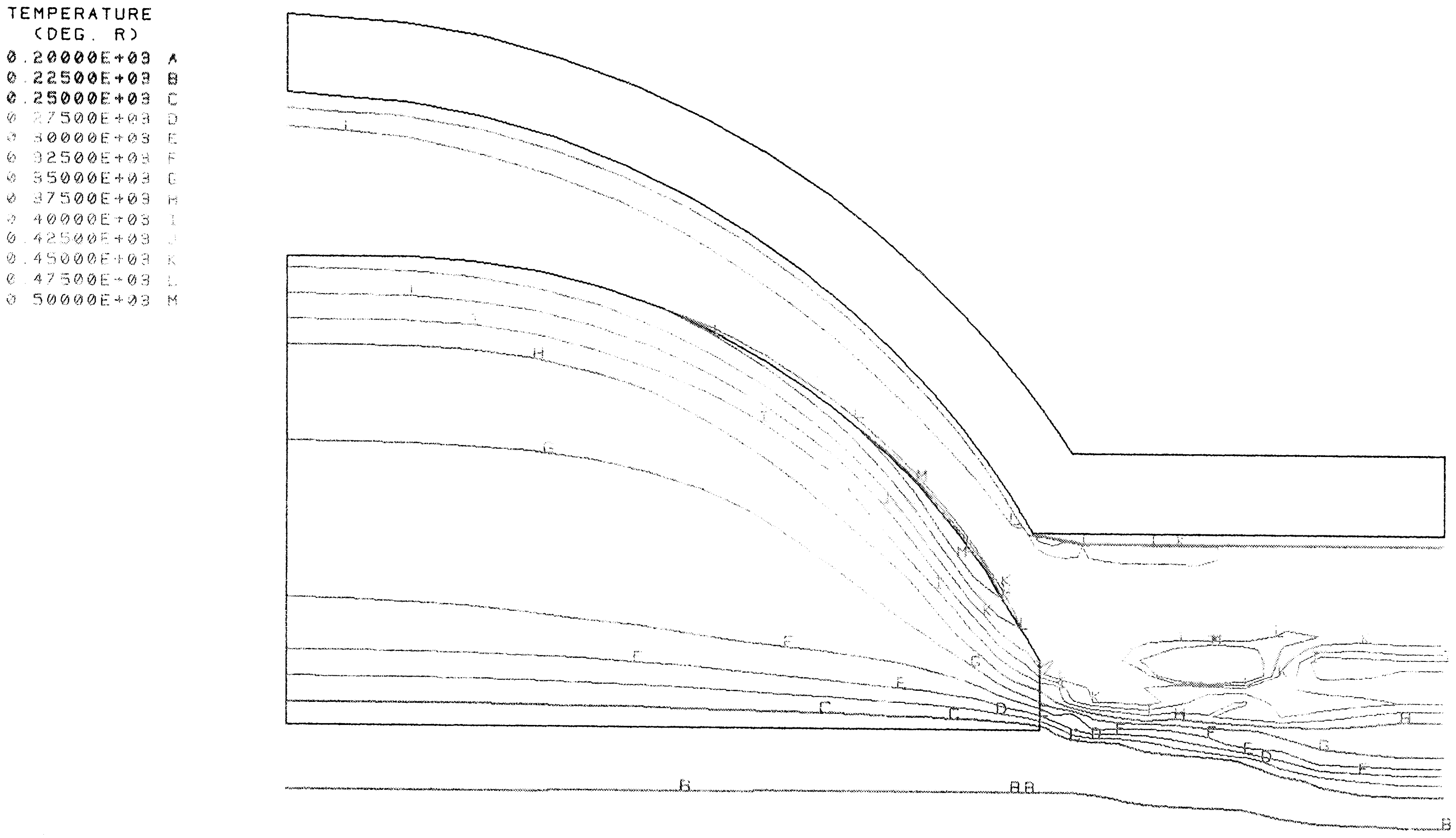


Fig. 14. TEMPERATURE IN BAFFLE ELEMENT EXIT

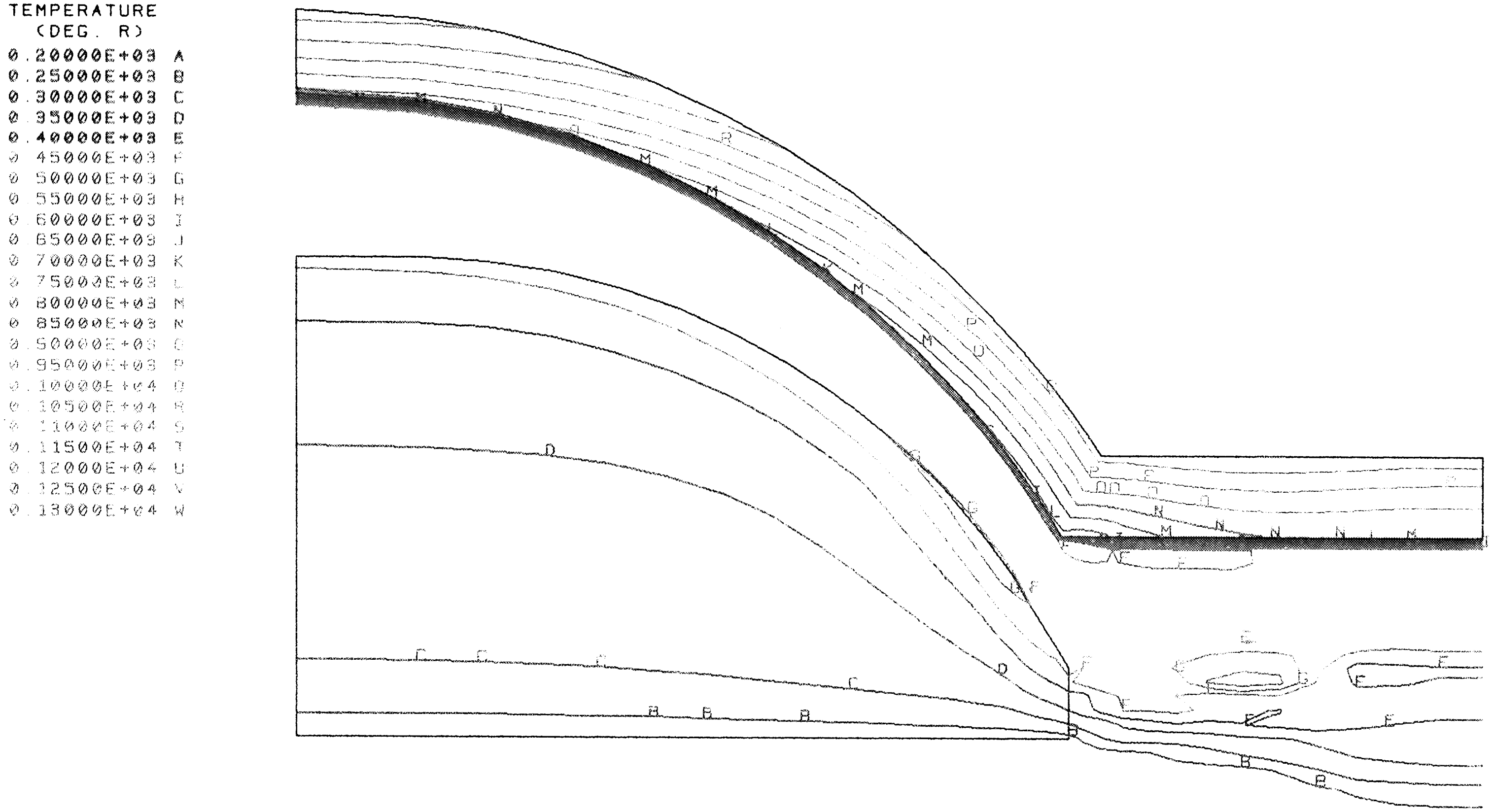


Fig. 15. TEMPERATURE IN BAFFLE ELEMENT EXIT

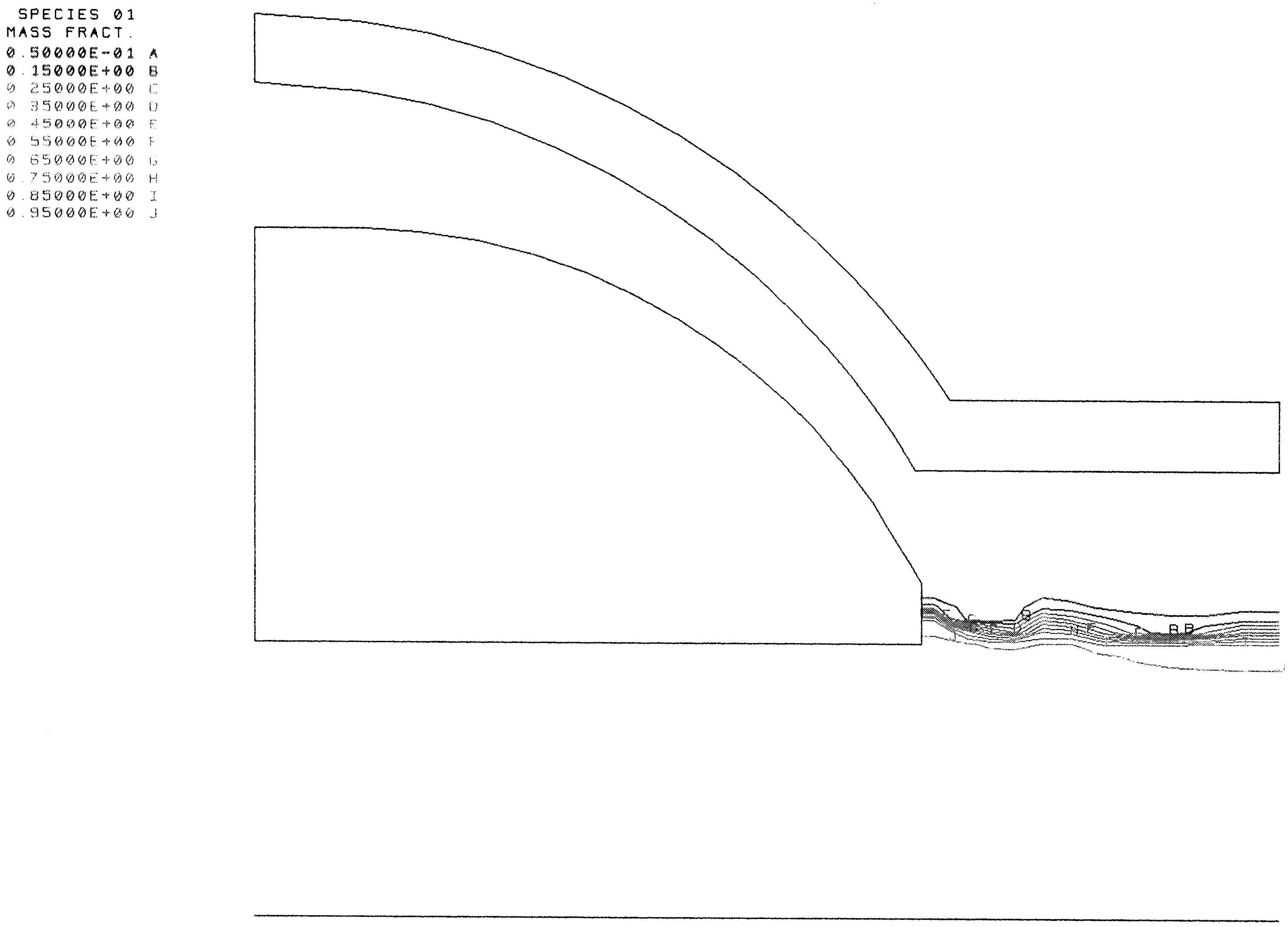


Fig. 16. OXYGEN MASS FRACTION IN BAFFLE ELEMENT EXIT

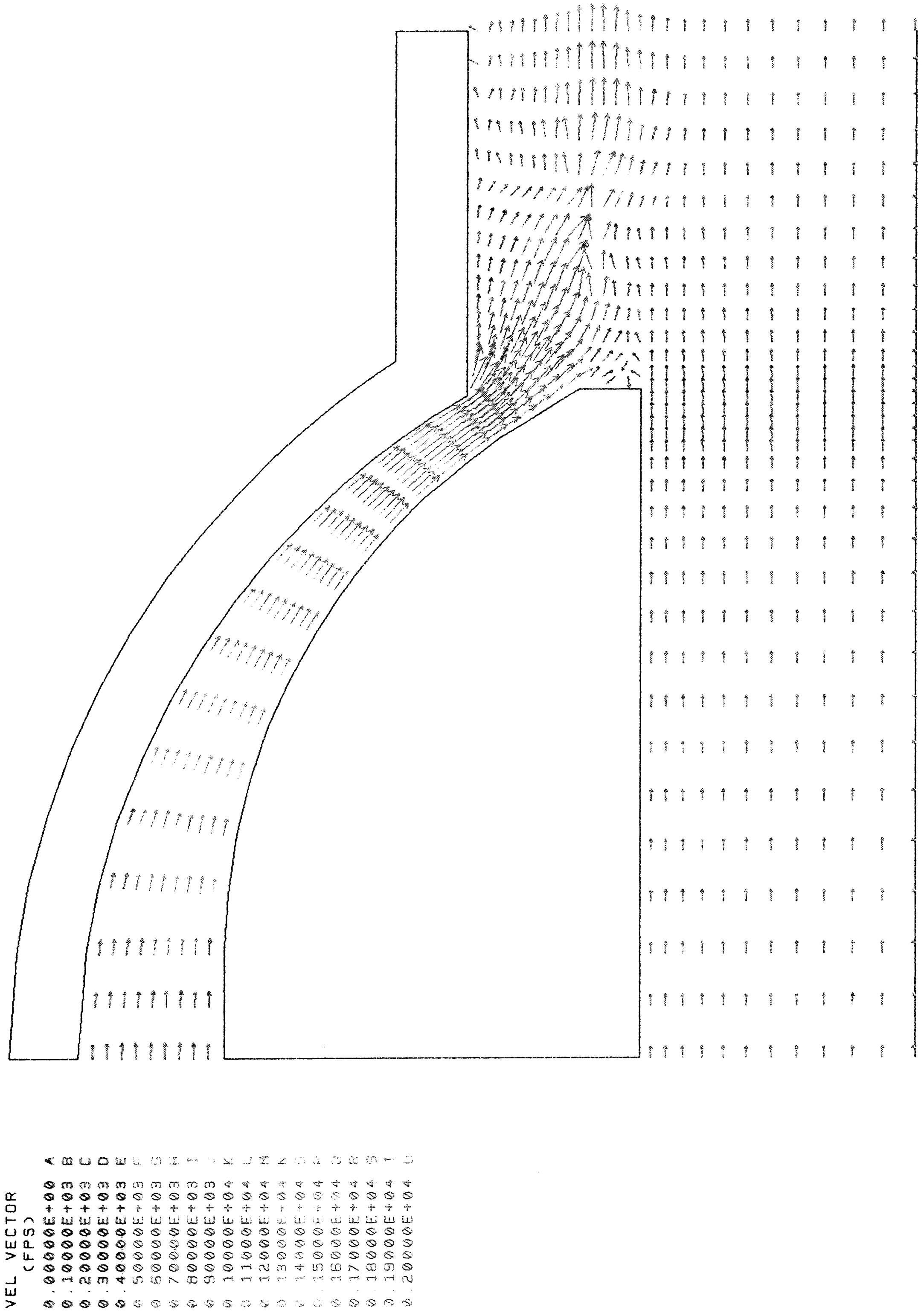


Fig. 17. VELOCITY VECTORS IN BAFFLE ELEMENT EXIT

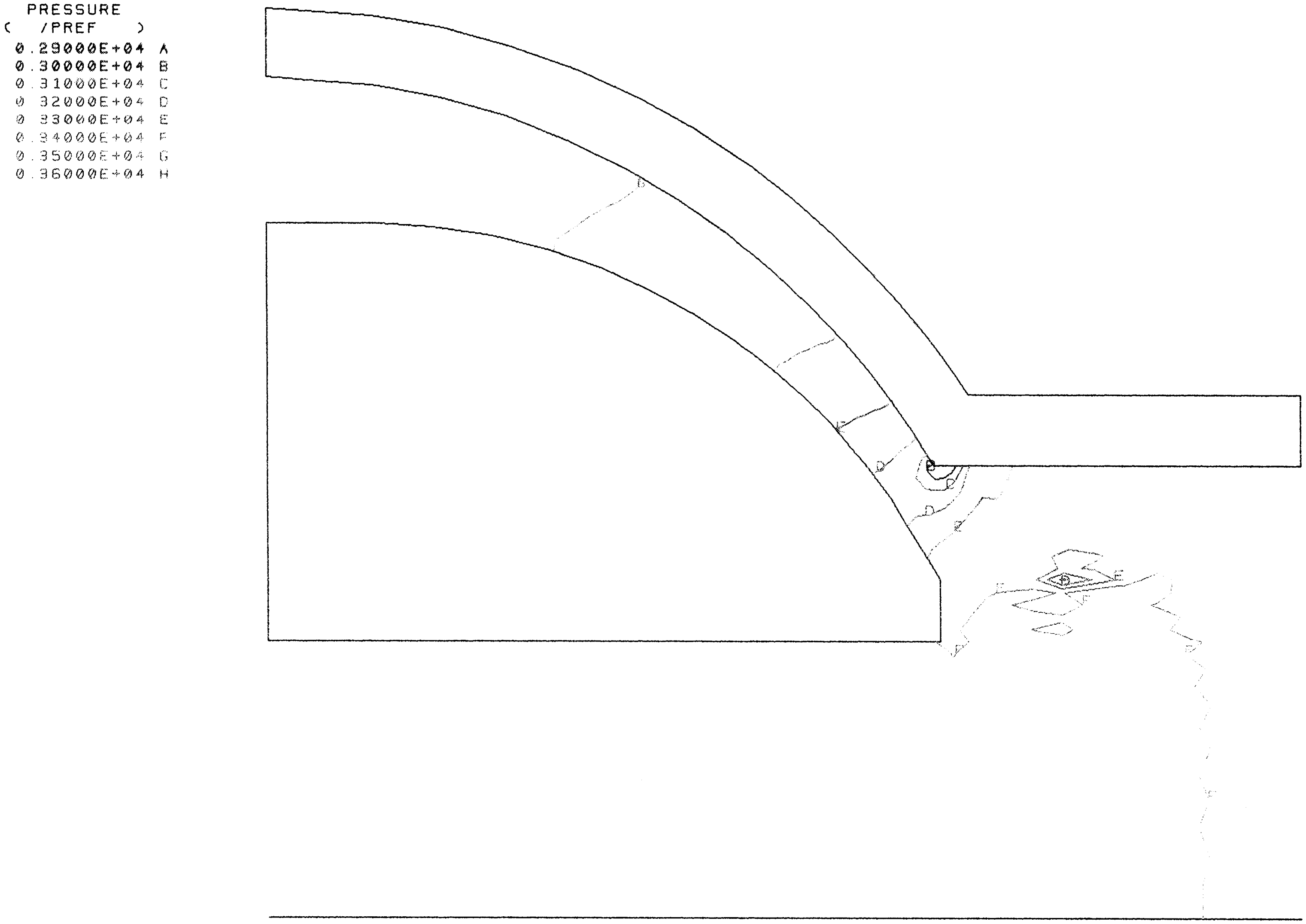


Fig. 18. PRESSURE IN BAFFLE ELEMENT EXIT

LOX post, reaching temperatures of 223 R to 230 R. The lessened heating, compared to the main injector element, is due to the presence of coolant hydrogen with a temperature range of 465 R to 450 R. Again, this analysis provided a good approximation to the fluid properties as they leave the baffle element and enter the combustion chamber.

3.4 Main Combustion Chamber Streamtube

The results of the main injector element flow and heat transfer analysis was used as upstream boundary conditions for a combustion chamber streamtube flow analysis. An analysis of a streamtube around the baffle element will not be performed since the baffle elements are to be eliminated. This streamtube analysis will be performed to investigate the combustion process as the fluids enter the combustion chamber. Since the outlet boundary conditions used in the main injector element analysis probably affected the computation of the mixing region, the streamtube analysis begins slightly upstream of the LOX post exit plane and, therefore, includes all of the mixing region. In addition to the fluids entering the streamtube from the main injector element, coolant hydrogen bleeds through the porous injector plate and enters the streamtube.

3.4.1 Geometry

The geometry for this analysis starts slightly upstream of the LOX post exit plane and consists of a streamtube which extends 5 inches downstream of the primary injector plate. The streamtube cross sectional area was determined by evaluating the relative proximity of the elements in rows 12 and 13. The resultant cross sectional area was slightly less than 1/600 of the chamber cross section, but was deemed appropriate for row 13. The cross sectional area, less the element exit plane area, determines the primary injector plate flow area for the flow of hydrogen through the porous plate into the streamtube. The grid representing this geometry consists of 13,920 nodes. There are 13 nodes across the LOX post exit plane, 6 nodes across the LOX post wall, 15 nodes across the hydrogen flowpath, 9 across the element wall, and 9 across the injector plate where the hydrogen enters the streamtube. There are 290 nodes in the axial direction. The inlet portion of the grid is shown in Fig. 19.

3.4.2 Fluid Properties

The HBMS fluid model discussed previously was used in this analysis. The combustion model used in the analysis, to be discussed later, involves the radicals O, H and OH. Critical properties for these radicals are not available. Work at the US Army Ballistic Research Laboratory has resulted in the development of a real gas chemical equilibrium code, BLAKE, which treats radical species (Ref. 20). This code uses a virial equation of state and molecular models to define fluid properties. Unfortunately, a technical manual is not available for describing the details of this modeling procedure. The BLAKE code has been obtained by SECA, but the program would have to be decoded to determine how the radical species are described. Since the accuracy of these virial equations to represent these radials is not known, further investigation did not seem warranted and these species were modeled as ideal gases.

3.4.3 Combustion Model

The combustion process was simulated using a finite-rate global reaction for the formation of water from diatomic hydrogen and oxygen (Ref. 21), and three reactions for the formation of the radicals O, H, and OH. These reactions are:

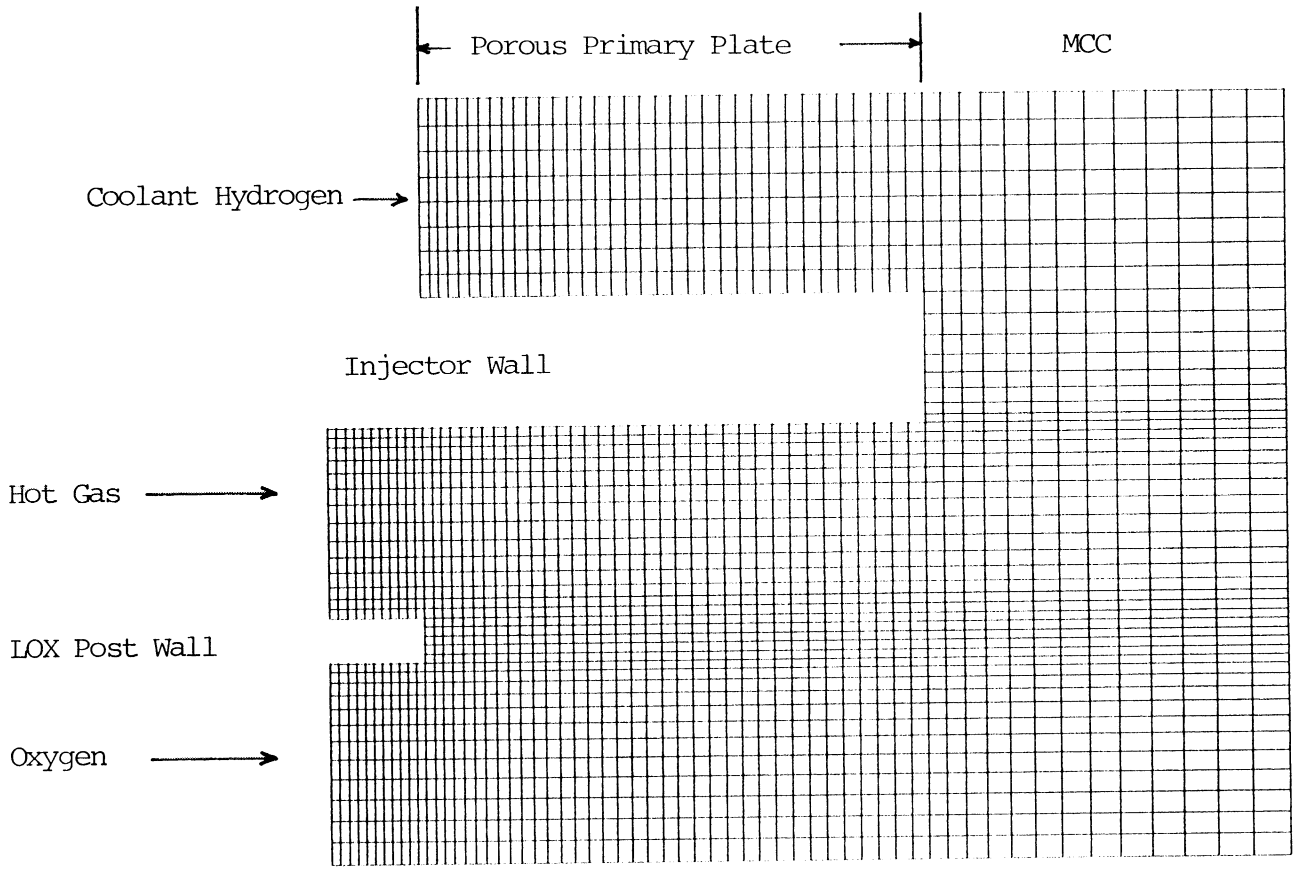


Fig. 19. Inlet Geometry for Combustion Chamber Streamtube

$2H + O_2 = 2H_2O$ (global reaction)

$O + O + M = O_2 + M$

$H + H + M = H_2 + M$

$H + OH + M = H_2O + M$

where M represents a third body. Reaction rate constant data for the global reaction was obtained from Reference 21, but the reaction rate was calculated using the square root of the stoichiometric coefficients. Reaction rate constant data for the last 3 reactions were obtained from Ref. 22.

The HBMS fluids model, which tracks the quality of each species, will directly account for the evaporation of liquids, assuming thermal equilibrium between phases. Since only gas phases are involved in chemical reactions, the quality of each species has been included in the reaction rate calculations discussed above.

The FDNS code has been modified to solve the finite-rate chemistry implicitly. This involves expanding the species production rate equations to obtain an implicit form of the equations, then solving the resulting 6 X 6 matrix to obtain implicit production rates for each species. These implicit production rate terms replace the explicit production rate terms in FDNS. This modification to the FDNS code has been completed and verified on related problems.

The streamtube was first computed without combustion, using the modified FDNS code, in order to establish a flow field with which to initiate combustion.

The combustion process was originally planned to be initiated by assuming equilibrium combustion at the exit plane of the streamtube (5 inches into the chamber), and letting the flame front move upstream. Equilibrium concentrations of the global species and the radicals in the exit plane would be computed using equilibrium constants for the 4 reactions. The coding of this equilibrium condition at the outflow boundary was completed and verified by comparing to CEC program results. However, the flame front progressed upstream too slowly, so this procedure was abandoned.

Another procedure was attempted whereby finite rate chemistry was employed throughout the flow field and ignition was initiated by gradually increasing the exit plane temperature. Problems were encountered in propagating this elevated temperature upstream because the FDNS code does not propagate, via diffusion, downstream boundary conditions.

It was found that the high temperature of the hot exhaust gases as they entered the combustion chamber (1400 - 1500 R) was sufficient to cause the global reaction to ignite. This reaction can also be accelerated by using an artificial elevated temperature in the calculation of the reaction rate.

3.4.5 Fluid Flow and Heat Transfer Analysis

The upstream boundary for the streamtube analysis consists of: (1) oxygen and hot exhaust gas flow properties from the main injector analysis, (2) wall temperatures in the LOX post wall and the injector element wall, and (3) coolant hydrogen bleeding through the porous primary injector plate. The coolant hydrogen enters the porous plate at a temperature of 465 R and a pressure of 3584 PSIA. The total hydrogen flowrate, for the entire MCC, was 5.29 LBM/SEC, resulting in an inlet velocity of 11.9 FPS. The hydrogen emerged from the MCC side of the primary plate at an assumed fixed temperature of 666 R. The downstream pressure was calculated by the FDNS code and the downstream density from the fluids model. The velocity of the hydrogen leaving the primary plate was calculated by mass conservation, typically 18.5 FPS.

Although the wall temperatures had been computed in the main injector analysis, this process was continued in this analysis.

The results of the analysis are presented in Figs. 20-28. Due to the high aspect ratio (length to diameter), the results are presented in 3 sets of figures. Figures 20-22 present the results in the injector and 1 inch into the chamber. Figure 23-25 are for 1 to 3 inches into the chamber and Figs. 26-28 are for 3 to 5 inches into the chamber.

Figure 20 shows some vortices just downstream of the LOX post which may be real or may be caused by a slight instability as the oxygen passes near critical temperature. Figure 21 shows a temperature spike resulting when steam passes near its critical temperature. Fluid properties near the critical point are not accurate and may be creating instabilities.

Figure 27, showing the temperature distribution near the end of the streamtube, indicates that mixing is not complete, as it should be. This is probably due to employing a straight, constant area streamtube. The actual chamber starts to converge, and turn the flow sharply toward the chamber axis, immediately downstream of the primary injector plate. This convergence would obviously enhance mixing.

The maximum temperature obtained was 7000 °R, slightly higher than the predicted chamber temperature of 6650 °R. This is probably due to the global reaction rate for the formation of water being slightly too high.

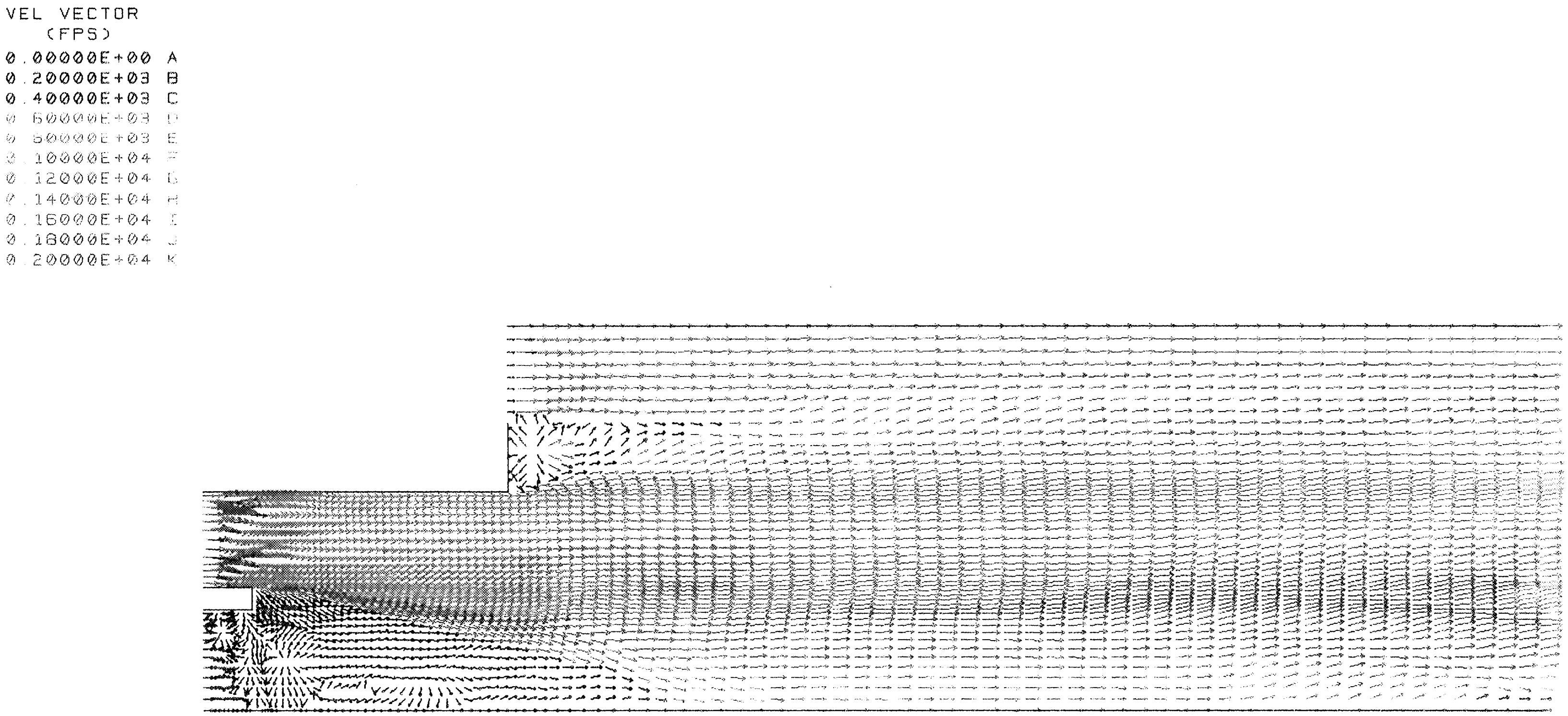


Fig. 20. VELOCITY VECTORS FROM INJECTOR TO ONE INCH INTO CHAMBER

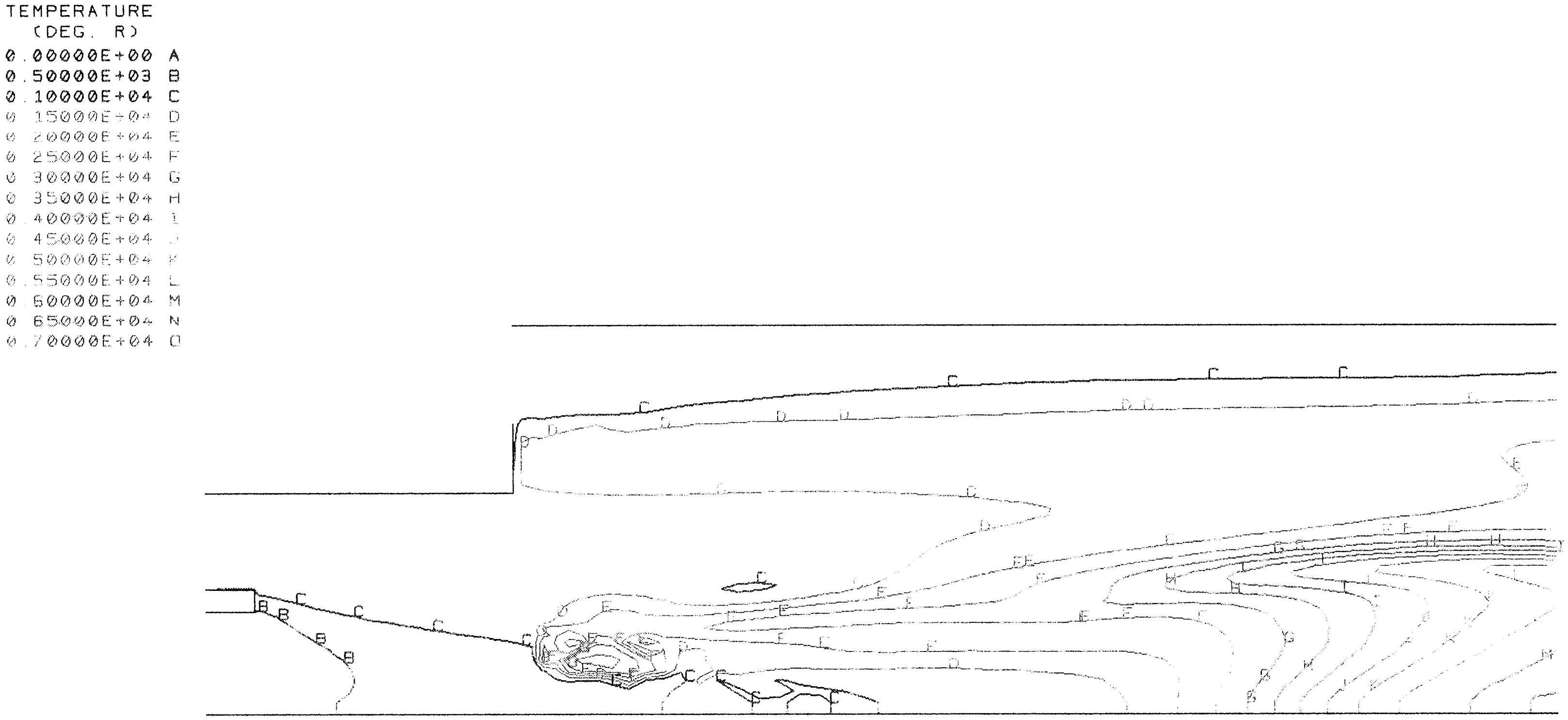


Fig. 21. TEMPERATURE FROM INJECTOR TO ONE INCH INTO CHAMBER

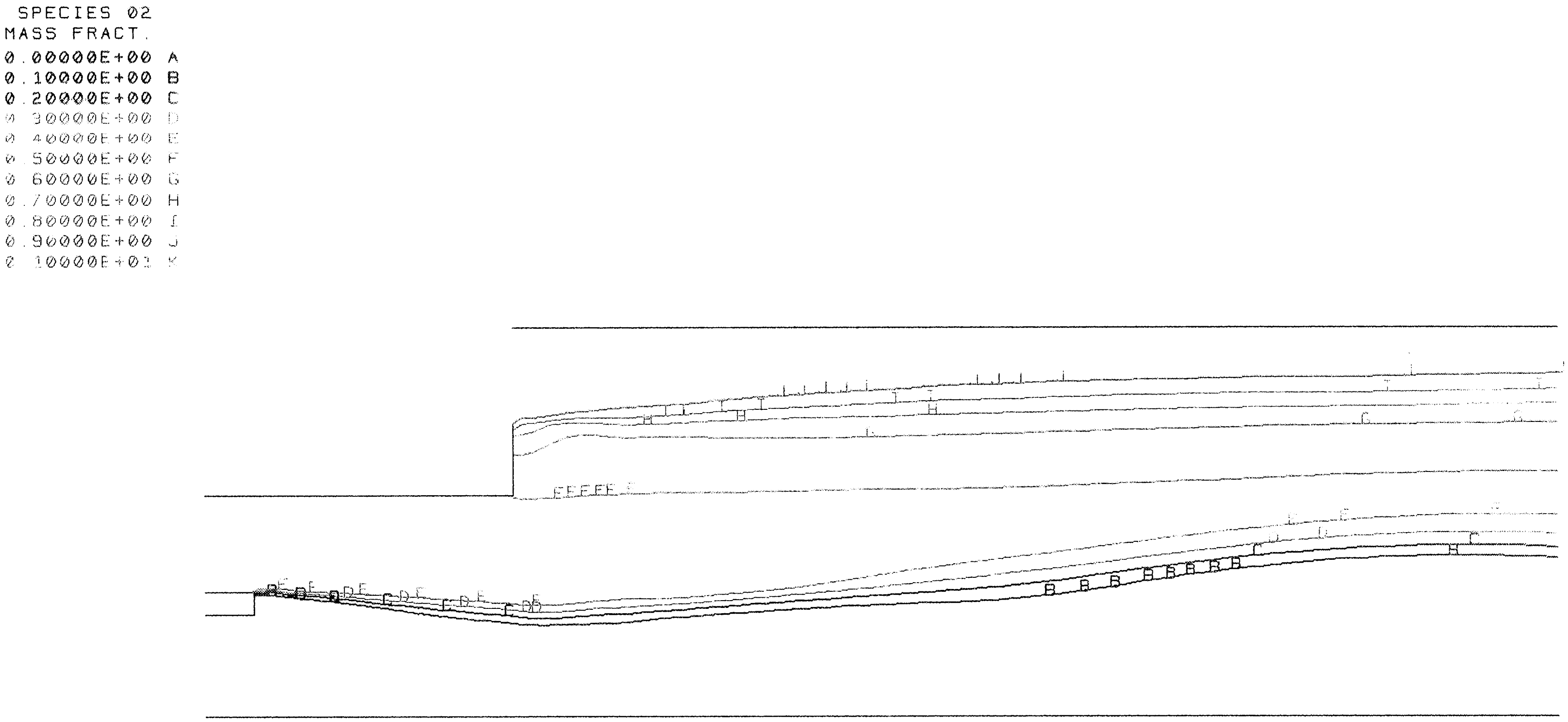


Fig. 22. HYDROGEN MASS FRACTION FROM INJECTOR TO ONE INCH INTO CHAMBER

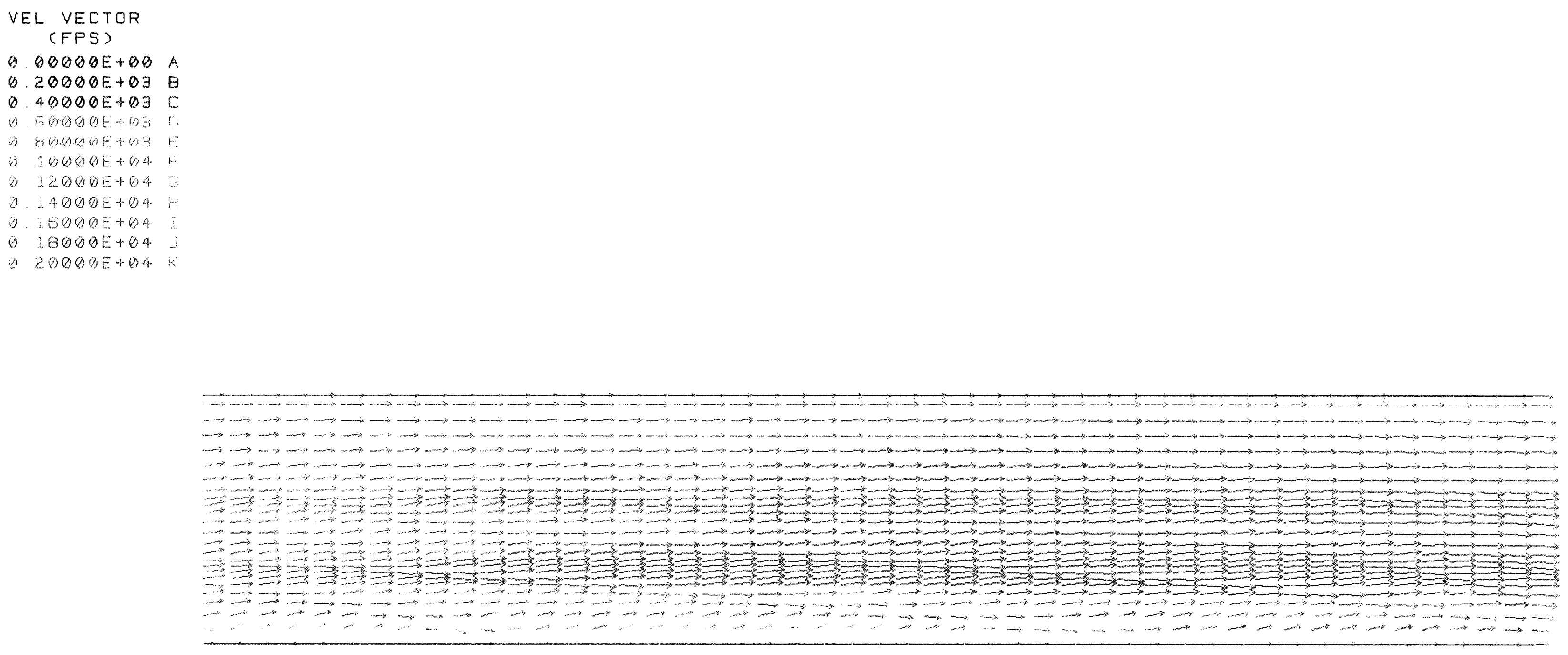


Fig. 23. VELOCITY VECTORS FROM ONE TO THREE INCHES INTO CHAMBER

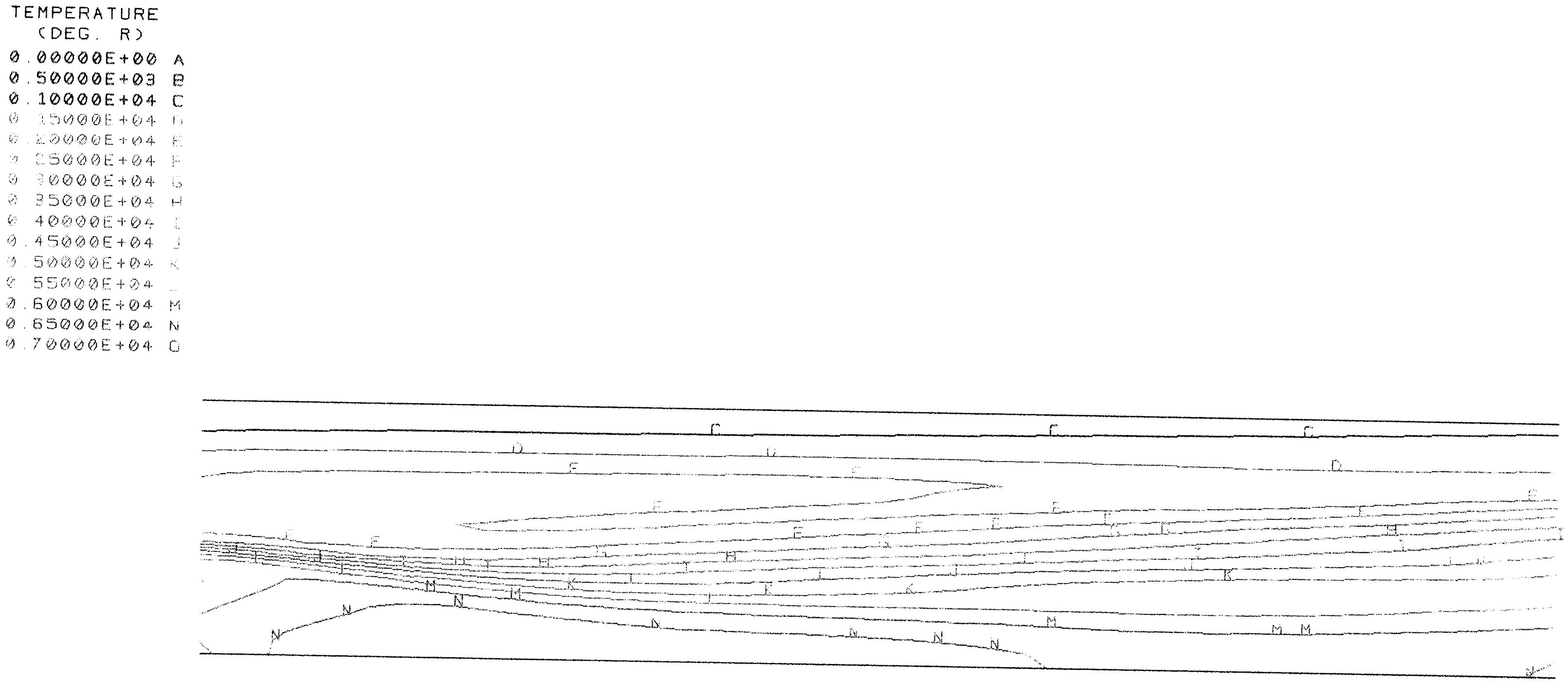


Fig. 24. TEMPERATURE FROM ONE TO THREE INCHES INTO CHAMBER

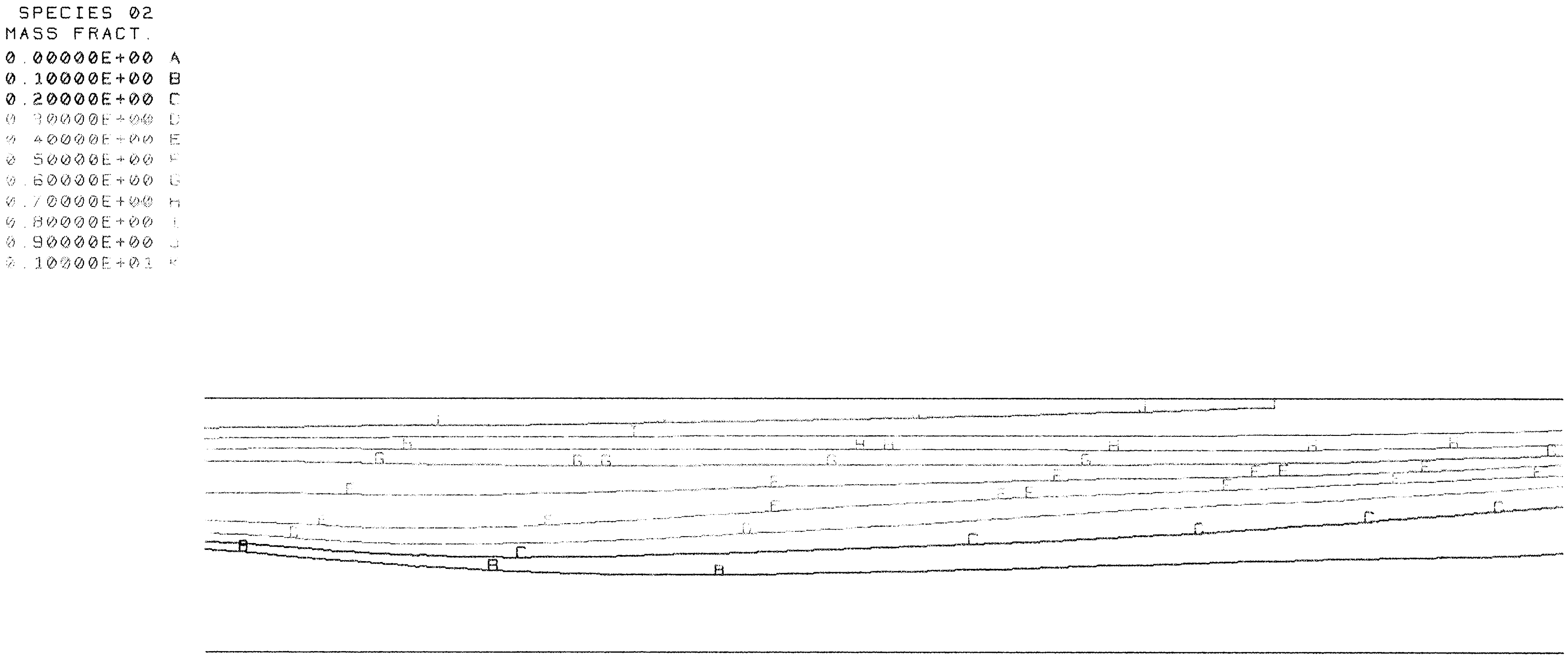


Fig. 25. HYDROGEN MASS FRACTION FROM ONE TO THREE INCHES INTO CHAMBER

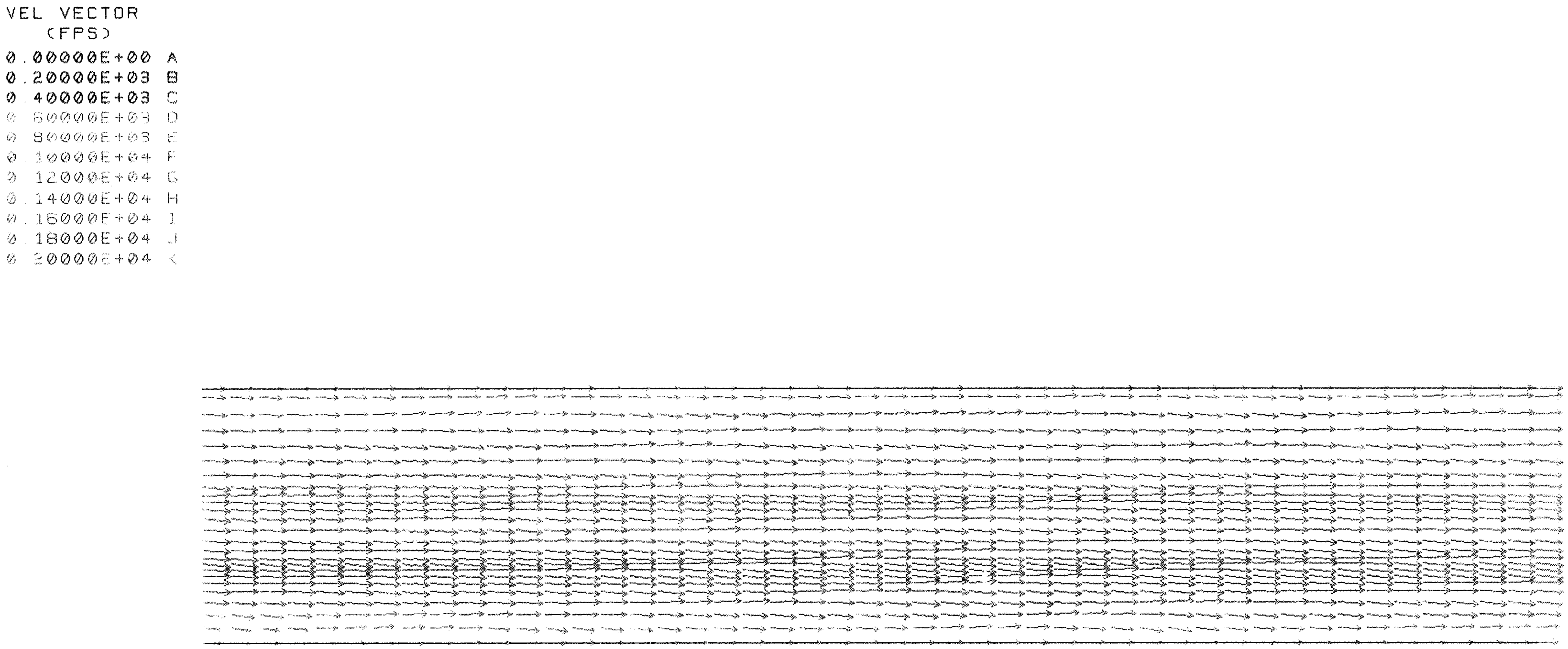


Fig. 26. VELOCITY VECTORS FROM THREE TO FIVE INCHES INTO CHAMBER

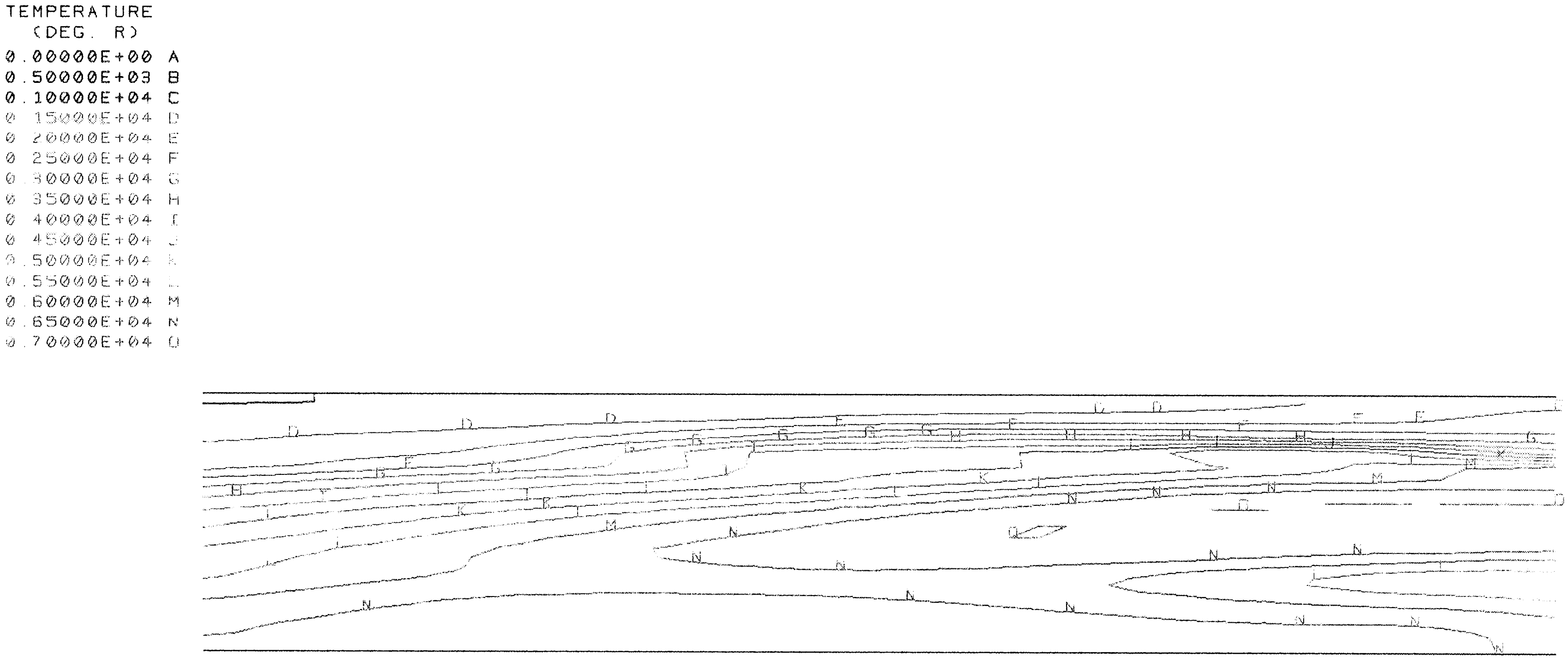


Fig. 27. TEMPERATURE FROM THREE TO FIVE INCHES INTO CHAMBER

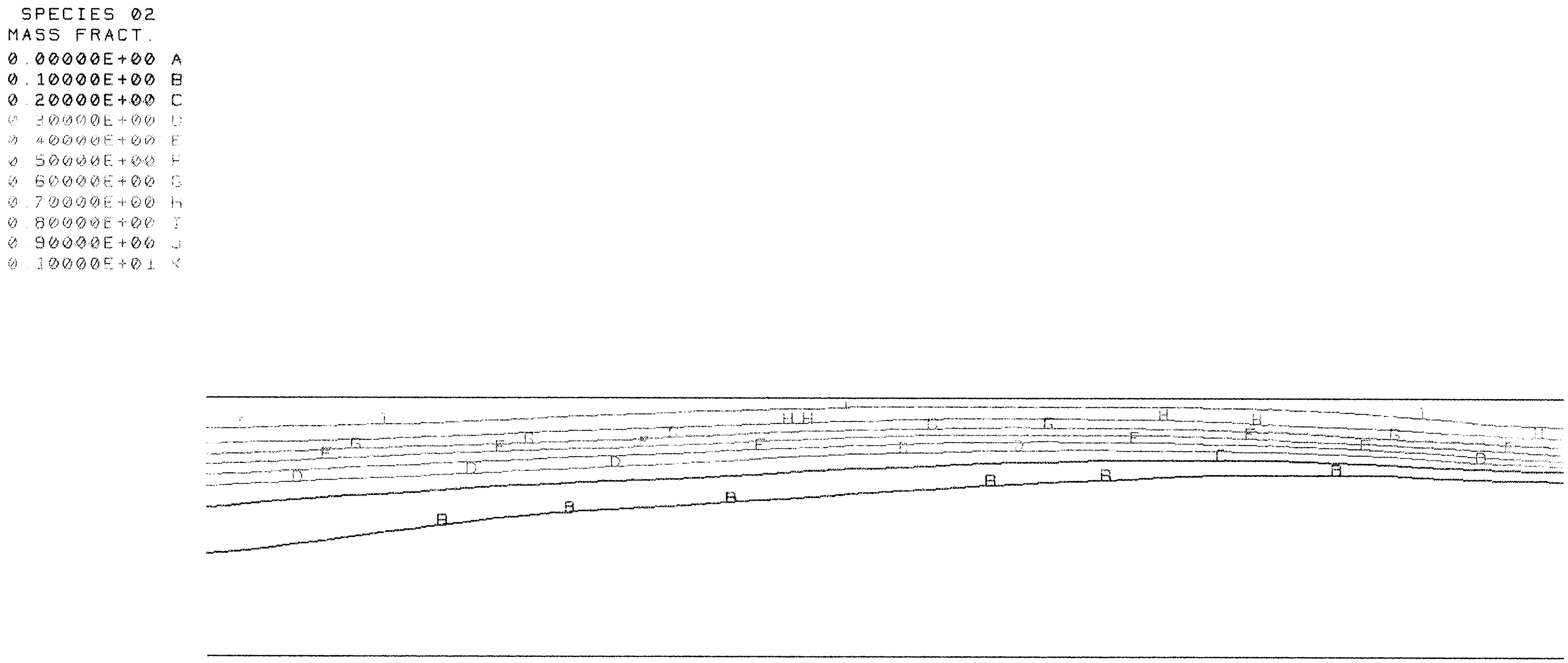


Fig. 28. HYDROGEN MASS FRACTION FROM THREE TO FIVE INCHES INTO CHAMBER

4.0 OVERALL ENGINE HEAT TRANSFER

Three predictions of heat transfer along the chamber wall of rocket motors all the way back to the injector face have been reported (Refs. 23-25). These are shown in Figs. 29-31. All three of these analyses show the expected reduced heat transfer near the injector face; however, method one attributes this reduction entirely to film cooling, method two to combustion kinetics, and method three to finite-rate vaporization. The difficulty with using method three is that an initial gas phase mixture-ratio and temperature must be specified; methodology to provide these boundary conditions is non-existent. If the physics reported with methods one and two is correct, a detailed spray vaporization model is not necessary to describe wall heating for these test conditions. These investigators did not find these simulations and analyses convincing, therefore the following analyses were performed to verify the overall engine heat transfer model developed herein.

Two sets of test cases were identified to serve as validation cases for the conjugate heat transfer model for liquid rocket motors; the 40k subscale STME experiments conducted by Pratt & Whitney, and the motors studied by Rocketdyne for the LOX/Hydrocarbon Thrust Chamber Technology Program (Ref. 26). Both sets of tests varied the predominately radial O/F distribution over the face plate and measured wall heat fluxes along the main combustion chamber and nozzle walls. Although the initial flow from individual injectors is 3-dimensional, the major part of the flowfield can be well simulated with an axisymmetric model. Details of these two studies are described below.

A third set of test data serves as further validation of the conjugate heat transfer model to predict film cooling. A final

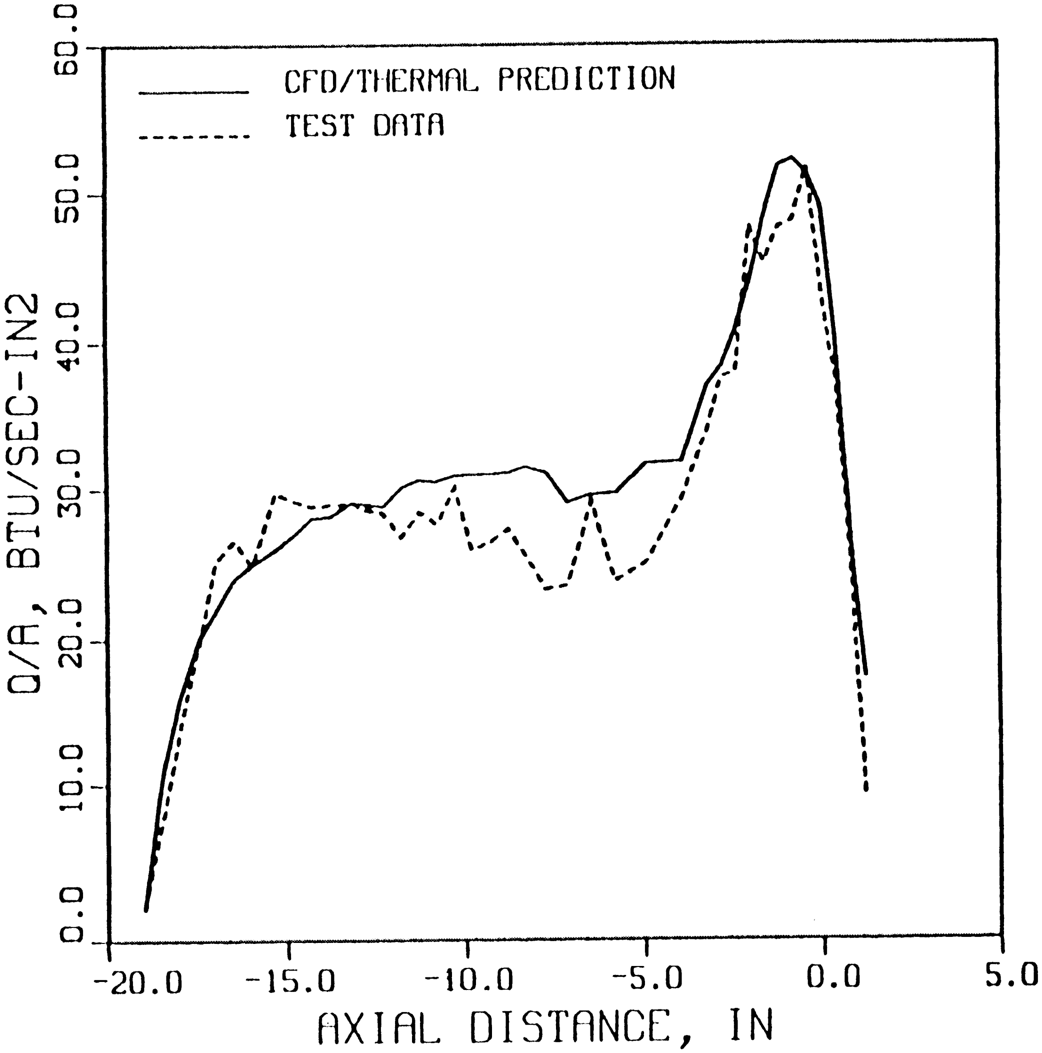


Fig. 29. Heat Transfer in the Subscale NLS Engine

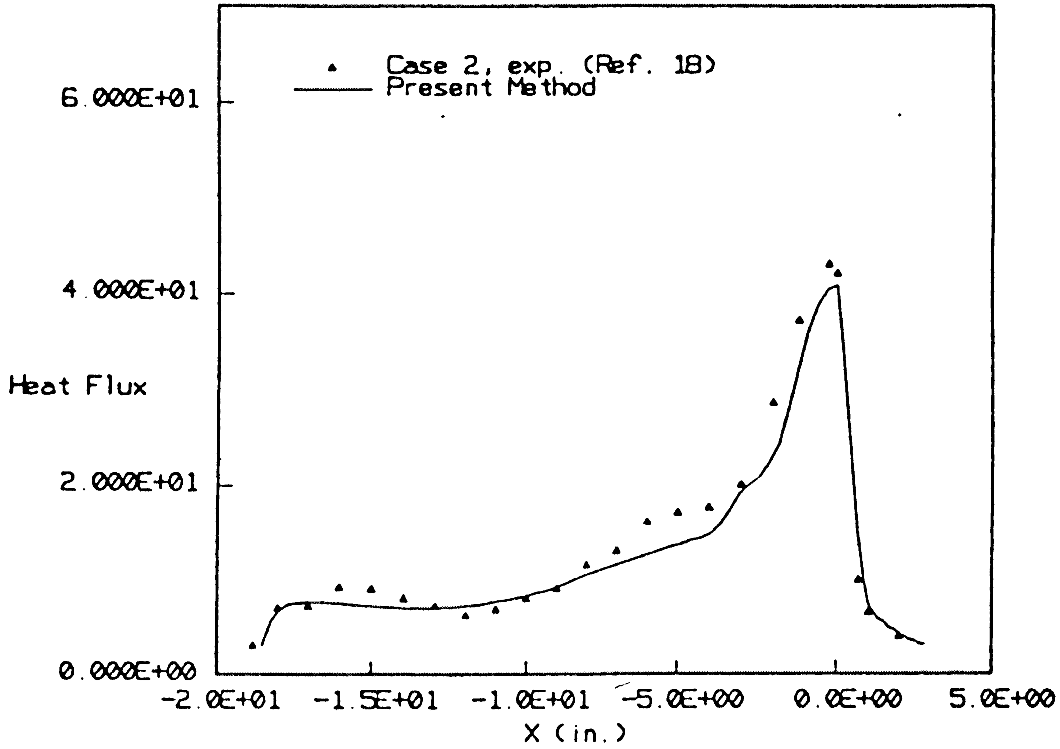


Fig. 30. Wall Heat Flux (in Btu/in^2-s) Comparisons for the Rocketdyne Hydrocarbon Thrust Chamber Test Motor

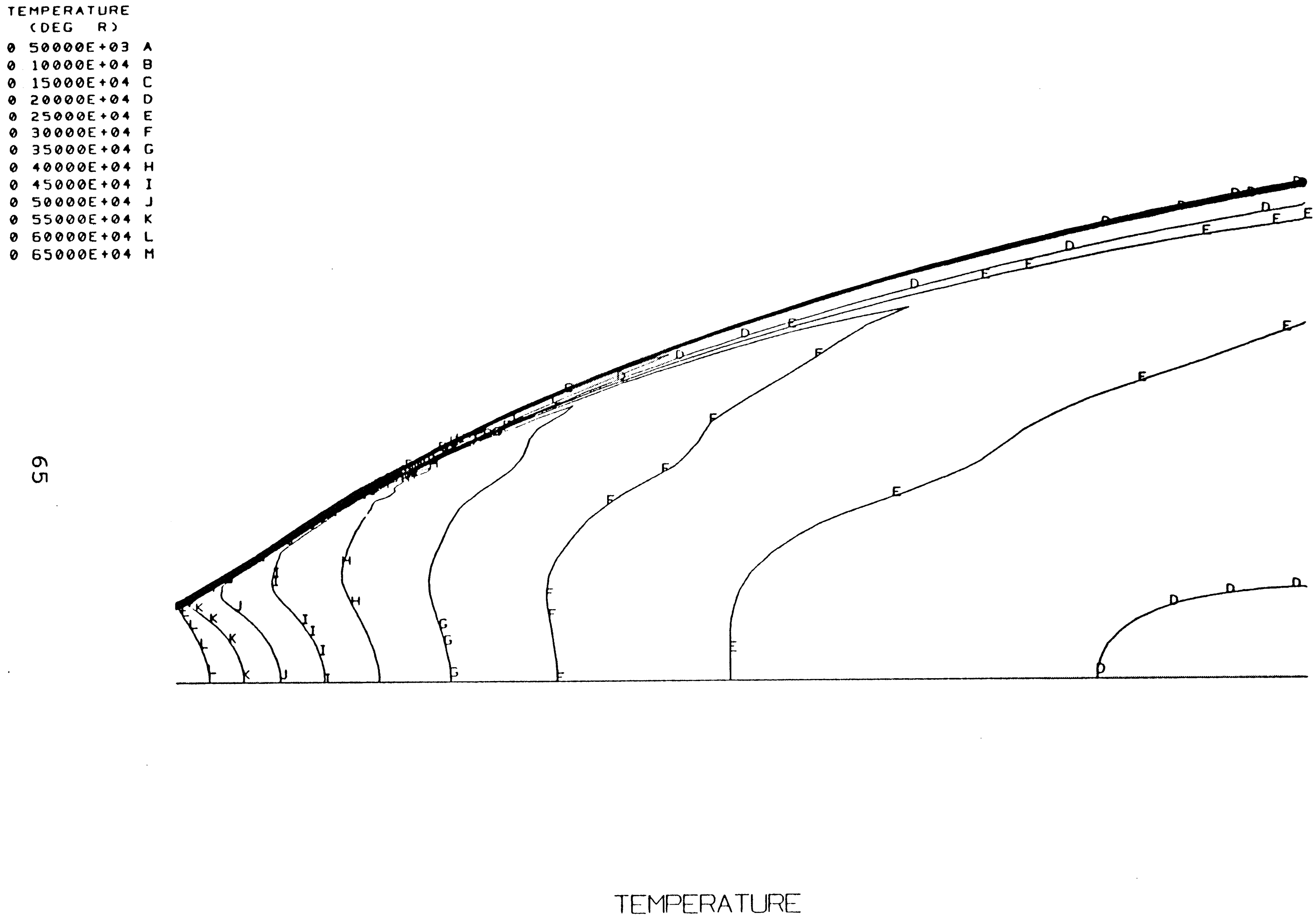


Fig. 34. Subscale STME Nozzle Flow - Temperature Contours

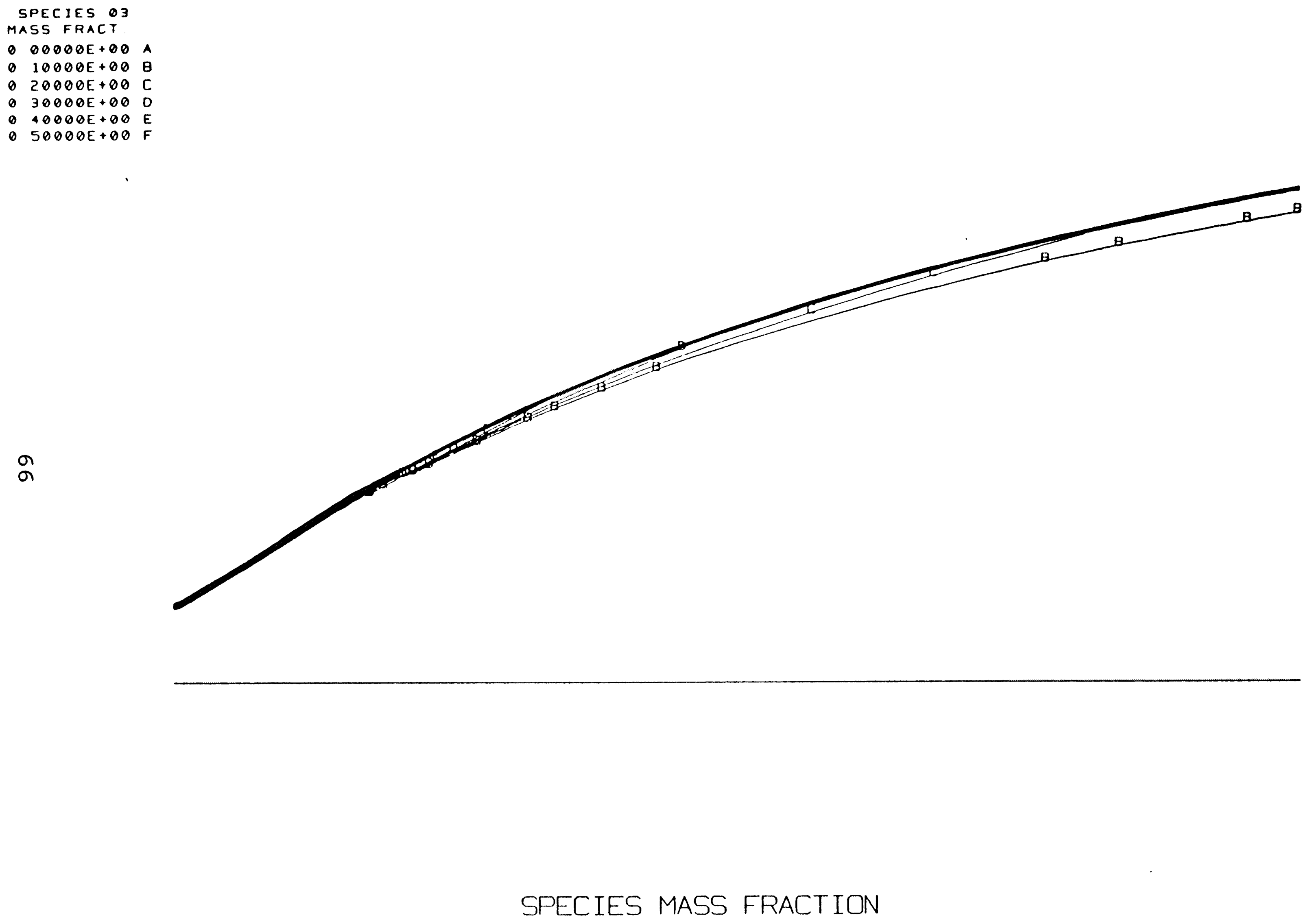


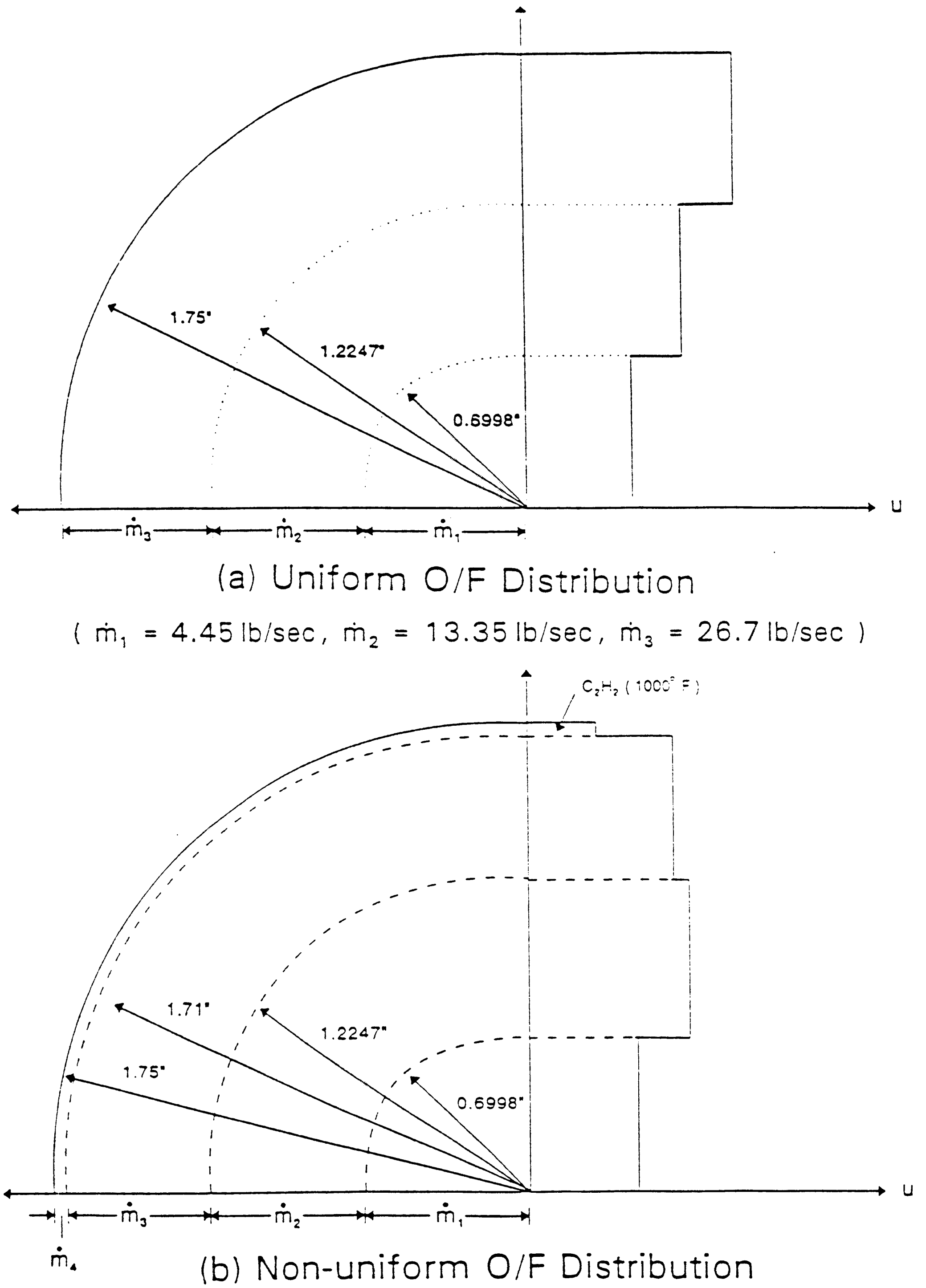
Fig. 35. Subscale STME Nozzle Flow - Species Distributions

4.2 Rocketdyne's Thrust Chamber Technology Tests

In the course of Rocketdyne's Thrust Chamber Technology Program for the Air Force, 3.4 inch diameter subscale motors were tested for two like-impinging circumferential fan injector configurations (Ref. 26). These tests showed that small changes in the injector configurations caused large changes in thrust chamber wall and nozzle heating.

The circumferential fan injectors utilized doublets of LOX impinging on LOX and RP-1 impinging on RP-1 to create circumferential fans which were aligned so that edge-to-edge intermixing of the fans occurred. The design change was to offset the outermost doublet rows so that the fuel and oxidizer did not circumferentially mix. The fuel rich layer near the wall apparently survived all the way to the nozzle throat to provide very effective cooling. When the outer row of fuel doublets and oxidizer doublets were offset they were also redirected so that the resulting fans were aligned parallel to the wall rather than inclined slightly toward the wall. This effect would also result in reducing the wall heat transfer rates along the motor. No measurements are reported which establish the quality of the mixing and dispersion along circumferential planes in the motor; however, measured heat transfer in various circumferential planes suggest that circumferential uniformity exists in the motor. SECA assumed such uniformity so that axisymmetric simulations of the motor operation could be made.

A constant mixture ratio (MR = 2.73) baseline case was analyzed using the FDNS code. Although the O/F ratio was held constant, the mass flux across the injector varied as specified in Fig. 36. Since the experiment was designed to provide a constant wall temperature, a constant wall temperature was assumed. The results of the analysis, presented in Fig. 37, show



(a) Uniform O/F Distribution

($\dot{m}_1$ = 4.45 lb/sec, $\dot{m}_2$ = 13.35 lb/sec, $\dot{m}_3$ = 26.7 lb/sec)

(b) Non-uniform O/F Distribution

($\dot{m}_1$ = 4.45 lb/sec, $\dot{m}_2$ = 13.35 lb/sec, $\dot{m}_3$ = 23.13 lb/sec, $\dot{m}_4$ = 3.57 lb/sec)

Fig. 36. Mass Flux at the Injector Face for the Rocketdyne Test Motor

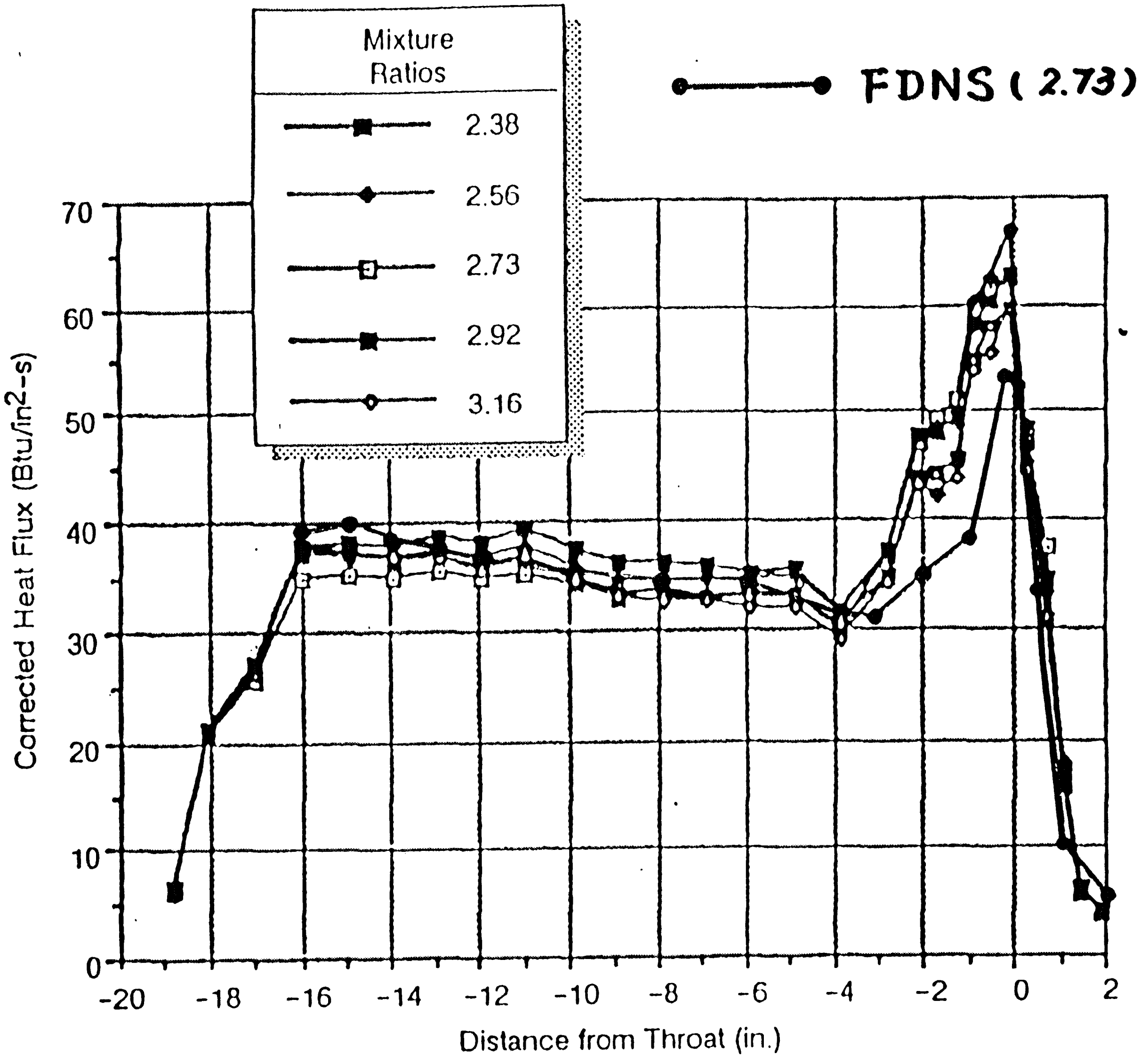


Fig. 37. Heat Flux Distribution for the Original Configuration of Rocketdyne's Circumferential Fan Injector

that the prediction is in good agreement with the experiment.

Another case using the Rocketdyne RP-1/O_2 test motor was analyzed using variable O/F and film cooled walls. The O/F distribution used for the film cooled case is shown in Fig. 36. The RP-1 fuel film was simulated as inert C_2H_2. Wall heat flux predictions for the film cooled case are shown in Fig. 38. It is evident that even though the film is initially cold enough to provide the correct wall heating, it mixes too fast to give the measured wall heat flux distribution. SECA investigators contend that the turbulent mixing, which is based on an incompressible k-ϵ turbulence model, is too fast. This phenomena has been observed repeatedly in variable density flowfield predictions. A thicker film specification on the startline would have a similar effect, but specifying a film thick enough to accomplish the heat reduction is not physically realistic. It is recognized that the delays associated with RP-1 droplet vaporization have not yet been quantitatively evaluated and could provide a similar effect in the region near the injector face. The problem was further analyzed by using the density correction (by temperature corrections) to the incompressible k-ϵ turbulence model which was successfully used to predict a dump combustor flowfield (Ref. 21). This correction was made by adjusting the production term in the ϵ transport equation to be:

$$(\rho\epsilon/k)\ (C_1 P_r - C_2\epsilon + C_3 T^{*C_4} P_r^2/\epsilon)$$

where C_1, C_2, and C_3 are the turbulence constants tuned for incompressible flows and T^* is the ratio of local to a 300 °K reference temperature. C_4 was taken to be 0.6 to describe the O_2/H_2 mixing and combustion in a dump combustor. This value over damped the film spreading rate. Rather a value of 0.4 was used. Predictions with this correction term are shown in Fig. 39 and

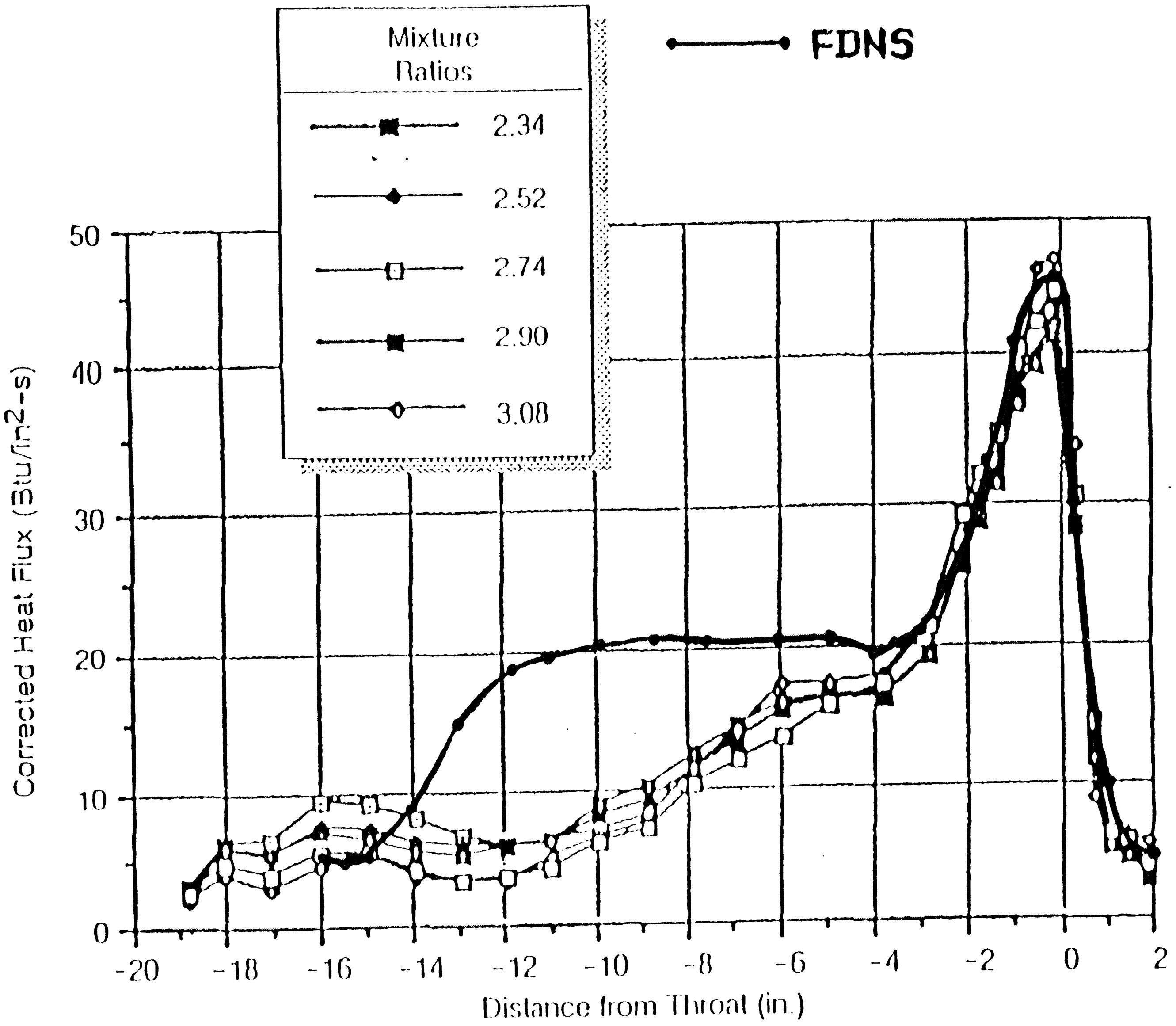


Fig. 38. Rocketdyne Film Cooled Test Motor

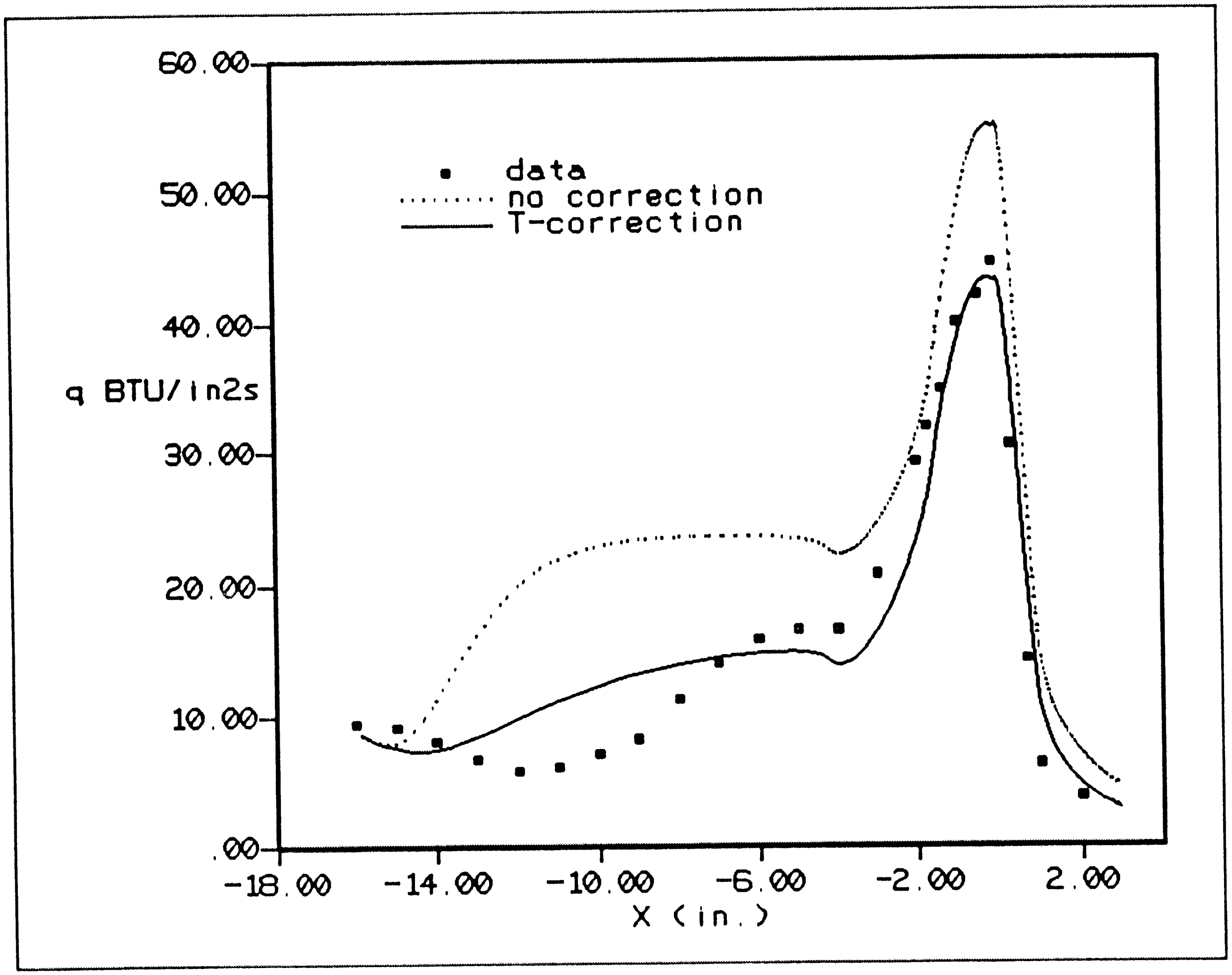


Fig. 39. Rocketdyne Film Cooled Test Motor Simulations with Various Turbulence Models

compared to the undamped solution. The improvement in the predicted wall heat flux distribution is dramatic. Although the experiment did not provide enough detailed data to verify all of the assumptions required in this analysis, the qualitative features of the heat transfer process are well predicted with the computation method described herein. Further justification of such methodology can only be made with experiments designed to elucidate the postulated computational analysis.

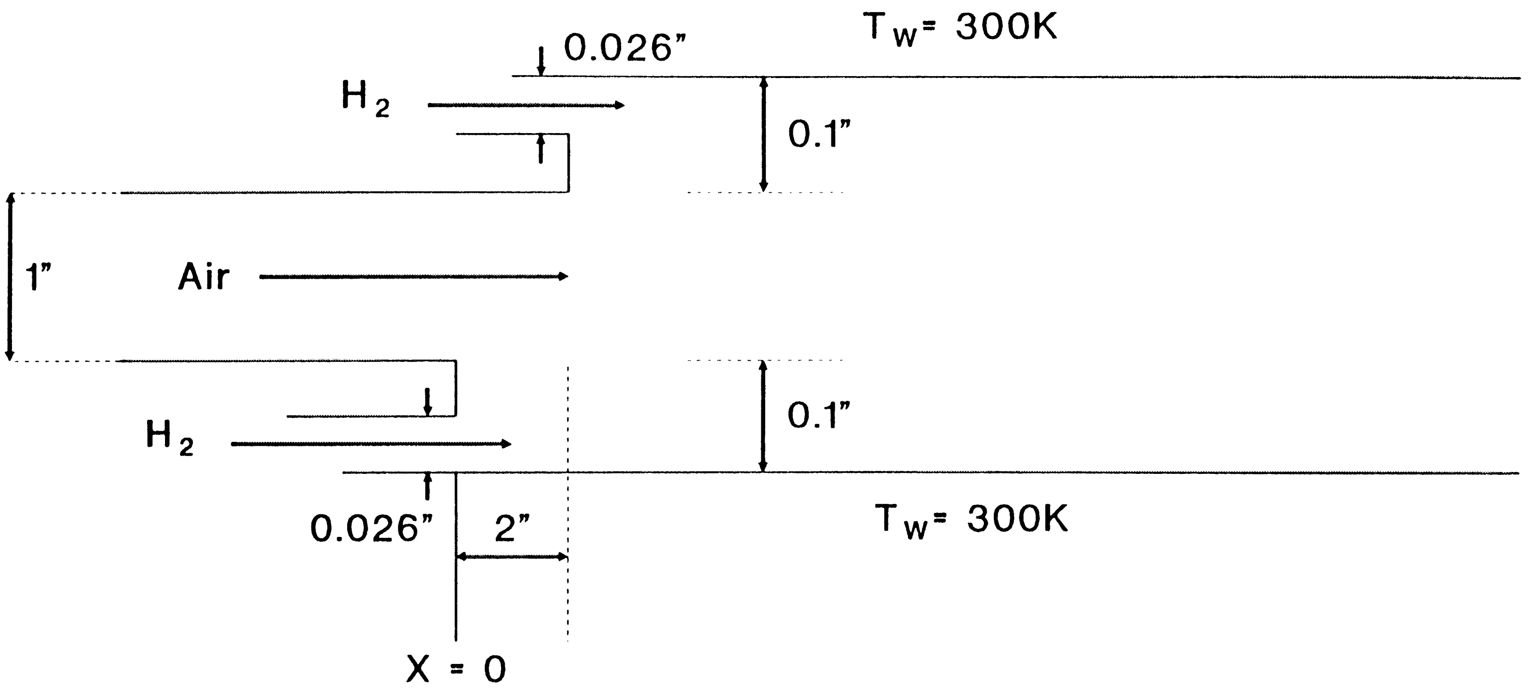
SECA interprets this comparison as proving that the modified mixing model is needed to simulate the behavior of a cooling film in a liquid rocket motor.

Since slow vaporization of RP-1 could also cause low heat transfer rates near the injector face, let us consider droplet vaporization. To analyze RP-1 vaporization, initial drop size, temperature, and velocity must be estimated, but more importantly the local droplet environment must also be estimated. If this environment is assumed to be vaporized fuel, this environment would be cool and the droplet could retain its identity for a long time. If the environment were combusted gases at a mixture ratio of the gases initially next to the spray film, the droplet would have a short lifetime. Wieber (Ref. 27) analyzed RP-1 droplets under typical rocket motor combustion conditions and found that a 100 μm drop would travel about 1 inch before it vaporized. Since the simulation shown in Fig. 39 was initiated 3 inches into the chamber and since the rapid mixing case mixes the film in 2 or 3 inches, SECA estimates that the rapid mixing curve could be displaced to the right by a maximum of 1 inch during the rise period. Such a simulation would not negate the conclusion that a slow gaseous film mixing model is required to simulate heat transfer to a rocket motor chamber wall. This conclusion is valuable design information, especially if the cases shown in Figs. 29-31 can be shown to correlate with the same analysis.

4.3 Film Cooling Verification Studies

Although several verification film cooling cases have been identified and run with the FDNS code, a case with heating rates comparable in magnitude to those experienced in a rocket nozzle and with well-defined initial conditions has not been evaluated. For low wall heating rates, the Holden case and the GASL case 41 have been run and the results of the calculation compared well with the experimental data when upstream boundary conditions were suitably defined (Refs. 28 & 29). Adequate measurements of upstream boundary conditions have not been reported for any film cooling experiments which have been found in the literature. Another interesting GASL experiment (Ref. 30) which involved a free shear layer between hydrogen and air, oxygen, and nitrogen (in successive tests) was also found and analyzed. Neither temperature nor species profiles were measured in the shear layer, rather wall heat transfer and pressure values were reported. The CFD simulation indicated that the wall heat transfer was not sensitive to the combustion kinetics rates, but were very sensitive to the inlet flowrate of hydrogen. The cold hydrogen film is computed to be very effective in insulating the surface from the hot shear layer, hence it controls wall heating regardless of how hot the shear layer actually is.

Yet another analysis involved the CFD simulation of another GASL experiment (Ref. 31), the double wall jet of hydrogen into air shown in Fig. 40. The measured and predicted wall heat fluxes are shown in Figs. 41 and 42. The hydrogen jets are very thin and expected to be laminar; therefore, the reported velocity was considered to be: first, the average for the stream and second, the maximum of the laminar jet profile. Using the maximum value is shown to simulate the test data quite well. The flowfield features from this simulation are shown in Figs. 43-46. This is exactly the same boundary condition adjustment which was



Air	Hydrogen (ϕ_{Total}= 1.7)	
P = 5.67 psi	(Bottom)	(Top)
T = 1094 K	P = 11.52 psi	P = 8.03 psi
M = 3.6	T = 132 K	T = 132 K
Mach 10 flight	M = 2.5	M = 2.5
Enthalpy		

Fig. 40. Slot Injection Configuration and Flow Conditions for Supersonic Film Cooling Study

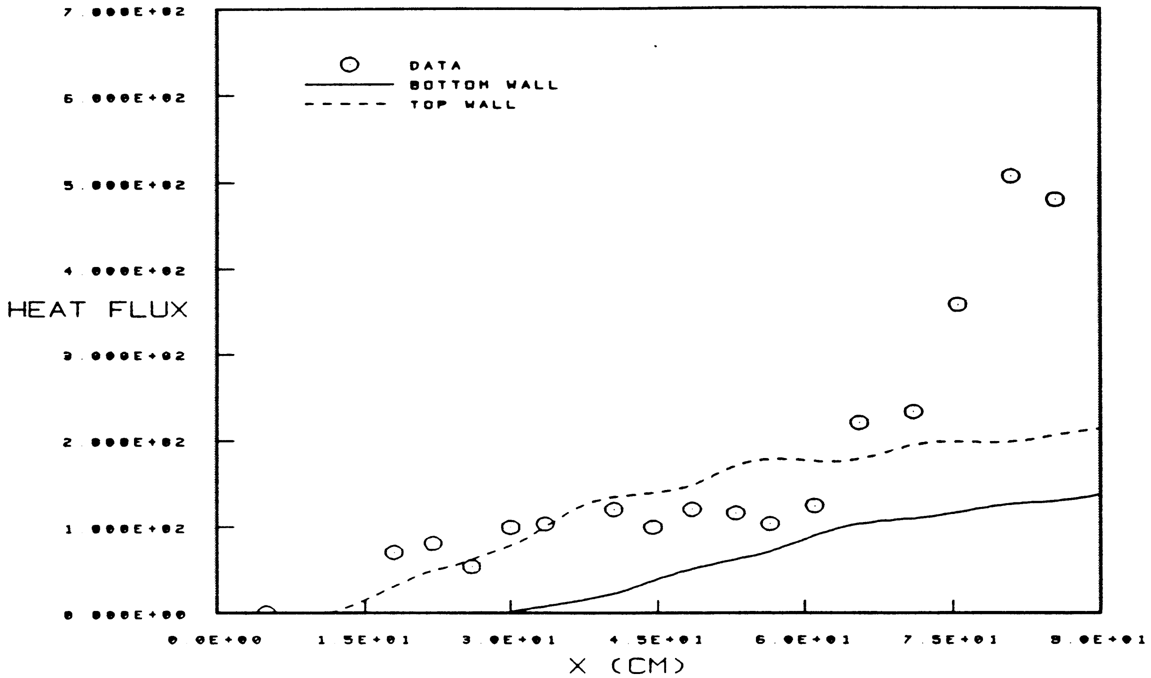


Fig. 41. The Heat Flux Distribution for Air/Hydrogen Film Cooling Flow with Finite Rate Chemistry and ε-Correction Model (Coolant Injected M_{ave} = 2.5)

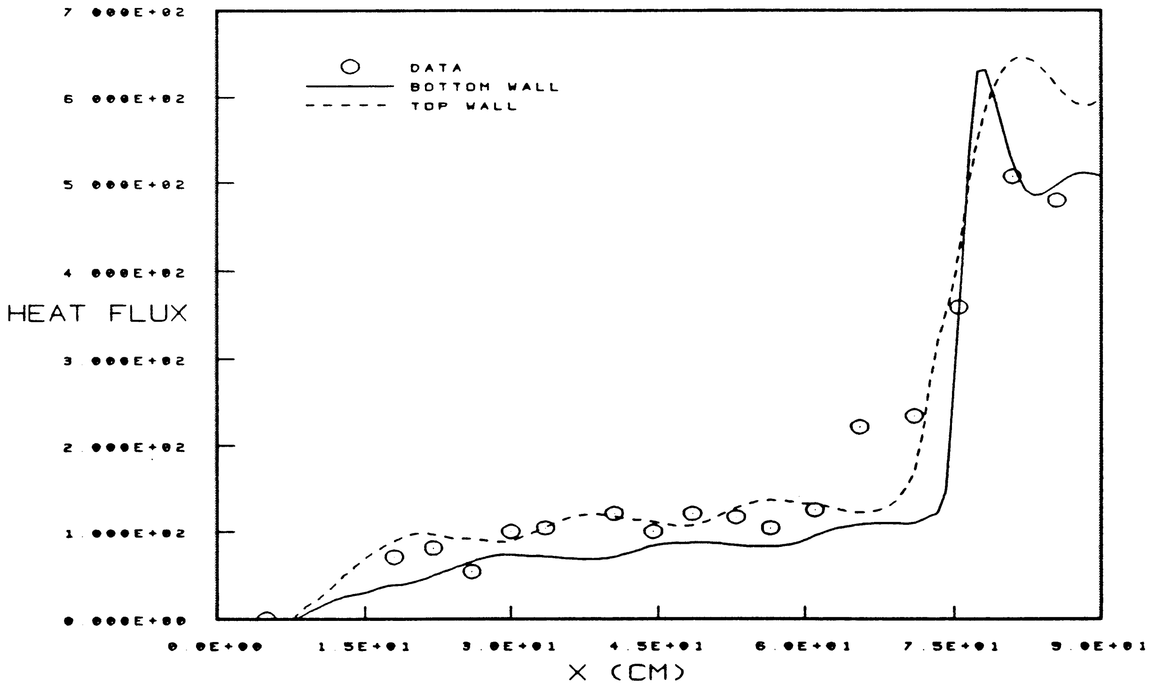


Fig. 42. The Heat Flux Distribution for Air/Hydrogen Film Cooling Flow with Finite Rate Chemistry and ε-Correction Model (Coolant Injected M_{max} = 2.5)

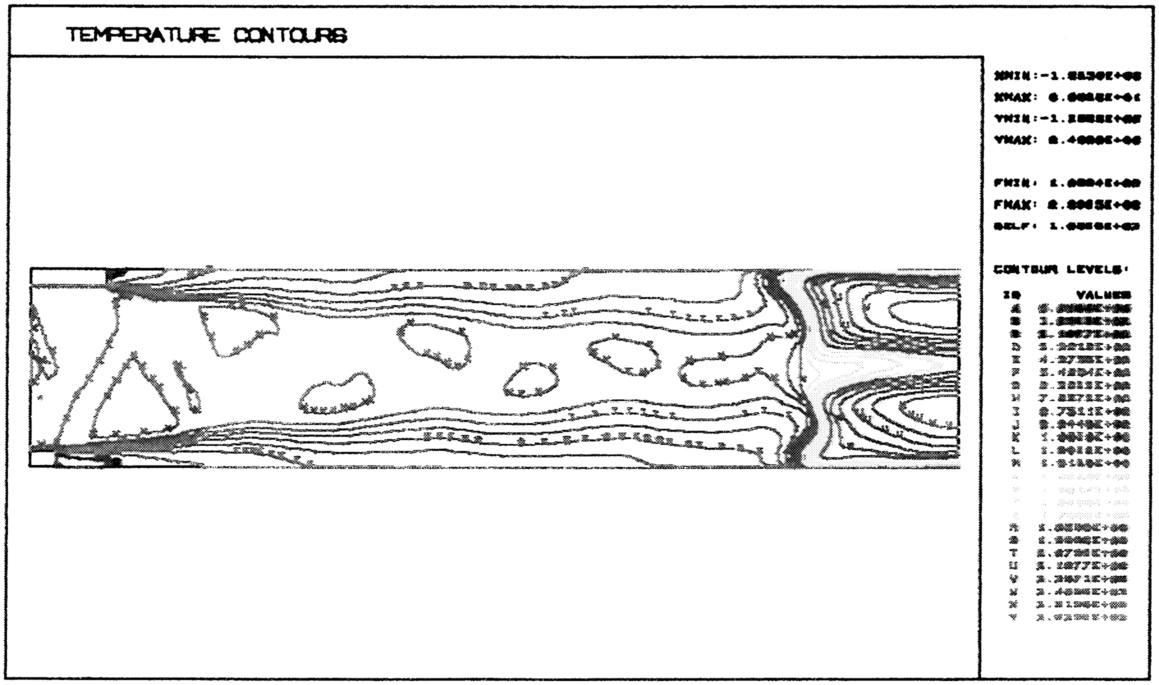


Fig. 43. The Temperature Contours for Air/Hydrogen Film Cooling Flow with Finite Rate Chemistry and ε-Correction Model (Coolant Injected M_{max} = 2.5)

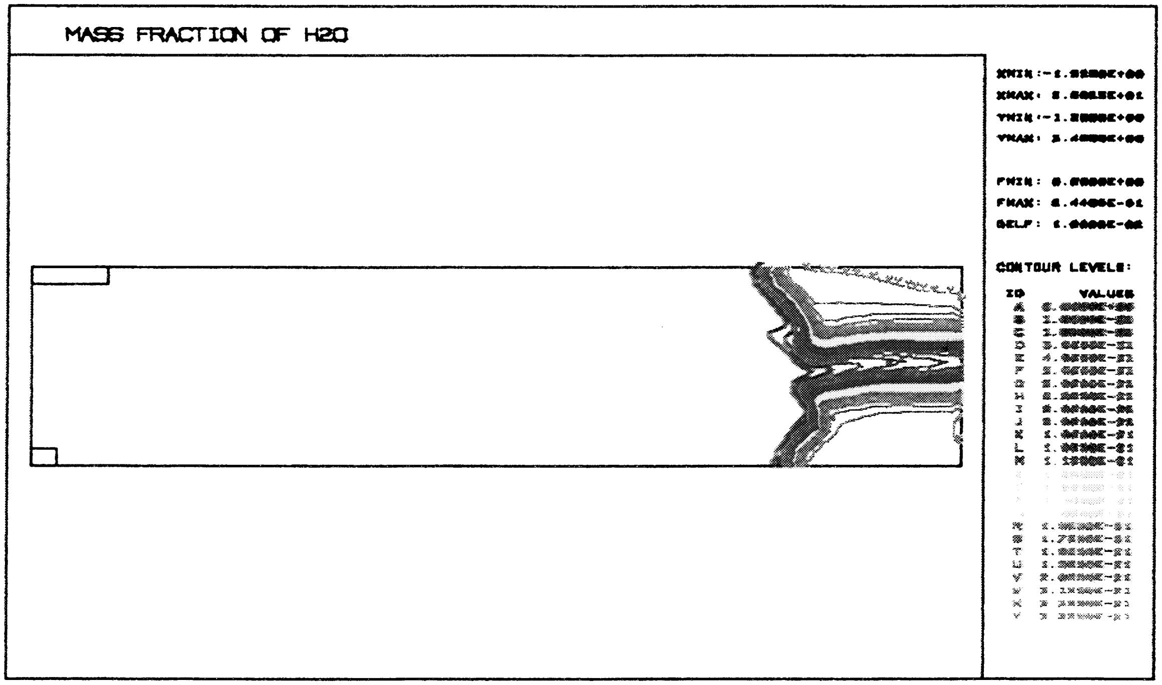


Fig. 44. The Mass Fraction Contours of H_2O for Air/Hydrogen Film Cooling Flow with Finite Rate Chemistry and ε-Correction Model (Coolant M_{max} = 2.5)

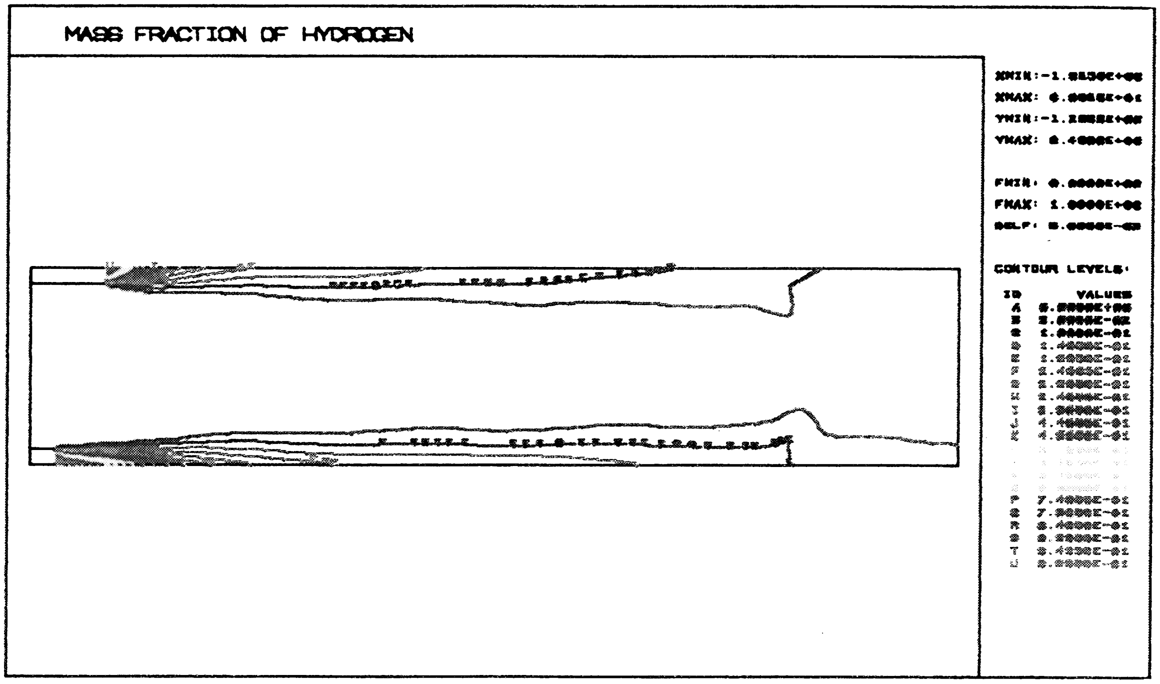


Fig. 45. The Mass Fraction Contours of H_2 for Air/Hydrogen Film Cooling Flow with Finite Rate Chemistry and ε-Correction Model (Coolant M_{max} = 2.5)

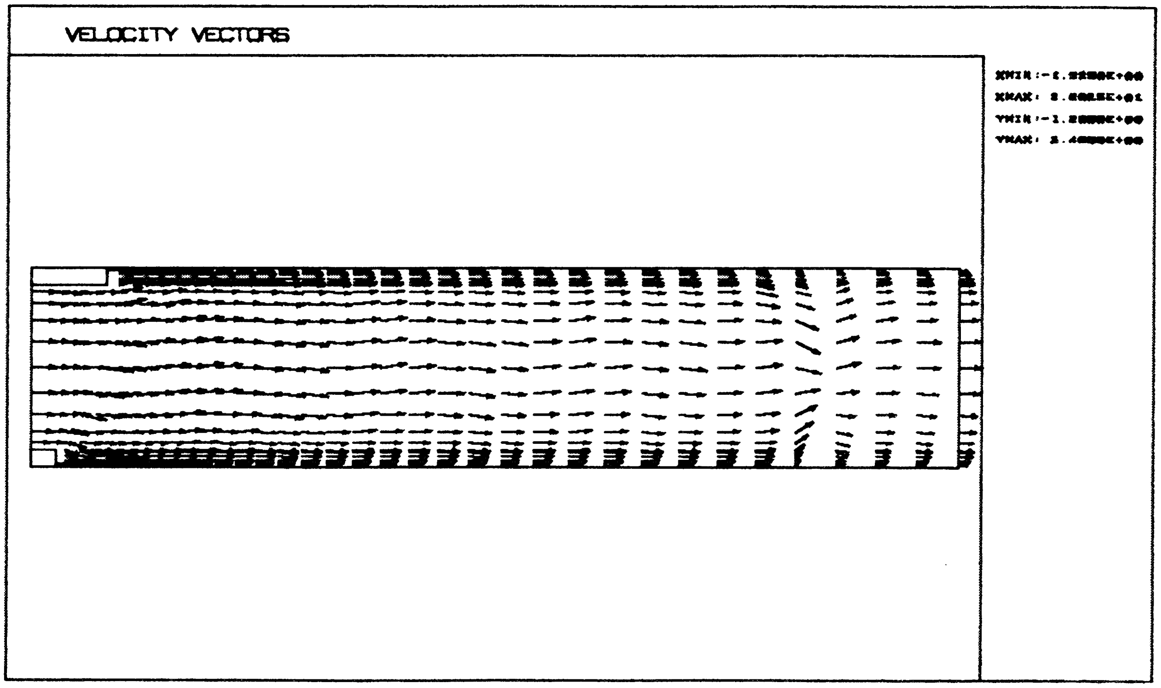


Fig. 46. The Velocity Vectors for Air/Hydrogen Film Cooling FLow with Finite Rate Chemistry and ε-Correction Model (Coolant Injected M_{max} = 2.5)

required to simulate the GASL case 41 test data. Apparently, slot flow conditions which are calculated with one-dimensional analyses are more indicative of centerline velocities than of mean velocities. It would be very worthwhile if such behavior could be experimentally verified in future wall jet studies. Any adjustment on kinetics rates does not result in a realistic prediction, unless the initial wall jet velocity is modified. The wall heating is controlled by both mixing in the flowfield and combustion kinetics. Variation of reaction rates while keeping the inlet flow boundary condition constant at the higher flowrate does not result in an acceptable simulation.

4.4 PSU's Gas/Gas Coaxial Injector Experiment

Penn State is currently conducting experiments to characterize the flowfield created by a single shear coaxial injector element. Initial data from this experiment are velocity and intensity measurements for GOX/GH_2 combustion (Ref. 32). Additional measurements to include temperature fields are in progress. SECA has made a preliminary CFD analysis of these experiments. The FDNS code was used for the simulation. The H_2/O_2 kinetics model shown in Table 2 was used to describe the combustion. Forward rates were specified; backward rates were calculated with equilibrium constants. In order to ignite the flow, a temperature of 2500°K was used to initially evaluate the rate constants; once the flame started, the actual temperature was used to continue the calculations. Two turbulence models were used: the extended k-ϵ model and this model with a temperature correction to account for reduced turbulence in regions of low density. The inlet turbulence intensities for both GOX and GH_2 were assumed to have the same level as fully developed channel flows. The results are shown in Figs. 47-54. The two turbulence models give very similar results. Though the mean axial velocity was relatively well predicted, the estimate

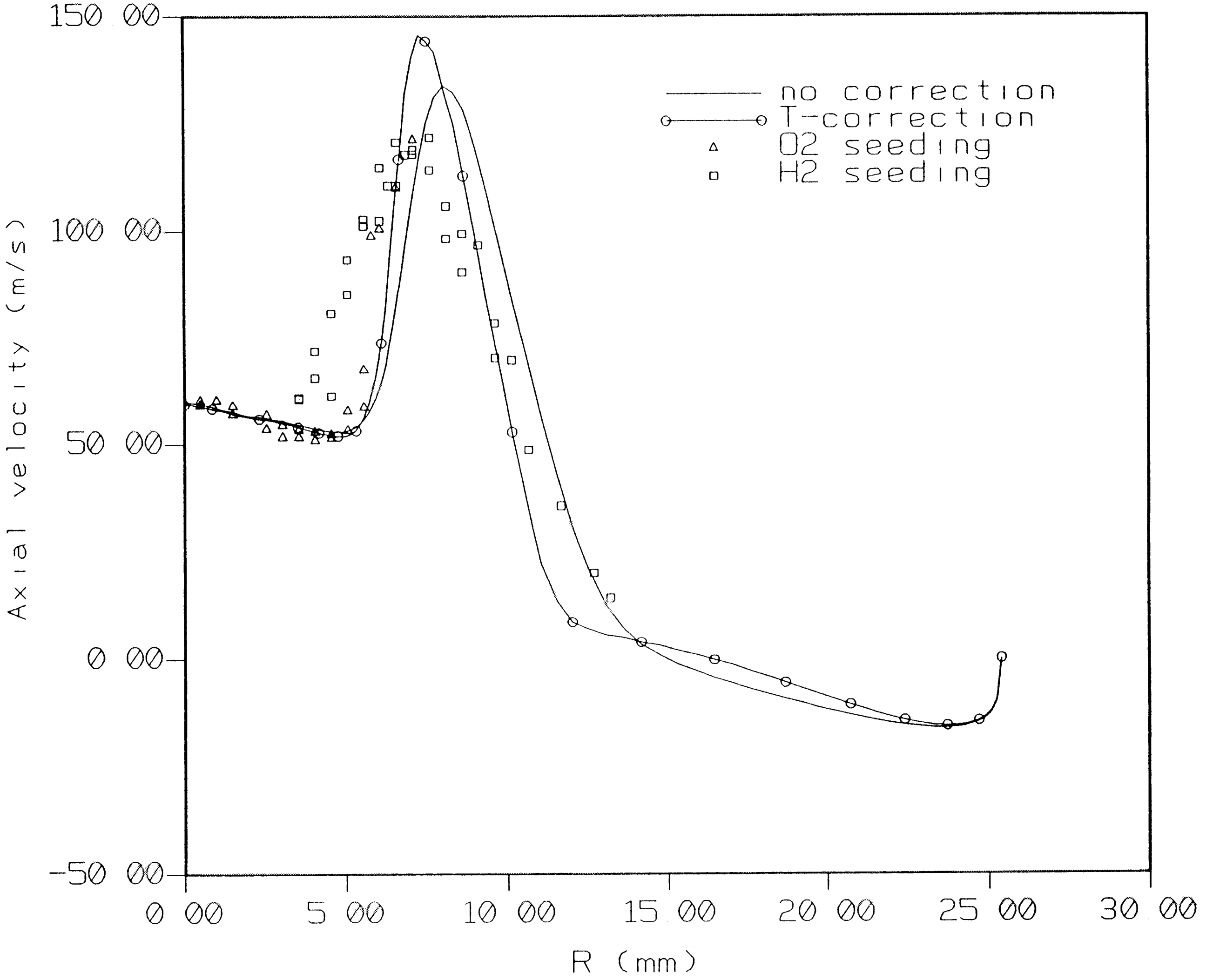


Fig. 47. Mean Axial velocity profile the the axial location X=25 4mm (finite rate at 2500 deg K, explicit species production)

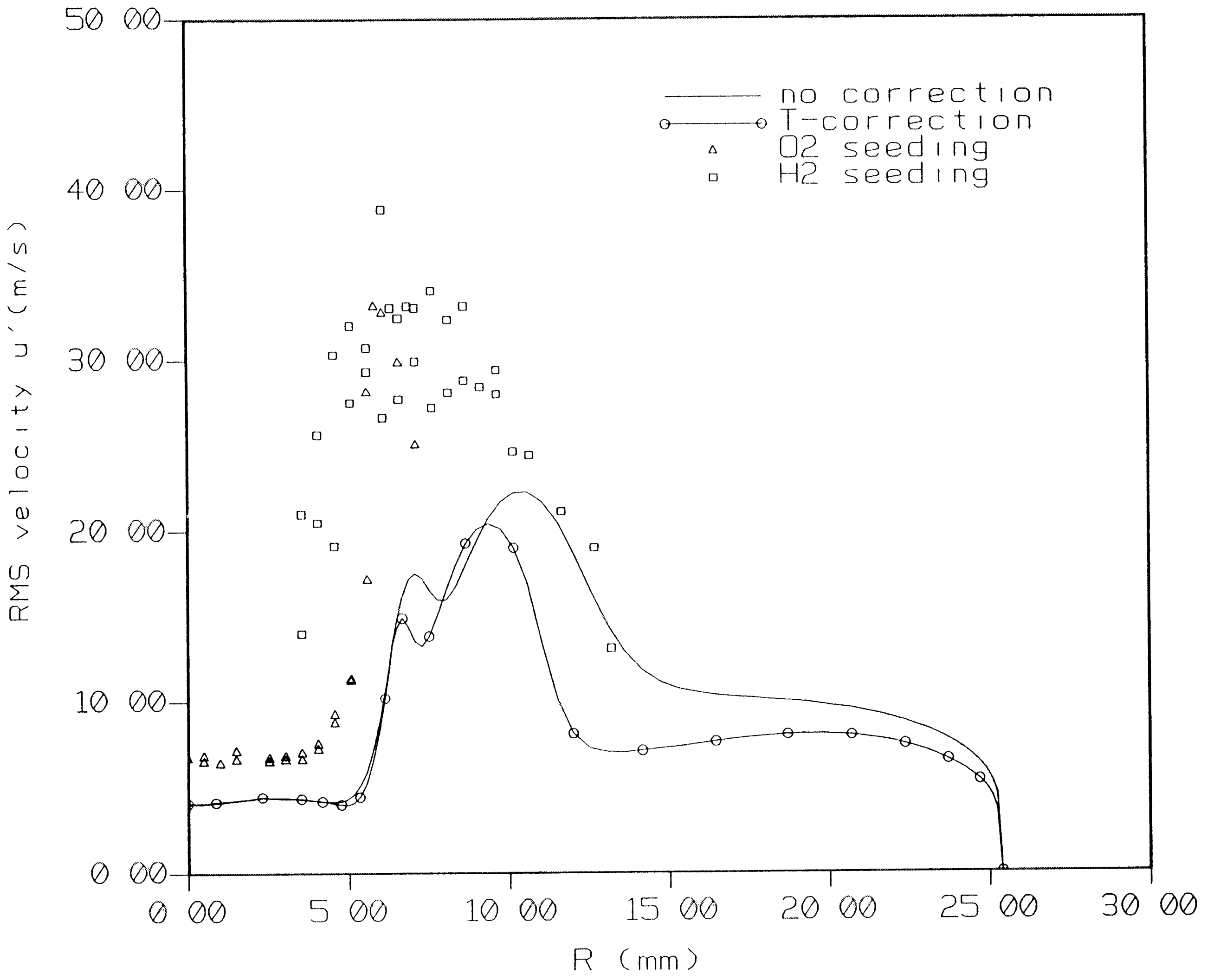


Fig. 48. Root mean square axial velocity profile at location X=25 4mm (finite rate at 2500 deg K, explicit species production)

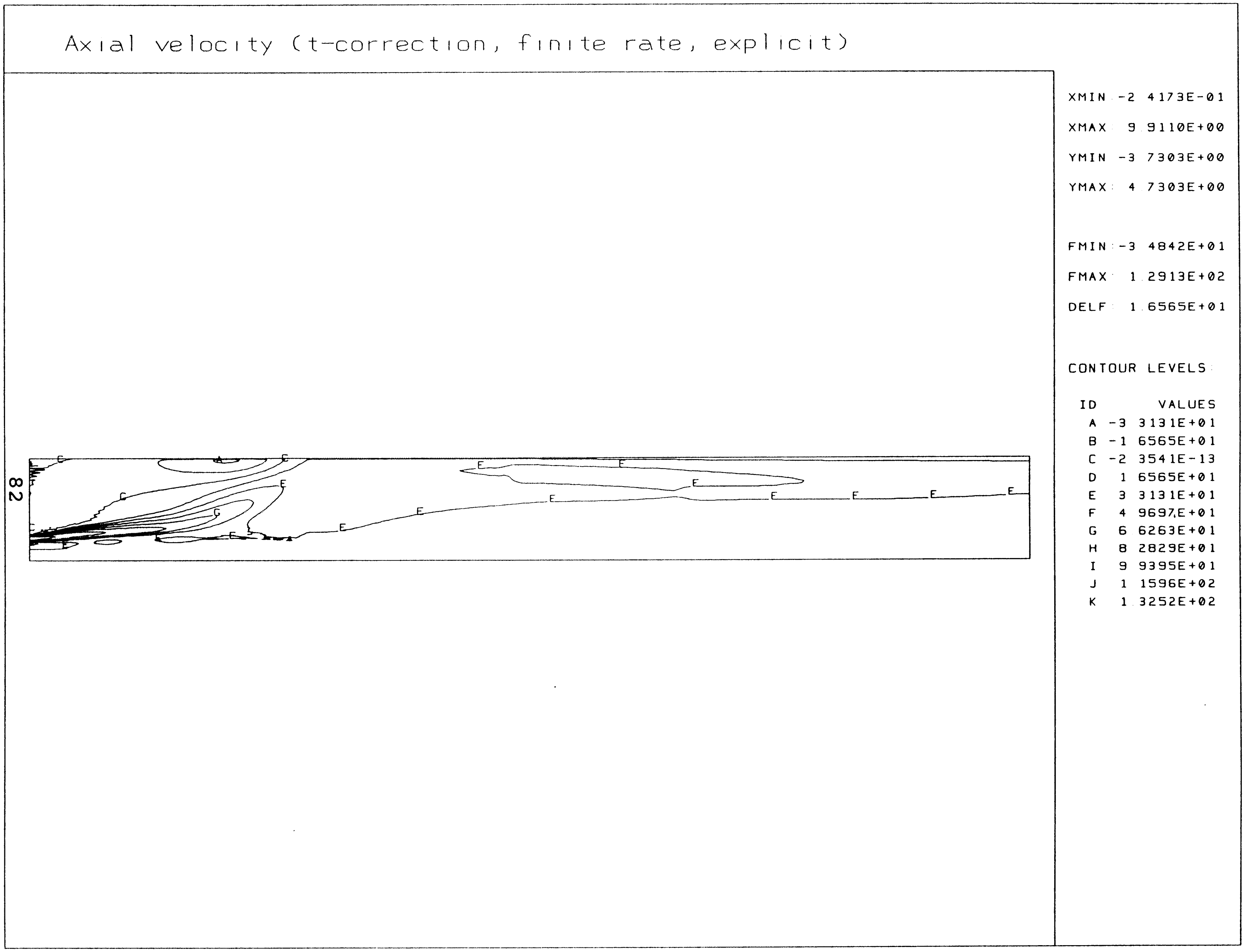


Fig. 49. Velocity Predictions for the PSU Single Injector Combustor (SIC)

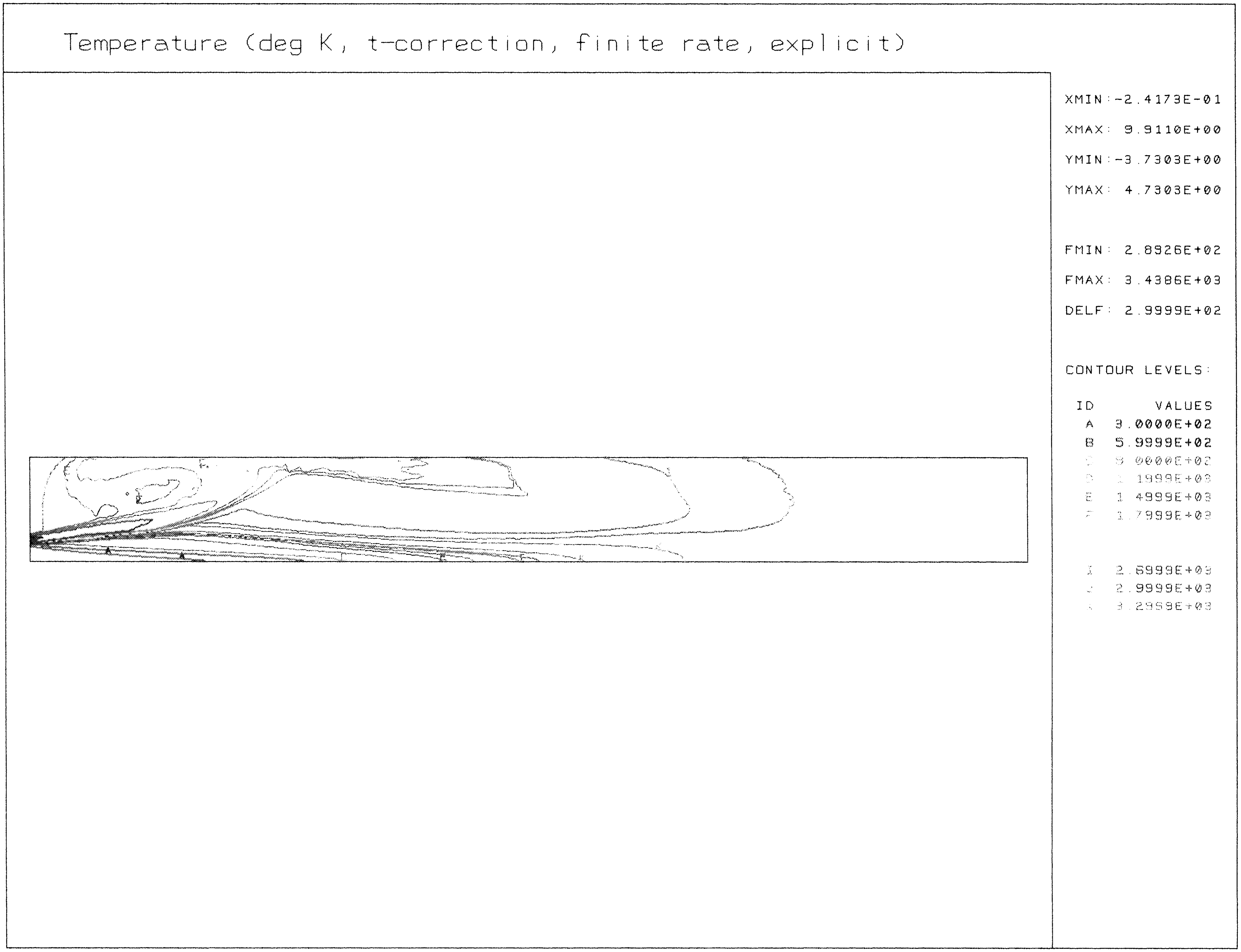


Fig. 50. Temperature Predictions for the PSU (SIC)

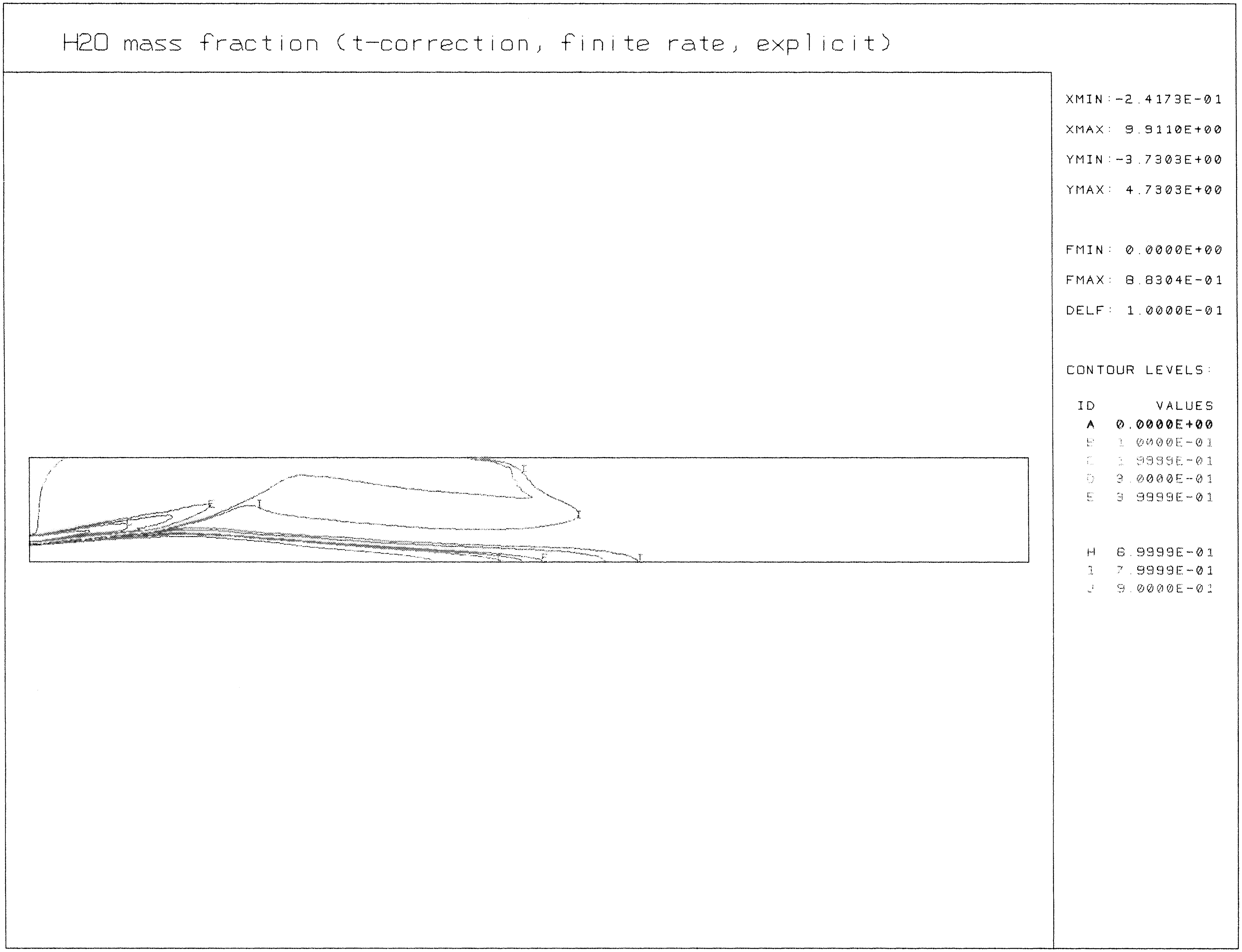


Fig. 51. Concentration Predictions for the PSU SIC

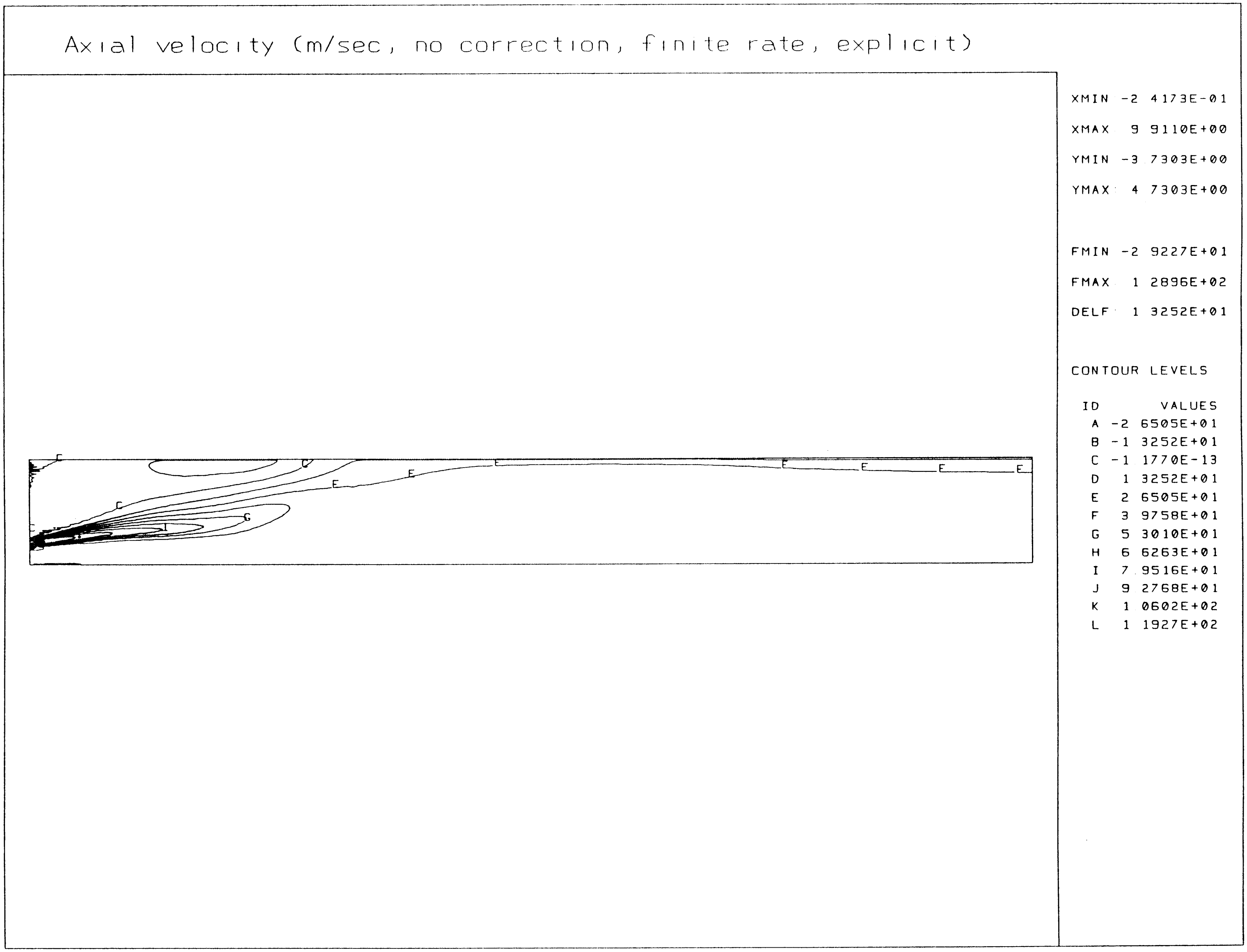


Fig. 52. Velocity Predictions for the PSU SIC

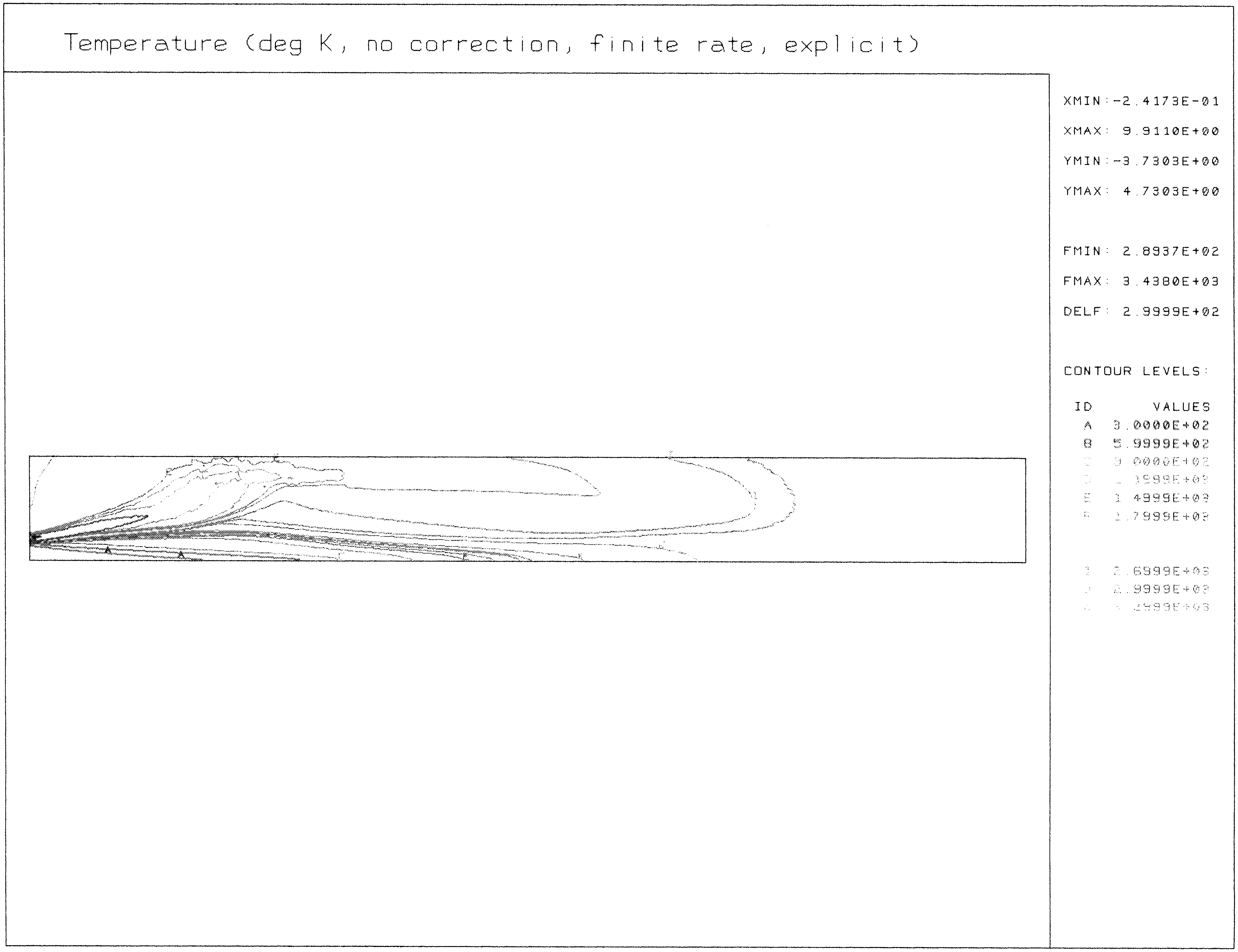


Fig. 53. Temperature Predictions for the PSU SIC

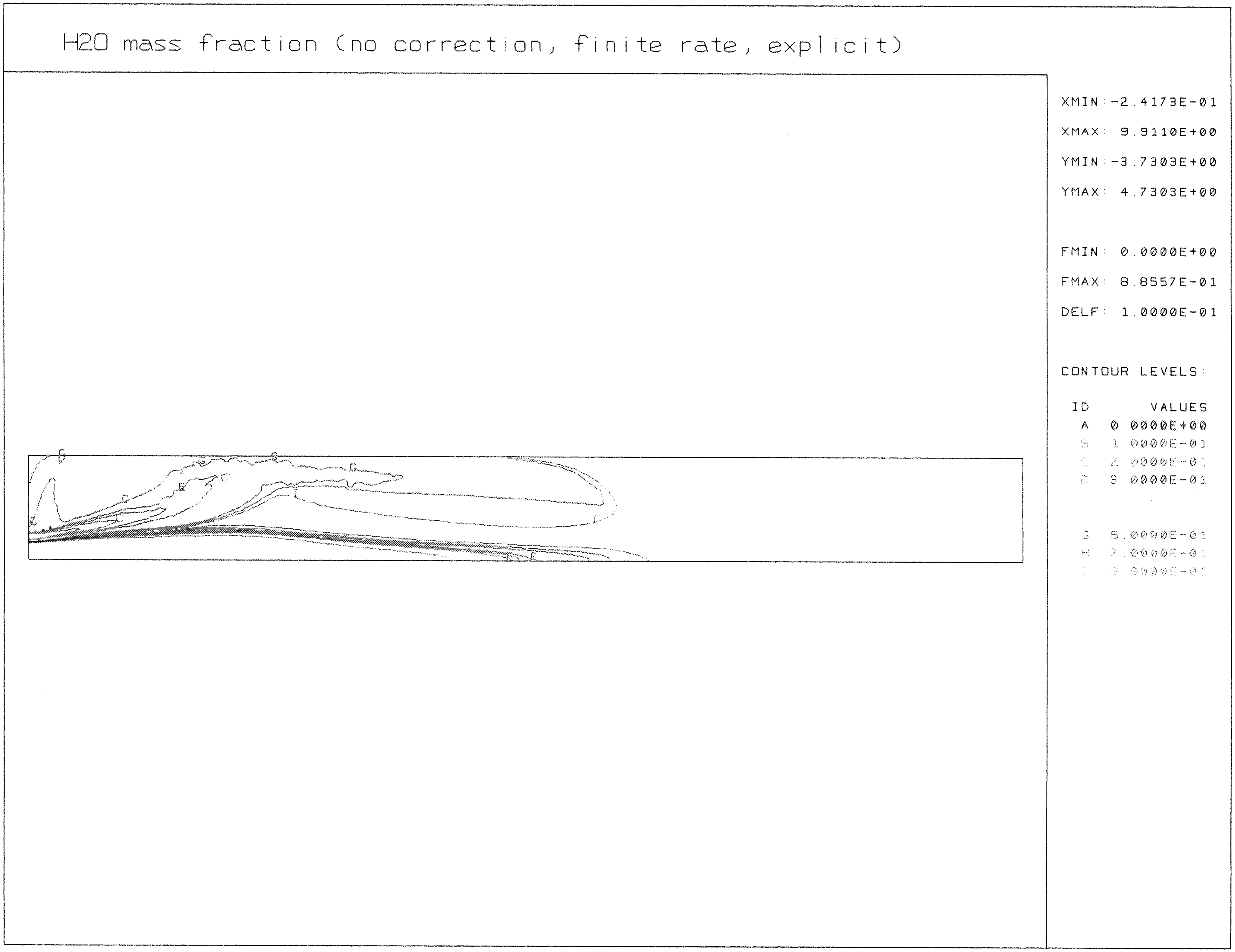


Fig. 54. Concentration Predictions for the PSU SIC

Table 2. H_2/O_2 Combustion Kinetics

Reaction	A	B	E/R
$H_2 + O_2 \rightleftharpoons OH + OH$	1.7000E13	0	2.4070E4
$OH + H_2 \rightleftharpoons H_2O + H$	2.1900E13	0	2.5900E3
$OH + OH \rightleftharpoons O + H_2O$	6.0230E12	0	5.5000E2
$O + H_2 \rightleftharpoons H + OH$	1.8000E10	1.0	4.4800E3
$H + O_2 \rightleftharpoons O + OH$	1.2200E17	-0.91	8.3690E3
$M + O + H \rightleftharpoons OH + M$	1.0000E16	0	0
$M + O + O \rightleftharpoons O_2 + M$	2.5500E18	-1.0	5.9390E4
$M + H + H \rightleftharpoons H_2 + M$	5.0000E15	0	0
$M + H + OH \rightleftharpoons H_2O + M$	8.4000E12	-2.0	0

of the initial turbulence levels were apparently too low.

Since the flame appears to extend to near the injector face, it is expected that the reported GOX and GH_2 temperatures were measured too far upstream to be typical of injector exit conditions. Further temperature measurements are in progress at PSU to verify this observation. Because more critical measurements are expected, this analysis must be considered preliminary and further simulations should be made as the new test data become available.

A further simulation with higher inlet turbulence levels (10% intensities for both GOX and GH_2 injector exit flows) was conducted. The extended $k-\epsilon$ turbulence model was employed in this study, and the same ignition procedure as before was applied. These results are shown in Figs. 55-57. The numerical prediction of mean axial velocity profiles was improved. The turbulence intensity for the flow within the inner jet stream (GOX stream) was well simulated, however, the turbulence level was under-estimated for the flow expanding from the outer jet stream (GH_2) into the chamber.

A higher turbulence intensity (20%) for the outer jet (GH_2), modified based on the above simulation, was investigated. Indications from Figs. 58-59 are that the intensity is still not high enough in the shear layer, but further investigation was not conducted. This preliminary investigation was stopped at this point. Notice that the effect of the temperature correction is to qualitatively modify the predicted velocity and intensity predictions so that they more closely match the test data. However, temperature measurements should be considered before further analyses are made.

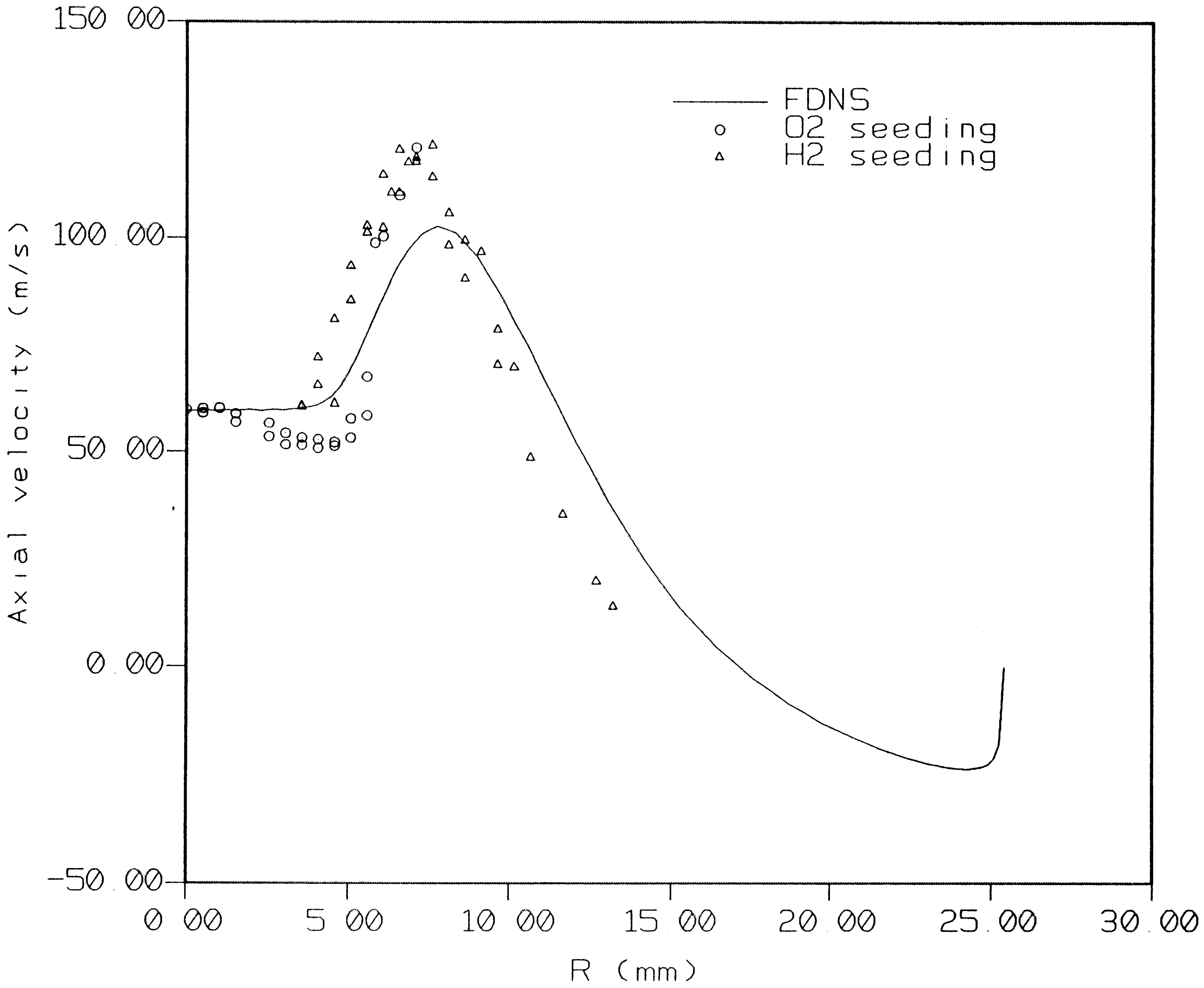


Fig. 55. Mean Axial Velocity Profile at the Axial Location X=25.4mm (finite rate, explicit, 10%/20% turbulence level for GOX/GH_2)

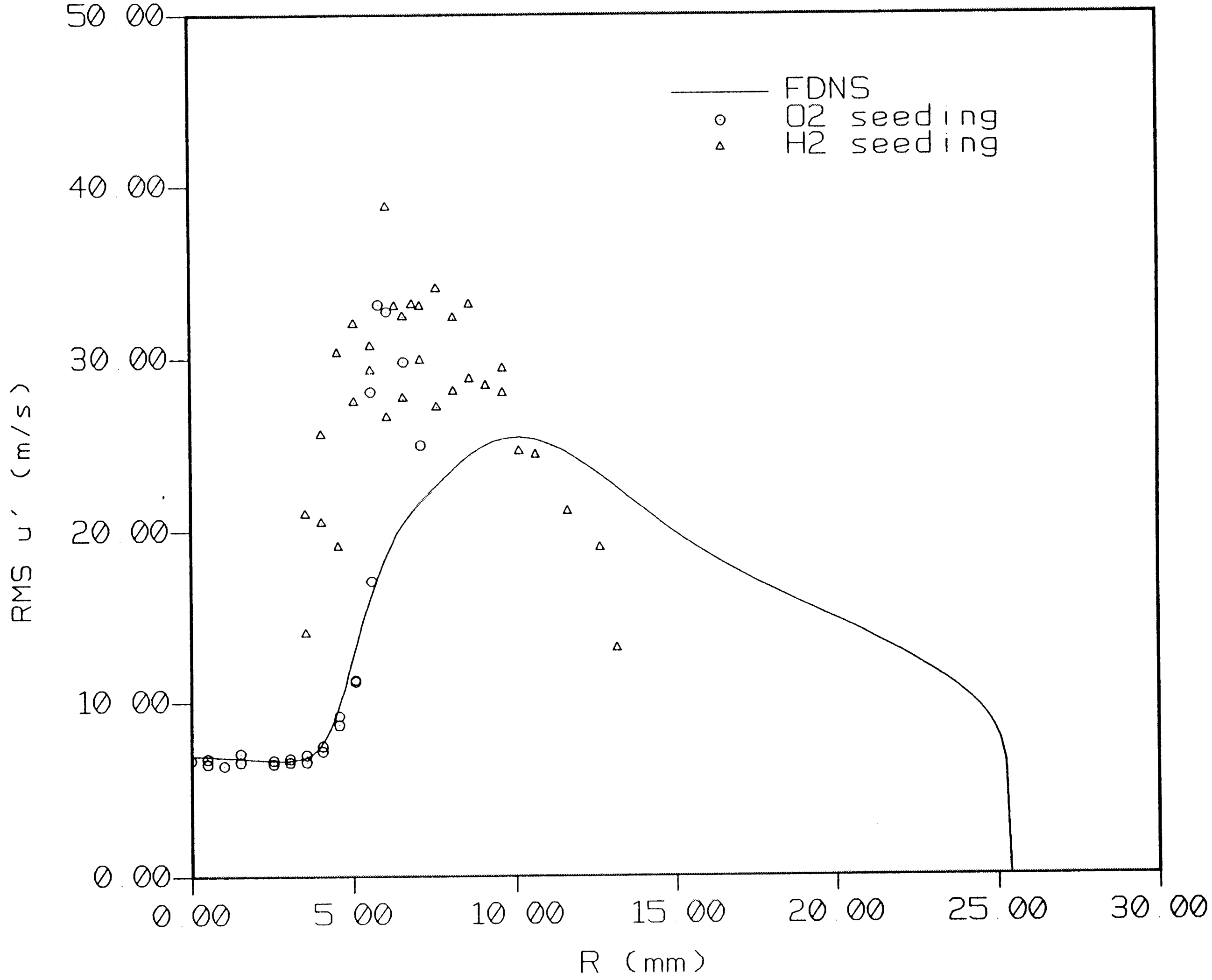


Fig. 56. RMS Axial Velocity Profile at the Axial Location X=25.4mm (finite rate, explicit, 10%/20% turbulence level for GOX/GH_2)

5.0 ADAPTIVE GRID GENERATION

As part of the Overall Engine Heat Transfer Model, Task 2.0, a special grid generation package utilizing the adaptive gridding features of the EAGLE code was developed by Mississippi State University under a subcontract. The final report for this subcontract from Mississippi State is presented in Appendix B.

region, injector analysis for injector configurations typical of the near wall locations and making a conjugate heat transfer analysis for the motor wall. The case selected for further analysis should be carefully selected and should have been experimentally evaluated. Several appropriate test cases are identified in the text.

4. Since the experimental data from the PSU single coaxial injector study for GOX and LOX flows will not be available for some time, test data which have already been collected for studying other combustor configurations should be used to further validate the FDNS injector/streamtube model reported herein.

7.0 REFERENCES

1. Chen, Y.S., J.A. Freeman, and R. C. Farmer, "Heat Transfer in Rocket Engine Combustion Chambers and Regeneratively Cooled Nozzles," SECA-P-90-09, SECA, Inc., Huntsville, AL, 1990.

2. Combs, L.P., "Liquid Rocket Performance Computer Model with Distributed Energy Release," NASA CR-114462, 1972.

3. Chen, Y.S., et al, "FDNS - A General Purpose CFD Code - User's Guide (Version 3.0)", ESI-TR-93-01, Engineering Sciences, Inc., Huntsville, AL, 1993.

4. Chiang, C.H., and Sirignano, W.A., "Axisymmetric Vaporizing Oxygen Droplet Computations", AIAA Paper AIAA-91-0281, 29th Aerospace Sciences Meeting, Jan. 7-10, 1991, Reno, NV.

5. Shearer, A.J., H. Tamura, and G.M. Faeth, "Evaluation of a Locally Homogeneous Flow Model of Spray Evaporation," J. Energy, 3, pp. 271-278, 1979.

6. Mao, C-P., G.A. Szekely, Jr., and G.M. Faeth, "Evaluation of a Locally Homogeneous Flow Model of Spray Combustion," J. Energy, 4, pp. 78-87, 1980.

7. Beer, et al, 9th Symp. (Intl.) on Combust., pp. 892-900,

8. Bachalo, W.D., "Diagnostic Development for the Characterization of Liquid Fuel Rocket Engine Injector Atomization," Report on NAS8-37323, Aerometrics, Inc., Mountain View, CA, October 1987.

9. Eskridge, Richard, Personal Communication, NASA/MSFC/EP55, August 1992.

10. Nurick, W.H., "DROPMIX - A PC Based Program for Rocket Engine Injector Design," W.J. Schafer Associates, Inc., Irvine, CA.

11. McCarty, R.D., and L.A. Weber, "Thermophysical Properties of Oxygen from the Freezing Liquid Line to 600 R for Pressures to 5000 Psia", NBS Technical Note 384 (1971.

12. McCarty, R.D., and L.A. Weber, "Thermophysical Properties of Parahydrogen from the Freezing Liquid Line to 5000 R for Pressures to 10,000 Psia", NBS Technical Note 617 (1972).

13. Hougen, O.A., K.M. Watson, and R.A. Ragatz, Chemical Process Principles, Part II, Thermodynamics, John Wiley & Sons, NY (1959).

14. Hirschfelder, J.O., et al, Ind. & Engr. Chemistry, Vol. 30, pp. 375-390, 1958.

15. Wang, T.S., Y.S. Chen, and R.C. Farmer, "Investigation of the Transient Fuel Preburner Manifold and Combustor", NAS8-37461, Final Report SECA-89-10, (1989).

16. Keenan, J.H., and Kayes, F.G., Thermodynamic Properties of Steam, Wiley and Sons, NY, 1963.

17. Reid, R.C., et al, The properties of Gases and Liquids, 3rd Ed., McGraw-Hill Book Co., NY, NY, 1977.

18. White, F.M., Heat and Mass Transfer, Addison-Wesley Pub.Co., Reading, Mass., 1988.

19. Patankar, S.V. and D.B. Spalding, "Heat and Mass Transfer in Boundary Layers," CRC Press, Cleveland, OH, 1968.

20. Freedman, E., "BLAKE - A Thermodynamics Code Based on TIGER, Users' Guide and Manual," TR ARBRL-02411, U. S. Army Ballistic Research Laboratory, Aberdeen Proving Ground, MD, July 1982.

21. Wang, T.,Chen, Y.S., and R.C. Farmer, "Numerical Study of Reactive Dump Combustor Flowfields with a Pressure Based CFD Method", AIAA Paper 89-2798, July, 1989.

22. Wang, T.S., and R.C. Farmer, "Turbulent Combustion Kinetics for Complex Hydrocarbon Fuels", AIAA Paper 88-0733, Jan. 1988.

23. Wang, T.S., and V. Luong, "Numerical Analysis of the Hot-Gas-Side and Coolant-Side Heat Transfer in Film-Cooled Rocket Engine Combustors," submitted to AIAA J of Thermophysics and Heat Transfer, 1993.

24. Chen, Y.S., et al, "Numerical Simulation of Film Cooling Effectiveness in a Combustion Chamber," paper submitted to AIAA, 1993.

25. Tang, Y.L. and M.D. Schuman, "Numerical Modeling of Liquid-Liquid Bi-Propellant Droplet/Gas Reacting Flows," AIAA 92-0232, 30th Aerospace Sciences Meeting, Reno, NV, January 1992.

26. Singler, Lisa, Phillips Laboratory, Edwards Laboratory, Edwards AFB, CA, personnel communication, May 1992.

27. Wieber, P.R., "Calculated Temperature Histories of Vaporizing Droplets to the Critical Point," AIAA Journal, 1, No. 12, pp. 2764-2770.

28. Chen, Y.S., and G.C. Cheng, "CFD Modeling of Film Cooling Test Cases," presentation to NASA/MSFC Combustion Devices Technology Team, January 1992.

29. Holden, M.S., "A Data Base of Experimental Studies of Shock Wave/Wall Jet Interaction in Hypersonic Flow," CALSPAN-VB Research Center, Buffalo, NY, April 1990.

30. Erdos, J., et al, "Experiments on Shear Layer Mixing at Hypervelocity Conditions," AIAA-92-0628, January 1992.

31. Anderson, G., A. Kumar, and J. Erdos, "Progress in Hypersonic Combustion Technology with Computation and Experiment," AIAA-90-5254, October 1990.

32. Pal, S., et al, "An Experimental Study of Characteristic Combustion-Driven Flows for CFD Validation," Fifth Annual Symposium, NASA Propulsion Research Center, The Pennsylvania State University, Sept. 8-9, 1993.

APPENDIX A

Listing of Grid Code POST

```
      PROGRAM POST
C
C     GRID GENERATOR FOR LOX POSTS
C
      PARAMETER (LTAPE=3,LPRT=4,NUMN=12000)
      CHARACTER ANS,title*7
      COMMON/COM1/ ISTA(200),JSTA(100),KSTA(100)
      COMMON/COM2/ XSTA(20,100),RSTA(20,100),AL2(13)
      COMMON/COM3/ X(NUMN),Y(NUMN),Z(NUMN)
      COMMON/COM4/ DELXI(30,3),DELXF(30,3),teta(350,3)
      COMMON/COM5/ PI,RADDEG,A25,A30,A35
      COMMON/COM6/ ITYPE,IROW,ITOT,JTOT,KTOT
      COMMON/COM7/ ISTART,ISTOP,JSTART,JSTOP,KSTART,KSTOP
      PI=3.141593
      RADDEG=180.0/PI
      A25=25.0/RADDEG
      A30=30.0/RADDEG
      A35=35.0/RADDEG
      IROW=13
C
C     L2 FROM RS009207
C
      DO 100 I=1,5
      AL2(I)=6.390
  100 CONTINUE
      AL2(6)=6.672
      AL2(7)=6.96
      AL2(8)=7.26
      AL2(9)=7.56
      AL2(10)=7.85
      AL2(11)=8.15
      AL2(12)=8.44
      AL2(13)=8.74
C
C     INPUT GRID DATA
C
      JSTART=1
      JSTOP=5
      DO 300 I=1,100
      DO 200 J=1,20
      XSTA(J,I)=0.0
  200 RSTA(J,I)=0.0
      RSTA(2,I)=0.094
      RSTA(3,I)=0.115
  300 CONTINUE
      KSTA(1)=1
      KSTA(2)=1
      KSTART=1
      KSTOP=1
      KTOT=1
      PRINT*,'ENTER CHOICE OF MAIN INJECTOR OR BAFFLE (M/B):'
      READ(*,'(A)') ANS
      IF(ANS.EQ.'m'.OR.ANS.EQ.'M') THEN
      TITLE='MAIN'
      CALL MAININJ
      ELSE
      TITLE='BAFFLE'
      CALL BAFFLE
      ENDIF
C
C     PRINT RESULTS
C
      ITOT=ISTA(ISTOP)-ISTA(ISTART)+1
      JTOT=JSTA(JSTOP)-JSTA(JSTART)+1
      KTOT=KSTA(KSTOP)-KSTA(KSTART)+1
      NTOT=ITOT*JTOT*KTOT
      DO 600 I=ISTART,ISTOP
      IF(I.GT.1.AND.XSTA(1,I).LT.0.001) GO TO 650
      IST=ISTA(I)-ISTA(ISTART)+1
      WRITE(LPRT,2200) I,IST,XSTA(1,I),RSTA(1,I)
      DO 500 J=2,JSTOP
      WRITE(LPRT,2300) XSTA(J,I),RSTA(J,I)
  500 CONTINUE
      WRITE(LPRT,2350)
  600 CONTINUE
  650 CONTINUE
      IPRT=1
      IF(IPRT.GT.0) THEN
      CALL NODAL
      CALL GRID
      WRITE(LPRT,2400)
      IPD=4
      DO 900 IST=ISTART,ISTOP-1
      I1=ISTA(IST)-ISTA(ISTART)+1
      I2=ISTA(IST+1)-ISTA(ISTART)+1
      IS=I1
      IF(IST.GT.ISTART) IS=I1+1
      DO 900 I=IS,I2,IPD
      DO 800 JST=JSTART,JSTOP-1
      J1=JSTA(JST)-JSTA(JSTART)+1
      J2=JSTA(JST+1)-JSTA(JSTART)+1
      JS=J1
      IF(JST.GT.JSTART) JS=J1+1
      JPD=1
      IF(JST.EQ.6) JPD=3
      DO 800 J=JS,J2,JPD
      DO 700 K=KSTA(2),KSTA(2)
      N=KTOT*(JTOT*(I-1)+(J-1))+K
      WRITE(LPRT,2500) I,J,K,N,X(N),Y(N),Z(N)
  700 CONTINUE
  800 CONTINUE
  900 CONTINUE
```

```
      ENDIF
C
C     CREATE OUTPUT FILE
C
      SCALE=1.0/12.0
      REWIND LTAPE
      WRITE(LTAPE,'(A)') TITLE
      WRITE(LTAPE,2600) ITOT,JTOT,KTOT
      DO 1000 N=1,NTOT
      X(N)=X(N)*SCALE
      Y(N)=Y(N)*SCALE
c     Z(N)=Z(N)*SCALE
      Z(N)=0.0
 1000 CONTINUE
      WRITE(LTAPE,2700) (X(N),N=1,NTOT)
      WRITE(LTAPE,2700) (Y(N),N=1,NTOT)
      WRITE(LTAPE,2700) (Z(N),N=1,NTOT)
      ENDFILE LTAPE
      STOP
 2000 FORMAT(I5)
 2100 FORMAT(3I5,4(2X,E13.6))
 2200 FORMAT('  STATION ',I3,I5,2(2X,G13.6))
 2300 FORMAT(18X,2(2X,G13.6))
 2350 FORMAT(' ')
 2400 FORMAT(/40X,'NODE LOCATIONS',//)
 2500 FORMAT(1X,4(I5,2X),3X,3(E12.5,3X))
 2600 FORMAT(15I5)
 2700 FORMAT(6E13.6)
      END

      SUBROUTINE MAININJ
C
C     MAIN INJECTOR
C
      PARAMETER (LTAPE=3,LPRT=4,numn=12000)
      COMMON/COM1/ ISTA(200),JSTA(100),KSTA(100)
      COMMON/COM2/ XSTA(20,100),RSTA(20,100),AL2(13)
      COMMON/COM3/ X(NUMN),Y(NUMN),Z(NUMN)
      COMMON/COM4/ DELXI(30,3),DELXF(30,3),teta(350,3)
      COMMON/COM5/ PI,RADDEG,A25,A30,A35
      COMMON/COM6/ ITYPE,IROW,ITOT,JTOT,KTOT
      COMMON/COM7/ ISTART,ISTOP,JSTART,JSTOP,KSTART,KSTOP
      OPEN(LTAPE,FILE='MAIN.OUT',STATUS='UNKNOWN')
      OPEN(LPRT,FILE='MAIN.PRT',STATUS='UNKNOWN')
      ITYPE=1
C
C     CALCULATE X AND RADIUS FOR EACH STATION
C
C     REFERENCE POINT (RS009122)
C     MCC SIDE OF PRIMARY FACE PLATE AT X=9.538
C     HOT SIDE OF SECONDARY FACE PLATE AT X=9.538-2.750
```

```
C     X=0.0 AT REFERENCE PLANE -A-
C
      XL2=AL2(IROW)
      XL1=XL2-3.025
      XSTA(1,17)=9.538
      XSTA(1,16)=XSTA(1,17)-0.250
      XSTA(5,12)=XSTA(1,17)-2.750
      XSTA(1,13)=XSTA(5,12)+0.250
C
C     REFER BACK TO DOME BASED ON L2
C     DESCRIBE LOX POST FROM DOME THRU L1
C     INDENTATION IN DOME WHERE POST FITS IS 0.058 INCHES
C
      BASE=XSTA(1,16)-XL2+0.058
      XSTA(1,1)=AL2(13)-XL2
      XSTA(1,2)=BASE-0.058+0.45
      XSTA(1,3)=XSTA(1,2)+0.332
      XSTA(1,4)=XSTA(1,3)+0.250
      XSTA(1,6)=XSTA(5,12)-0.950
      XSTA(1,7)=BASE-0.058+XL1
      XSTA(1,5)=XSTA(1,7)-0.770
      DO 100 J=2,JSTOP
      DO 100 I=1,7
  100 XSTA(J,I)=XSTA(1,I)
      DO 120 I=1,2
      RSTA(4,I)=0.163
      RSTA(5,I)=0.250
  120 CONTINUE
      DO 140 I=3,6
      RSTA(4,I)=0.140
      RSTA(5,I)=0.185
  140 CONTINUE
      RSTA(5,6)=0.2075
C
C     DESCRIBE POST AND RETAINER, FILTER OMITTED
C     HOLES IN RETAINER MODELED AS SLOT WITH SAME FLOW AREA
C     HOLE ENTRANCE AT R=0.245, HOLE AT 30 DEGREES
C
      RAD=0.5*0.142
      AREA=6.0*PI*RAD*RAD
      ROUT=0.2075
      RIN=0.15625
      DO 200 I=7,9
      RSTA(4,I)=RIN
      RSTA(5,I)=ROUT
  200 CONTINUE
      RSTA(4,11)=0.2025
      RSTA(5,12)=0.285
      RA=0.245
      T30=TAN(A30)
      T35=TAN(A35)
```

```
C
C     INLET
C
      XRAMP=(RSTA(5,12)-RSTA(5,8))/T30
      DXA=(RA-RSTA(5,8))*XRAMP/(RSTA(5,12)-RSTA(5,8))
      XSTA(5,11)=XSTA(5,12)-0.100
      XSTA(5,8)=XSTA(5,11)-XRAMP
      XA=XSTA(5,8)+DXA
      DEL=AREA/(4.0*PI*RA*COS(A25))
      DX=DEL*COS(A30)
      DY=DEL*SIN(A30)
      XSTA(5,9)=XA-DX
      RSTA(5,9)=RA-DY
      XSTA(5,10)=XA+DX
      RSTA(5,10)=RA+DY
C
C     EXIT
C
      XRMP=(RSTA(4,11)-RSTA(4,8))/T30
      DX=0.100+XRAMP-0.069-DXA
      DXB=(RA-RIN+DX*T30)/(T30+T35)
      DXB=DXB-DX
      XB=XSTA(5,12)-0.069+DXB
      RB=RSTA(4,8)+DXB*T30
      DEL=AREA/(4.0*PI*RB)
      DD=DEL/COS(A25)
      XSTA(4,10)=XB+DD*COS(A30)
      RSTA(4,10)=RB+DD*SIN(A30)
      B=DXB/COS(A30)
      DEL=DEL-B*COS(A25)
      XSTA(4,9)=XB-DXB-DEL/SIN(A25)
      XSTA(4,8)=XSTA(4,9)-(XSTA(5,9)-XSTA(5,8))
      XSTA(4,11)=XSTA(5,12)+XRMP-0.069
      XSTA(4,12)=XSTA(4,11)+(XSTA(5,12)-XSTA(5,11))
      DO 320 I=8,12
      DO 300 J=1,3
  300 XSTA(J,I)=XSTA(4,I)
  320 CONTINUE
      RSTA(4,12)=RSTA(4,11)
      DO 340 I=11,12
      RSTA(5,I)=0.2850
  340 CONTINUE
C
C     DESCRIBE POST AND SLEEVE (BETWEEN INJECTOR PLATES)
C     AND HOLE IN PRIMARY FACE INJECTOR
C
      XSTA(1,14)=XSTA(1,13)+0.762
      XSTA(1,15)=XSTA(1,14)+1.038
      DO 400 J=2,5
      DO 400 I=13,17
      XSTA(J,I)=XSTA(1,I)
  400 CONTINUE
      DO 420 I=13,17
      RSTA(4,I)=RSTA(4,11)
      RSTA(5,I)=0.2900
      IF(I.GT.14) RSTA(5,I)=0.265
  420 CONTINUE
C
C     NODAL DISTRIBUTION DATA FOR I DIRECTION
C
C
C     READ FILE CONTAINING NODAL DISTRIBUTION DATA FOR I
DIRECTION
C
      OPEN(10,FILE='main01.out')
      READ(10,1000) NSEG
      ISTA(1)=1
      DO 600 N=1,NSEG
      READ(10,1100) M,NUMSEG,ISTA(M+1),XLEN,DXI,DXF,RAT
      DELXI(N,1)=DXI/XLEN
      DELXF(N,1)=DXF/XLEN
  600 CONTINUE
      ISTART=1
      ISTOP=NSEG+1
      CLOSE(10)
C
C     READ FILE CONTAINING NODAL DISTRIBUTION DATA FOR J
DIRECTION
C
      OPEN(10,FILE='main02.out')
      READ(10,1000) NSEG
      JSTA(1)=1
      DO 700 N=1,NSEG
      READ(10,1100) M,NUMSEG,JSTA(M+1),XLEN,DXI,DXF,RAT
      DELXI(N,2)=DXI/XLEN
      DELXF(N,2)=DXF/XLEN
  700 CONTINUE
      CLOSE(10)
      RETURN
 1000 FORMAT(I5)
 1100 FORMAT(3I5,4(2X,E13.6))
      END

      SUBROUTINE BAFFLE
C
C     BAFFLE INJECTOR ELEMENT
C
      PARAMETER (LTAPE=3,LPRT=4,numn=12000)
      COMMON/COM1/ ISTA(200),JSTA(100),KSTA(100)
      COMMON/COM2/ XSTA(20,100),RSTA(20,100),AL2(13)
      COMMON/COM3/ X(NUMN),Y(NUMN),Z(NUMN)
      COMMON/COM4/ DELXI(30,3),DELXF(30,3),teta(350,3)
      COMMON/COM5/ PI,RADDEG,A25,A30,A35
      COMMON/COM6/ ITYPE,IROW,ITOT,JTOT,KTOT
      COMMON/COM7/ ISTART,ISTOP,JSTART,JSTOP,KSTART,KSTOP
      OPEN(LTAPE,FILE='GRID.OUT',STATUS='UNKNOWN')
      OPEN(LPRT,FILE='BAFF.PRT',STATUS='UNKNOWN')
      ITYPE=2
      XL2=AL2(IROW)+2.302
      XL1=AL2(IROW)-3.025
C
C     REFERENCE POINT (RS009122)
C     X=0.0 AT REFERENCE PLANE -A-
C     MCC SIDE OF PRIMARY FACE PLATE AT X=9.538
C     BAFFLE EXTENDS 4.540 INCHES BEYOND PRIMARY INJECTOR
C     BAFFLE POST EXTENDS 11.59 INCHES FROM PLANE -A-
C     BAFFLE TIP EXTENDS 0.15 BEYOND POST
C     HOT SIDE OF SECONDARY FACE PLATE AT X=9.538-2.750
C
      XSTA(5,16)=9.538
      XSTA(5,14)=XSTA(5,16)-0.250
      XSTA(5,8)=XSTA(5,16)-2.750
      XSTA(1,21)=11.59
      XSTA(1,23)=XSTA(1,21)+0.15
C
C     DESCRIBE POST FROM DOME THRU SECONDARY PLATE
C
      BASE=XSTA(1,21)-XL2+0.058
      XSTA(1,1)=AL2(13)+2.302-XL2
      XSTA(1,2)=BASE-0.058+0.45
      XSTA(1,3)=XSTA(1,2)+0.332
      XSTA(1,4)=XSTA(1,3)+0.250
      XSTA(1,5)=BASE-0.058+XL1-0.770
      XSTA(1,6)=XSTA(5,8)-0.950
      XSTA(1,7)=6.6499
      XSTA(1,8)=6.713
      XSTA(1,9)=XSTA(5,8)+0.250
      DO 100 J=2,5
      DO 100 I=1,9
  100 XSTA(J,I)=XSTA(1,I)
      XSTA(5,8)=XSTA(5,16)-2.750
      DO 160 I=1,2
      RSTA(4,I)=0.163
      RSTA(5,I)=0.250
  160 CONTINUE
      DO 170 I=3,5
      RSTA(4,I)=0.140
      RSTA(5,I)=0.185
  170 CONTINUE
      DO 180 I=5,9
  180 RSTA(4,I)=0.164
      RSTA(5,6)=0.2075
      RSTA(5,7)=0.2075
      RSTA(5,8)=0.2735
      RSTA(5,9)=0.249
C
C     DESCRIBE BAFFLE BETWEEN INJECTOR PLATES
C     SLEEVE HAS 512 HOLES WITH DIA=0.017 INCHES THRU WHICH
C     COOLANT GH2 ENTERS ELEMENT
C
      XSTA(1,10)=XSTA(5,8)+0.882
      XSTA(1,11)=XSTA(1,9)+0.841
      XSTA(1,12)=XSTA(1,9)+0.912
      XSTA(1,13)=XSTA(1,12)+1.222
      XSTA(1,14)=XSTA(1,9)+2.277
      DO 210 I=10,13
      DO 210 J=2,5
      XSTA(J,I)=XSTA(1,I)
  210 CONTINUE
      DO 220 J=2,4
  220 XSTA(J,14)=XSTA(1,14)
      DO 230 I=10,14
      RSTA(4,I)=0.1875
      IF(I.EQ.10) RSTA(4,I)=0.164
      RSTA(5,I)=0.249
      IF(I.GT.11) RSTA(5,I)=0.2235
  230 CONTINUE
      RSTA(5,14)=0.249
C
C     STATIONS 15 THRU 17 DESCRIBE THE 8 HOLES IN SLEEVE WITH
C     DIA= 0.092 AND EXITING AT RADIUS 0.229
C
      XSTA(1,15)=9.3696
      XSTA(5,15)=9.3696
      XSTA(1,16)=9.4892
      XSTA(5,16)=9.5380
      XSTA(1,17)=9.6899
      XSTA(5,17)=9.7223
      DO 300 I=15,17
      DO 300 J=2,4
  300 XSTA(J,I)=XSTA(1,I)
      XSTA(4,17)=XSTA(5,17)
      RSTA(4,15)=0.17288
      RSTA(5,15)=0.249
      RSTA(4,16)=0.21175
      RSTA(5,16)=0.3115
      RSTA(3,17)=0.1802
      RSTA(4,17)=0.2875
      RSTA(5,17)=0.3115
C
C     DESCRIBE JACKET,SLEEVE AND POST DOWNSTREAM OF PRIMARY
C     INJECTOR PLATE
C
      REF=XSTA(1,9)+4.540
      RSTA(3,18)=0.237
      RSTA(4,18)=0.2875
      RSTA(5,18)=0.3115
      DO 400 J=1,5
      XSTA(J,18)=REF-1.695
      XSTA(J,19)=REF-0.215
      RSTA(J,19)=RSTA(J,18)
  400 CONTINUE
C
C     DESCRIBE END OF BAFFLE
C
      XSTA(1,20)=REF-0.177
      XSTA(1,22)=11.60823
      XSTA(1,23)=XSTA(1,21)+0.15
      DO 500 I=20,23
      DO 500 J=2,5
      XSTA(J,I)=XSTA(1,I)
  500 CONTINUE
      XSTA(4,21)=11.6058
      XSTA(5,21)=11.6186
      XSTA(4,22)=XSTA(4,21)+0.01823
      XSTA(5,22)=XSTA(5,21)+0.01823
      RSTA(3,19)=0.237
      RSTA(4,19)=0.2875
      RSTA(5,19)=0.3115
      RSTA(3,20)=0.237
      DX=XSTA(1,20)-XSTA(1,19)
      RSTA(4,20)=SQRT(RSTA(4,19)**2-DX*DX)
      RSTA(5,20)=SQRT(RSTA(5,19)**2-DX*DX)
      RSTA(3,21)=0.14300
      DO 520 I=21,23
      RSTA(4,I)=0.154
  520 RSTA(5,I)=0.178
C
C     NODAL DISTRIBUTION DATA FOR I DIRECTION
C
C
C     READ FILE CONTAINING NODAL DISTRIBUTION DATA FOR I
DIRECTION
C
      OPEN(10,FILE='baff01.out')
      READ(10,1000) NSEG
      ISTA(1)=1
      DO 600 N=1,NSEG
      READ(10,1100) M,NUMSEG,ISTA(M+1),XLEN,DXI,DXF,RAT
      DELXI(N,1)=DXI/XLEN
      DELXF(N,1)=DXF/XLEN
  600 CONTINUE
      ISTART=1
      ISTOP=NSEG+1
      CLOSE(10)
```

```
C
C     READ FILE CONTAINING NODAL DISTRIBUTION DATA FOR J
DIRECTION
C
      OPEN(10,FILE='baff02.out')
      READ(10,1000) NSEG
      JSTA(1)=1
      DO 700 N=1,NSEG
      READ(10,1100) M,NUMSEG,JSTA(M+1),XLEN,DXI,DXF,RAT
      DELXI(N,2)=DXI/XLEN
      DELXF(N,2)=DXF/XLEN
  700 CONTINUE
      CLOSE(10)
      RETURN
 1000 FORMAT(I5)
 1100 FORMAT(3I5,4(2X,E13.6))
      END

      SUBROUTINE NODAL
C
C     NODAL DISTRIBUTION (CUBIC STRETCHING)
C
      PARAMETER (LTAPE=3,LPRT=4,numn=12000)
      CHARACTER ANS
      COMMON/COM1/ ISTA(200),JSTA(100),KSTA(100)
      COMMON/COM2/ XSTA(20,100),RSTA(20,100),AL2(13)
      COMMON/COM3/ X(NUMN),Y(NUMN),Z(NUMN)
      COMMON/COM4/ DELXI(30,3),DELXF(30,3),TETA(350,3)
      COMMON/COM5/ PI,RADDEG,A25,A30,A35
      COMMON/COM6/ ITYPE,IROW,ITOT,JTOT,KTOT
      COMMON/COM7/ ISTART,ISTOP,JSTART,JSTOP,KSTART,KSTOP
      DO 500 ID=1,2
      IF(ID.EQ.1) THEN
      LSTART=ISTA(ISTART)
      LSTOP=ISTA(ISTOP)
      ELSE IF(ID.EQ.2) THEN
      LSTART=JSTA(JSTART)
      LSTOP=JSTA(JSTOP)
      ELSE
      LSTART=KSTA(KSTART)
      LSTOP=KSTA(KSTOP)
      ENDIF
      LT=LSTOP-LSTART+1
      TOTL=FLOAT(LT)
      DO 100 L=1,LT
  100 TETA(L,ID)=FLOAT(L-1)/(TOTL-1.0)
      IF(ID.EQ.1) THEN
      LSTART=ISTART
      LSTOP=ISTOP-1
      ELSE IF(ID.EQ.2) THEN
      LSTART=JSTART
```

```
   LSTOP=JSTOP-1
   ELSE
   LSTART=KSTART
   LSTOP=KSTOP-1
   ENDIF
   DO 300 L=LSTART,LSTOP
   IF(ABS(DELXI(L,ID)-DELXF(L,ID)).LT.0.0001) GO TO 300
   IF(ID.EQ.1) THEN
   NI=ISTA(L)-ISTA(LSTART)+1
   NF=ISTA(L+1)-ISTA(LSTART)+1
   ELSE IF(ID.EQ.2) THEN
   NI=JSTA(L)-JSTA(LSTART)+1
   NF=JSTA(L+1)-JSTA(LSTART)+1
   ELSE
   NI=KSTA(L)-KSTA(LSTART)+1
   NF=KSTA(L+1)-KSTA(LSTART)+1
   ENDIF
   DEL=TETA(NF,ID)-TETA(NI,ID)
   NT=NF-NI+1
   DI=DELXI(L,ID)
   DF=DELXF(L,ID)
   IF(NT.EQ.3) THEN
   C=0.5-DI
   D=0.0
   ELSE
   FT=FLOAT(NT)
   FT1=FT-1.0
   FT2=FT-2.0
   FT3=FT-3.0
   C=(3.0-(2.0*FT-3.0)*DI-FT*DF)/(FT2*FT3)
   D=(DI+DF-2.0/FT1)/(FT2*FT3)
   ENDIF
   DO 200 N=2,NT-1
   M=N+NI-1
   FN=FLOAT(N)
   ETA=(FN-1.0)*(DI+(FN-2.0)*(C+D*FN))
   TETA(M,ID)=TETA(NI,ID)+ETA*(TETA(NF,ID)-TETA(NI,ID))
 200 CONTINUE
 300 CONTINUE
 500 CONTINUE
   RETURN
   END

   SUBROUTINE GRID
C
C  GENERATE GRID
C
   PARAMETER (LTAPE=3,LPRT=4,numn=12000)
   CHARACTER ANS
   COMMON/COM1/ ISTA(200),JSTA(100),KSTA(100)
   COMMON/COM2/ XSTA(20,100),RSTA(20,100),AL2(13)
```

```
      COMMON/COM3/ X(NUMN),Y(NUMN),Z(NUMN)
      COMMON/COM4/ DELXI(30,3),DELXF(30,3),teta(350,3)
      COMMON/COM5/ PI,RADDEG,A25,A30,A35
      COMMON/COM6/ ITYPE,IROW,ITOT,JTOT,KTOT
      COMMON/COM7/ ISTART,ISTOP,JSTART,JSTOP,KSTART,KSTOP
      DO 1000 IST=ISTART,ISTOP-1
      IST1=IST+1
      I1=ISTA(IST)-ISTA(ISTART)+1
      I2=ISTA(IST1)-ISTA(ISTART)+1
      IS=I1
      IF(IST.GT.ISTART) IS=IS+1
      DO 1000 I=IS,I2
      EPS=(TETA(I,1)-TETA(I1,1))/(TETA(I2,1)-TETA(I1,1))
      DO 900 JST=JSTART,JSTOP-1
      JST1=JST+1
      J1=JSTA(JST)-JSTA(JSTART)+1
      J2=JSTA(JST1)-JSTA(JSTART)+1
      JS=J1
      IF(JST.GT.JSTART) JS=JS+1
      DO 900 J=JS,J2
      ETA=(TETA(J,2)-TETA(J1,2))/(TETA(J2,2)-TETA(J1,2))
      IF(ITYPE.EQ.1) GO TO 100
      IF(IST.NE.20) GO TO 100
      IF(JST.EQ.1) GO TO 100
      GO TO 200
  100 CONTINUE
      X1=(1.0-EPS)*XSTA(JST,IST)+EPS*XSTA(JST,IST1)
      X3=(1.0-EPS)*XSTA(JST1,IST)+EPS*XSTA(JST1,IST1)
      R1=(1.0-EPS)*RSTA(JST,IST)+EPS*RSTA(JST,IST1)
      R3=(1.0-EPS)*RSTA(JST1,IST)+EPS*RSTA(JST1,IST1)
      GO TO 500
  200 CONTINUE
C
C     COMPUTE CIRCULAR ARC FOR EDGE 1
C
      IF(JST.EQ.2) THEN
      X1=(1.0-EPS)*XSTA(JST,IST)+EPS*XSTA(JST,IST1)
      R1=(1.0-EPS)*RSTA(JST,IST)+EPS*RSTA(JST,IST1)
      ELSE
      XCURV=XSTA(1,19)
      IF(JST.EQ.3) XCURV=XSTA(1,20)
      RCURV=RSTA(JST,19)
      DX=XSTA(1,20)-XCURV
      ANG1=ACOS(DX/RCURV)
      DX=XSTA(JST,IST1)-XCURV
      ANG2=ACOS(DX/RCURV)
      ANG=(1.0-EPS)*ANG1+EPS*ANG2
      X1=XCURV+RCURV*COS(ANG)
      R1=RCURV*SIN(ANG)
      ENDIF
C
```

```
C     COMPUTE CIRCULAR ARC FOR EDGE 3
C
      IF(JST.EQ.6) THEN
      X3=(1.0-EPS)*XSTA(JST1,IST)+EPS*XSTA(JST1,IST1)
      R3=(1.0-EPS)*RSTA(JST1,IST)+EPS*RSTA(JST1,IST1)
      ELSE
      XCURV=XSTA(1,19)
      IF(JST.EQ.2) XCURV=XSTA(1,20)
      RCURV=RSTA(JST1,19)
      DX=XSTA(1,20)-XCURV
      ANG1=ACOS(DX/RCURV)
      DX=XSTA(JST1,IST1)-XCURV
      ANG2=ACOS(DX/RCURV)
      ANG=(1.0-EPS)*ANG1+EPS*ANG2
      X3=XCURV+RCURV*COS(ANG)
      R3=RCURV*SIN(ANG)
      ENDIF
  500 CONTINUE
      XX=(1.0-ETA)*X1+ETA*X3
      RR=(1.0-ETA)*R1+ETA*R3
      DO 800 K=1,KTOT
c     PSI=TETA(K,3)
c     ANG=PI*PSI
      N=KTOT*(JTOT*(I-1)+(J-1))+K
      X(N)=XX
c     Y(N)=RR*SIN(ANG)
      Y(N)=RR
      IF(ABS(Y(N)).LT.0.00001) Y(N)=0.0
c     Z(N)=-RR*COS(ANG)
c     IF(ABS(Z(N)).LT.0.00001) Z(N)=0.0
  800 CONTINUE
  900 CONTINUE
 1000 CONTINUE
      RETURN
      END
```

APPENDIX B

Implementation of Adaptive Methodology in the General Purpose CFD Code FDNS

Implementation of Adaptive Methodology in the General Purpose CFD Code FDNS

Prepared for

SECA

3313 Bob Wallace Ave.

Suite 202

Huntsville, AL 35805

Mississippi State University
NSF Engineering Research Center for
Computational Field Simulation
P. O. Box 6176
Mississippi State, MS 39762

Table of Contents

Page

List of Figures

I. Introduction

The accurate numerical simulation of fluid flows in complex geometries, such as exist in components of advanced lift systems, requires the construction of well–designed grids upon which to obtain solutions to the governing partial differential equations. This is never a trivial task, and in fact requires considerable ingenuity, as well as a basic understanding of the physics of the flow involved. Since the flow solution is not, in general, known *a priori*, the computational grid may be far from optimal and can induce significant errors because of the dependence of truncation error on grid spacing, smoothness, and skewness.

This is the impetus for the development of adaptive grid methodologies in computational fluid dynamics (CFD). Adaptive grids are those which adjust to the evolving flow solution, usually in response to some measure of solution error, or to derivative functions in the solution, such as gradients, curl, or curvature. The idea is that grid points will be moved into regions of large error or gradients, and away from smooth regions where the points are not needed, thereby reducing the overall truncation error. Adaptive strategies may also include the addition of points to the grid rather than the redistribution of existing points. In principle, either such a local refinement approach or the point migration approach can result in better CFD solutions at the same or reduced computational costs.

The goal of this project was to enable the general pupose CFD code , FDNS, to be able to utililize adaptive methods to increase the accuracy of the solutions. As originally proposed, the foundation for this code is the EAGLE [1–4] grid generation system. For purposes of development, validation, and application by SECA, this adaptive version of EAGLE was coupled with FDNS [5], the Finite Difference Navier Stokes Solver developed by Mr. Y.S. Chen.

Originally developed at Mississippi State University under U. S. Air Force sponsorship, EAGLE is a set of two computer programs written in Fortran. One code is for the geometric construction of grid boundary surfaces, either from data or from specified functions. The second generates volume or surface grids by algebraic interpolation from boundaries or by the numerical solution of systems of elliptic partial differential equations in which the Cartesian spatial coordinates are the dependent variables. The adaptive version adjusts the grid by moving points in response to user–specified weight functions, such as fluid vorticity or gradients in the dependent variables. In addition, weight functions may be constructed to contain measures of grid quality, such as skewness or aspect ratio, so that the resulting grid will have points concentrated in regions of high skewness or aspect ratio so that error induced by these defects can thus be reduced.

EAGLE is based upon the composite–block principle so that large and geometrically complex flow configurations can be broken down into smaller components which can then be filled with smooth, continuous grids. Complete or lesser continuity of grid lines can be assured across these block interfaces through the use of overlapping layers of points. This multi–block approach not only allows arbitrarily shaped domains to be treated, but also enables the computational simulation of very large problems, since only one block of the grid and solution data need occupy central memory at any time.

For the large, geometrically complex regions which must be treated in the simulation of fluid combustion and heat transfer processes using FDNS, the multi–block approach to grid generation and adaptation was deemed essential to the success of this project.

In this work, the control function approach is the mechanism for adapting grids, as detailed in References 7–8, where preliminary work is thoroughly documented. In contrast to those earlier efforts, the elliptic grid generation procedure has been separated from the main grid code in the present work. The elliptic grid routine now can be called either by the flow solver to generate a new adaptive grid based on flow variables and quality measures through dynamic adaptation, or by the grid code itself to generate a grid based on quality measures through static adaptation. In addition, an existing flow solution can be read in and an existing grid adapted to that solution, so that subsequent solutions can be started on a more nearly optimum grid. In any case, communication between the grid generation system and FDNS is through scratch files, so that only minimal changes had to be made to the source code.

The adaptive mechanism includes adaptation to either the gradient of a variable, as in the original case [7,8], to the curvature of a variable, or to the variable itself. The variable, of course, can be whatever the user wishes it to be; the use of the term "variable" does not imply that only dependent flow variables can be used. In addition, the mechanism also includes the ability to calculate the weight functions as weighted averages of weight functions from several variables, as well as grid quality measures.

The adaptation can also take into account the effect of many of the solution variables, instead of just one, and provides for different weight functions in each coordinate direction. The construction of the weighted average of flow variables and quality measures, and the choice of adaptation to gradient, curvature, or variable are all controlled in each coordinate direction through user input. The quality measures currently available in the system are skewness, aspect ratio, spacing, and smoothness of the grid as measured by the grid Laplacian.

II. Elliptic Grid Generation

Among the various techniques for generating grids with partial differential equations, sets of elliptic equations derived from the Laplace or Poisson equations are the most common. As can be shown from the variational calculus, the Laplace system produces the smoothest possible grid since when it is satisfied, the grid is uniformly spaced. Therefore, even with non–uniform boundary distributions, the coordinate lines in the interior of the field tend to be equally spaced. Control of the coordinate line distribution in the field can be obtained with an elliptic system derived from the Poisson equations,

$$\nabla^2 \xi^i = P^i, \quad i = 1, 2, 3, \tag{2.1}$$

where the functions p^i serve to control the coordinate line spacing.

Warsi [9] has shown that if a curvilinear coordinate system $\bar{\xi}^i$, which satisfies the Laplace system $\nabla^2 \bar{\xi}^i = 0$, is transformed to another coordinate system ξ^i, then the new curvilinear coordinates ξ^i satisfy the inhomogeneous elliptic system as defined by Equation (2.1) with the control functions

$$P^i = \sum_{j=1}^{3} \sum_{k=1}^{3} g^{jk} P^i gk , \quad i = i, 2, 3, \tag{2.2}$$

with the P^i_{jk} defined by the transformation from $\bar{\xi}^i$ to ξ^i by

$$P^i_{jk} = \sum_{m=1}^{3} \sum_{n=1}^{3} \frac{\partial \bar{\xi}^m}{\partial \xi^j} \frac{\partial \bar{\xi}^n}{\partial \xi^k} \frac{\partial^2 \xi^i}{\partial \bar{\xi}^m \partial \bar{\xi}^n} \tag{2.3}$$

and

$$g^{ik} = \frac{1}{g} \left(g_{jm}\, g_{ln} - g_{jn}\, g_{lm} \right)$$

with *(i, j, l), (k, m, n)* cyclic. Here g is the square of the Jacobian of the transformation and $g_{ij} = r^i_{\xi} \bullet r^j_{\xi}$ are the elements of the covariant metric tensor.

In these relations, $r = x\mathbf{i} + y\mathbf{j} + z\mathbf{k}$ is the Cartesian position vector of a grid point and ξ^i, i = 1, 2, 3, are the three curvilinear coordinates. The combination of Equations (2.1) and (2.2) gives

$$\nabla^2 \xi^i = \sum_{j=1}^{3} \sum_{k=1}^{3} g^{jk} P^i_{jk} , \quad i = 1, 2, 3 \tag{2.4}$$

These results show that a coordinate system obtained by application of a stretching transformation to one generated as the solution of the Laplace system can be generated directly by solving Equation (2.4) with appropriate control functions P^i_{jk} as defined by Equation (2.3). Therefore, the Poisson system [Equation (2.1)] can be taken as the generation system with the control functions considered to be specified. Among these control functions, P^i_{ii} *(i = 1, 2, 3)*, are the most important since correspond to one–dimensional stretching in each coordinate direction. By taking all the other control functions to be zero, $P^i_{jk} = \delta^j_k \, \delta^i_k \, P_i$, Equation (2.4) becomes

$$\nabla^2 \xi^i = g^{ii} P_i \, , \quad i = 1, 2, 3 \tag{2.5}$$

Transformation of Equation (2.5) into curvilinear space then yields

$$\sum_{i=1}^{3} \sum_{j=1}^{3} g^{ij} \, r_{\xi^i \xi^j} + \sum_{k=1}^{3} g^{kk} \, P_k \, r \xi^k = 0 \tag{2.6}$$

which is the form commonly used. The spacing and orientation of grid lines in the field are controlled by the control functions P_i. For example, a negative value of P_i causes the coordinate lines to concentrate in the direction of decreasing ξ^i. Control functions can be evaluated such that the grid generated by Equation (2.6) reflects the spacing of some initial grid, generated perhaps by algebraic methods. Or they may be based upon boundary properties, such as spacing and curvature, so that these are projected into the field. Procedures for the determination of control functions in the EAGLE code are discussed in Reference 1.

III. Adaptive Grid Generation

3.1 Variational Approach

Minimization of the integral of some grid property over the computational domain is known as the variational approach. The resulting Euler variational equations from the calculus of variations then constitute the grid generation system. The choice of what property is to be minimized depends upon what is expected from the grid. For example, Saltzman and Brackbill [10] developed adaptive grids by minimizing a weighted combination of integrals which emphasize smoothness, orthogonality, and point concentration. A similar approach developed by considering smoothness, a measure of the grid cell area, and the orthogonality of the grid lines can be found in Reference 11. Several other grid properties that might be considered, such as the square of cell volume, inverse cell volume, are discussed by Thompson and Warsi [12].

However, due to the complexity of the Euler equations, they are difficult to solve, and solution algorithms may not converge. A survey of the types of integrals that may that be included in a variational problem, and the geometric properties that each integral imposes upon the grid, can be found in Reference 13. In this work, the variational approach was found to require an order of magnitude more computational time than the control function approach.

3.2 Control Function Approach

The control function approach to adaptation is developed by nothing the correspondence between the one–dimensional form of Equation (2.6),

$$x_{\xi\xi} + PX_{\xi} = 0 \tag{3.1}$$

and the differentiated form of the equidistribution principle, W_ξ = constant,

$$Wx_{\xi\xi} + W_{\xi}x_{\xi} = 0 \tag{3.2}$$

where P is the function to control the coordinate line spacing, and W is some weight function.

From Equation (3.1) and (3.2), the control function can be defined in terms of the weight function and its derivative as

$$P = \frac{W_{\xi}}{W} \tag{3.3}$$

This equation can be expressed in a general three–dimensional form as

$$P_i = \frac{W_{\xi^i}}{W} \tag{3.4}$$

This approach was developed by Anderson [14,15] and has been applied with success in two–dimensional configurations by Johnson and Thompson [16] and in three–dimensional configurations by Kim and Thompson [8], and by Tu and Thompson [7].

The complete generalization of Equation (3.4) was proposed by Eiseman[17] as

$$P_i = \sum_{j=1}^{3} \frac{g^{ij}}{g^{ii}} \frac{(W_i)_{\xi^i}}{W_i} , \tag{3.5}$$

where W_i is the weight function chosen for the ξ_i direction. This definition of the control functions provides a convenient means of specifying three separate functions, with one in each coordinate direction.

In order to preserve the geometric characteristics of the existing grid, the control functions are constructed so that those defined by Equation (3.5) can be added to the initial set, thus giving the grid a memory. To wit,[7]

$$P_i = (P_i)_g + C_i (P_i)_W , \quad i = 1, 2, 3, \tag{3.6}$$

where:

$(P_i)_g$ = *control function based on geometry*
$(P_i)_W$ = *control function based on weight function*
(C_i) = *weight coefficient to be specified*

In these equations the weight function W is defined according to what one wishes to adapt to. The geometric contribution $(P_i)_g$ can be based upon either the starting grid or the previous adapted grid, with the latter approach bringing greater deviation from the original grid. For adaptation to:

$$\textit{Variable;} \quad W = 1 + |V| \tag{3.7a}$$

$$\textit{Gradient;} \quad W = 1 + |\nabla V| \tag{3.7b}$$

$$\textit{Curvature;} \quad W = 1 + (1 + \beta|K|) \sqrt{1 + \alpha|\nabla V|^2} \tag{3.7c}$$

where V can be any scalar component of the solution, some derivative of the solution, such as the magnitude of vorticity, a grid quality measure, or some measure of truncation error. Here β, $\alpha \in [0, 1]$ and

$$K = \frac{\nabla^2 V}{(1 + |\nabla V|^2)^{\frac{3}{2}}} \tag{3.8}$$

is the usual definition of the curvature of V.

With the control functions defined in this way, the elliptic generation system given by Equation (2.6) becomes an adaptive system. It is then solved iteratively in this work by point SOR to generate the adapted grid. The control function approach to grid adaptation can be obtained directly from a variational principle, as is shown in Reference 11.

Although the obvious choice for the adaptive variable V is some component of the solution, such as velocity, pressure, or species concentration, the form of the adaptive mechanism does not constrain the choice. V could be defined as the vorticity, for example, or as the product of vorticity and velocity. Another choice might be an estimate of local solution error, as outlined in Reference 18. This choice has the advantage of limiting grid movement, since as the error is reduced, the adaptive control function is reduced and the grid stabilizes. Unfortunately, reliable measures of solution error tend to be difficult and costly to implement.

Another choice for the adaptive weight function is one based on measures of grid quality. Grids can be evaluated quantitatively by the computation of certain properties, such as skewness, aspect ratio, and stretching. These are known to affect solution accuracy. An adaptive methodology that effectively reduces undesirable grid qualities, however, requires the application of control laws through the variational approach, which has been found to be very expensive computationally. As an alternative, grid properties can be used in the control function approach to simply concentrate grid lines in regions of high skewness or aspect ratio, thus, in principle, reducing the overall truncation error. This capability is included in this work. Four grid quality measures are available to be used for grid adaptation. Although they can be used alone to statically adapt an existing grid, the intent is that they be combined with solution variables in obtaining adaptive grid solutions. The quality measures available are skew angle, aspect ratio, grid Laplacian (a measure of grid smoothness), and arc length, or the local rate at which grid spacing changes.

The minimum skew angle between intersecting grid lines is one of the most important measurable grid properties. This angle can be expressed in terms of the covariant metric elements as

$$\theta_{ij} = \cos^{-1}\left\{\frac{g_{ij}}{\sqrt{g_{ii}\, g_{jj}}}\right\} \tag{3.9}$$

Since $g_{12} = g_{21}$, $g_{13} = g_{31}$ and $g_{23} = g_{32}$, the three skew angles associated with each grid point in a 3–D grid are Θ_{12}, Θ_{23}, and Θ_{31}. The choice of the minimum as opposed to the maximum angle is arbitrary.

Clearly, aspect ratio can be defined in two different ways on any coordinate surface. For example, on a surface of constant ξ^k, the ratio can be expressed in terms of metric elements g_{ii} and g_{jj} as

$$AR_{ij} = \sqrt{g_{ii} \;/\; g_{jj}} \tag{3.10}$$

Large changes in aspect ratio from one part of the field to another are known to inhibit convergence.

A measure of the smoothness of a grid is the Laplacian of the curvilinear system, which is simply the rate of change of grid point density. For a perfectly uniform grid, the grid Laplacian would vanish everywhere, but exceedingly large values may arise in highly stretched grids. When a coordinate transformation is applied so that the Cartesian coordinates are the dependent variables, the grid Laplacian is given in terms of the contravariant metric elements g^{ij}, the contravariant base vectors a^l, and the position vector **r** as

$$\nabla^2 \xi^l = -\sum_{i=1}^{3} \sum_{j=1}^{3} g^{ij} a^l \cdot r_{\xi i \xi j} \qquad l = 1, 2, 3 \tag{3.11}$$

Another important measure of grid quality is the local rate at which grid spacing changes. On a coordinate surface of constant ξ^k, and along a coordinate line of constant ξ^j, the grid spacing is just

$$d_i = [(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2 (z_{i+1} - z_i)^2]^{\frac{1}{2}} \tag{3.12a}$$

The rate at which grid spacing changes (ARCL) is then just

$$(ARCL)_i = \frac{d_i - d_{i-1}}{\frac{1}{2}(d_i + d_{i-1})} \tag{3.12b}$$

IV. Adaptive FDNS

Over the past several years, adaptive grid methods have been applied to compressible flow problems with some success. For example, adaptive EAGLE has been coupled with the MISSE [19] Euler equation solver to produce better resolved shock waves on coarse grids at reduced computational cost. Another application of adaptive EAGLE was in the solution of incompressible problems using INS3D [6] as the CFD software. Clearly, the steepest gradients in incompressible viscous flows occur in the boundary layers, which predictably occur on physical boundaries where the no–slip velocity condition is applicable. This being the case, it is comparatively easy to generate grids which will resolve boundary layers, without any application of adaptive methodologies. In fact, flow solutions may simply diverge if boundary shear layers are not adequately resolved from the beginning of computations. In other words, in viscous flows, the grid simply must be "adapted" to the anticipated solution before any solution can be obtained.

However, viscous shear layers are not the only features of a flow which may profit from adaptive gridding. Recirculating regions may develop in unanticipated regions; mixing zones near injections sites may not be known in advance; species concentrations in reacting flows may migrate as the solution develops. In addition, the thickness of boundary layers cannot be well anticipated in many flows, so that an adaptive grid my be useful in moving points to better resolve the flow near surfaces. This is particularly true in turbulent flows, where it is critical that grid spacing near the wall be adequate to resolve the laminar sub–layer, whose thickness cannot be generally predicted in advance. In such cases, very small adjustments in the grid can result in significant changes in the solution, and these adjustments may not even be visible to the un–aided eye. This is in contrast to adaptation in the region of shock waves in where the clustering of points is dramatic.

Converting a code such as FDNS to one capable of operating on solution adaptive grids generated by the adaptive EAGLE grid system is a relatively simple procedure, since the principal linkage between the codes is a set of scratch files which each code reads and writes. True dynamic grid adaptation requires that additional unsteady terms be added to the discretized fluid flow equations to account for grid motion relative to the flow field. While this is certainly a reasonable thing to do, it is a clear impediment to rapid conversion of an existing code into an adaptive solver and is not necessary for convergence to steady–state. The addition of the terms requires changes to the Fortran source code far greater than those needed to link the solver with EAGLE.

As an alternative to dynamic adaptation, the technique of multiple, periodic grid adaptation was adopted in this work. In this approach, a new grid need not be produced at each time step, and no significant code modifications are required. In this periodic mode, the solution can, of course, be interpolated from the old grid to the new grid if desired, but since the grid motion is usually quite small, it has been found that simply using the existing solution as starting values on the new grid, with no interpolation, is sufficient for steady state applications. As will be shown in the results presented herein, this approach affects the overall convergence rate very little. However, when the transient solution process is of interest it is desirable to do the interpolation, but this in no way affects the functioning of the grid generation system. It merely requires additional subroutines to perform the interpolation after each adaptation.

The EAGLE grid generation system, upon which the adaptive code is built, uses a script file to supply the boundary information as well as any other information it needs to create the grid. The adaptive version relies on the same methods to get its information for grid construction purposes. The extensions to the regular EAGLE namelist commands can be found in Appendicies A and B.

The adaption process can be approached in two ways. The first of these is often referred to as static adaption with the second being referred to as dynamic adaption. The static adaption process is the easiest and non–intrusive way of implementing the adaptive methods. In this methodology the CFD code is started and run until a predetermined time step is reached or convergence. The grid and solution files are then readied for static adaption. The static adaption is handled by a stand alone version of the adaptive EAGLE code. This code requires that the grid file be written to disk in a triad format with no header information. The executable version reads an adaptive script file that determines how the adaption is to take place and what files are to be used. The grid is then read into the adaptive code and by applying the boundary conditions along with the adaptive parameters (supplied by the user) and the current solution file, the adaptive mesh is generated and stored to disk in PlOT3D format. This file can then be read back into the CFD code in the restart process and a solution continued from that point foward. The solution in this process is not interpolated since the usual desired result is a steady state solution. The error caused by applying the old solution over the new grid is damped out in the first few iterations after restart and therefore the error in the steady state solution will be non–existent. An example of the static adaption script file is included in Appendix C.

The second method of performing the adaption is that which is referred to as dynamic. In this method the parent CFD code has to undergo some minor modifications to allow for the adaptive routines to be incorporated directly into the solution process. The basic method used for inclusion of the adaptive source code into the FDNS code is shown in Appendix D. This method also differs from the static version in that the code does not have to be stopped and restarted to accomplish the adaption. The adaption is done on the fly during the solution process. One of the modifications that is applied to the parent code is that of including the logic necessary to allow the code to decide when the adaption needs to take place based on user inputs from a namelist file. An example of this file is shown in Appendx E. The dynamic process is very useful in that ,based on the users input, the code, without stopping, can generate the new adapted grid and continue on with the solution process. This method once set up and started requires no intervening steps on the users part to generate the adaptive grid.

The adaptive version of multi–block FDNS differs very little from the non–adaptive one. In fact, since it was very evident that the FDNS code was constructed from the outset to be as compartmentalized as possible, the addtions to the FDNS source code that needed to be made were constructed with that same goal in mind. The only alteration that the parent code had to undergo was the addition of three #ifdef constructs in the main source code file fdns.f. The #ifdef constructs serve the purpose of only including the adaptive code if the make command is issued with the argument adapt, ie. make adapt. With this method the code can be made clean of the adaptive commands and therfore have an unaltered code without having to change any source files.. The use of the #ifdef constructs also serve the purpose of allowing the user to include the files at whatever location in the source they wish. The limitations of course are the adaptive commands need to be included in the sections where they are relavent. The first #ifdef construct is responsible for the inclusion of the definitions needed by the adaptive code. This would necessarily need to be included in the same general area of the code as FDNS's definition statements. The second #ifdef construct is responsible for the inclusion of the decision logic which determines when or if to adapt. This construct would be located at some point in the time iteration loop of the main code. The current location is after the computations and prior to the next loop but could easily be moved to another location inside the loop. The final #ifdef construct is responsible for the inclusion of all the subroutines that will be needed to enable the code to dynamically adapt the grid to the solution and handle the scratch file transfers. The example locations of these three constructs are provided in Appendix G with the #ifdef constructs in bold type.

V. Computed Results

The test case that was chosen for this report was that of a 2D flat plate undergoing blown combustion of an oxygen buteyne mixture. The grid was a rectangular mesh with inlet/outlet boundaries as noted in Appendix F. The first case run was a 161–81 mesh size and this was used as the baseline reference. The solution was run for 20,000 iterations and the erruvw term dropped from 1.2836x10^–3 to 1.64x10^–5 which represented a 2 order of magnitude drop and was deemed sufficient for convergence. The baseline grid was made dense in the j direction in order to fully pick up the flow phenomena and represent as accurate a solution as possible for a baseline case. The grid is shown in Figure 1 with the resulting density field represented in Figure 2. Since most of the mixing and action occurred at the inlet to the problem this is what is depicted in all the pictures. The next set of figures show the result of halving the points in the j direction. With only half the points the solution suffers in that the flow phenomena is not as clearly defined. The erruvw term for the 161–41 case was 1.84x10^–5. While this error term also reached the 10^–5 threshold it is evident that the halving of points in the j direction had a detrimental effect on the convergence of the solution. Figure 3 shows the sparse grid while Figure 4 represents the inlet portion of the density field.

The adaption on the 161–41 sparse grid was accomplished through the static method. The solution variable that was decided upon to adapt to was density. The grid was adapted to the variable of density in the i direction and the gradient of density in the j direction. This decision was again made due to the fact that the most action was occurring in the j coordinate direction. The adaption to the variable in the i coordinate direction had the effect of smoothing the adaption. The values of the user chosen weight coefficients were 3.5 and 0.6 in the i and j directions respectively. The values of these were also chosen with respect to the guidelines mentioned above for the weight function choices. The grid was reloaded into FDNS using the codes restart capability and run for an extra 1000 iterations to smooth the error out caused by the current lack of interpolation of the solution to the new grid. The movement of the grid is evident in Figure 5. The grid lines were pulled toward the density gradient so as better to capture the local phenomena. The aligning of the grid lines with the density gradient is very evident in Figure 6 which plots the grid lines as scalar valued colored lines. The effect of this bunching of lines along the density gradient phenomena serves to reduce the truncation error associated with field discretization which results in a better solution to the equations. This increase in accuracy can be seen in the error term as well as Figure 7 which shows the density gradient being much more sharply defined than before. The error term was reduced to the value of 1.26x10^–5 which showed a significant increase in accuracy.

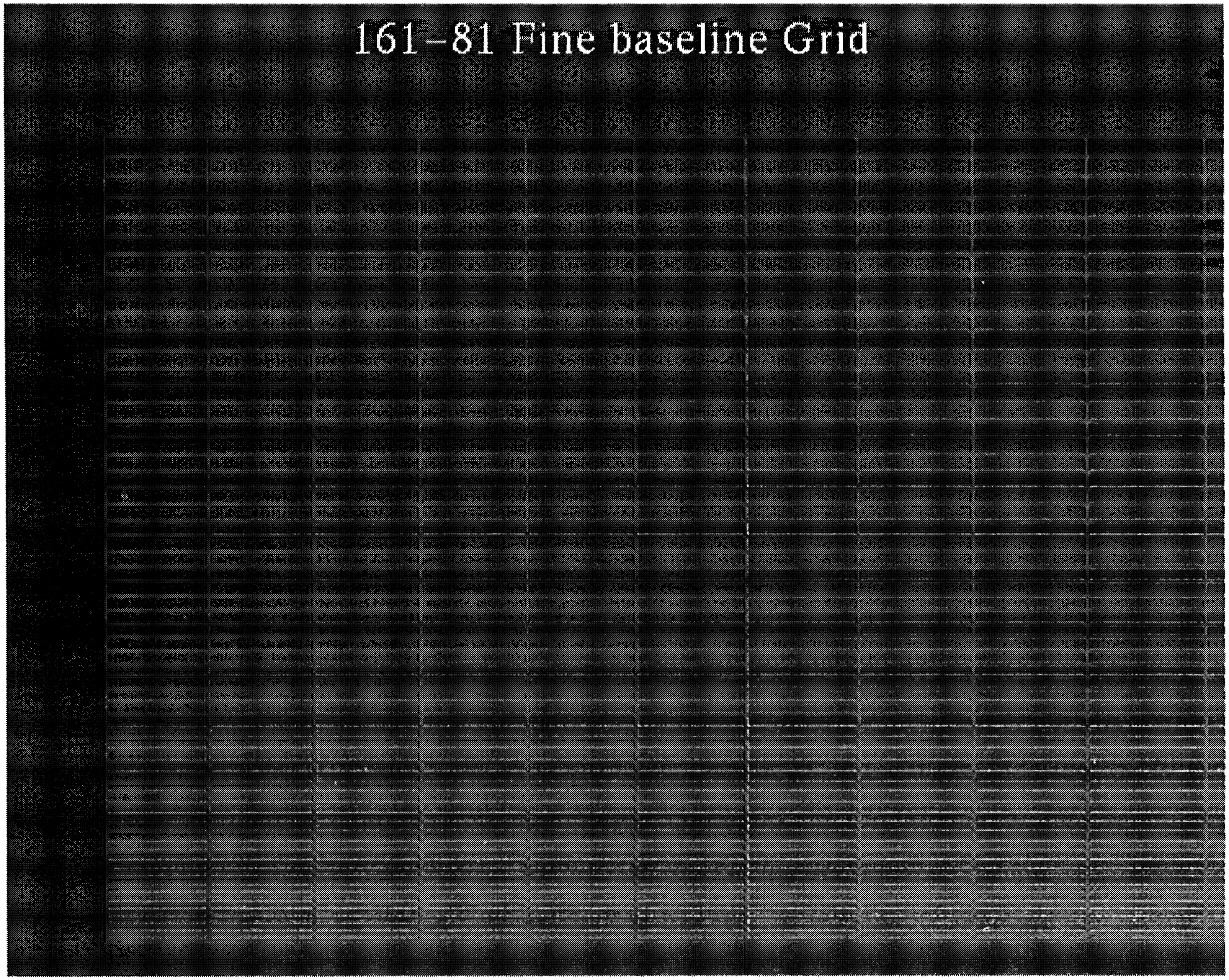


Figure 1

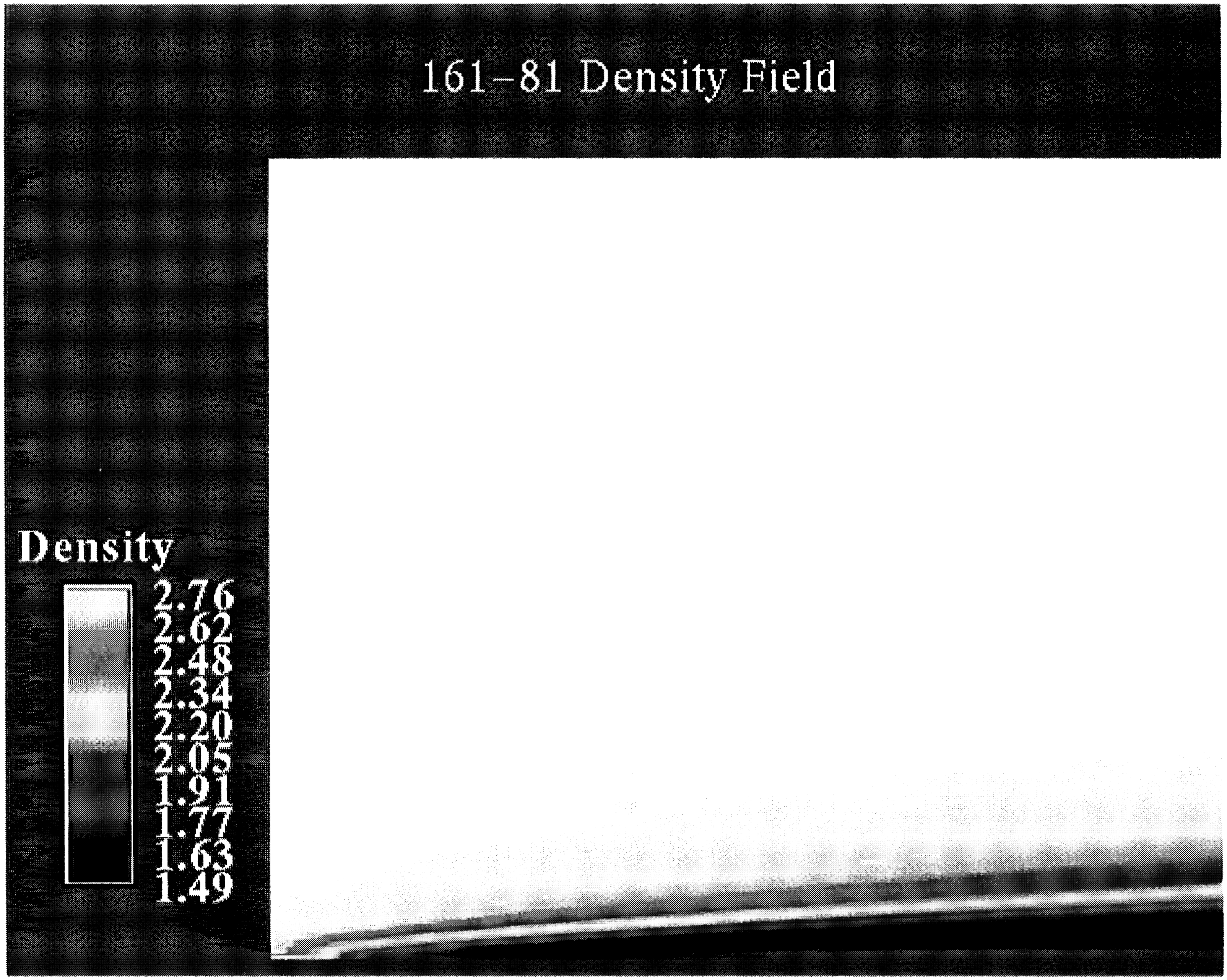

161–81 Density Field
Density
2.76
2.62
2.48
2.34
2.20
2.05
1.91
1.77
1.63
1.49

Figure 2

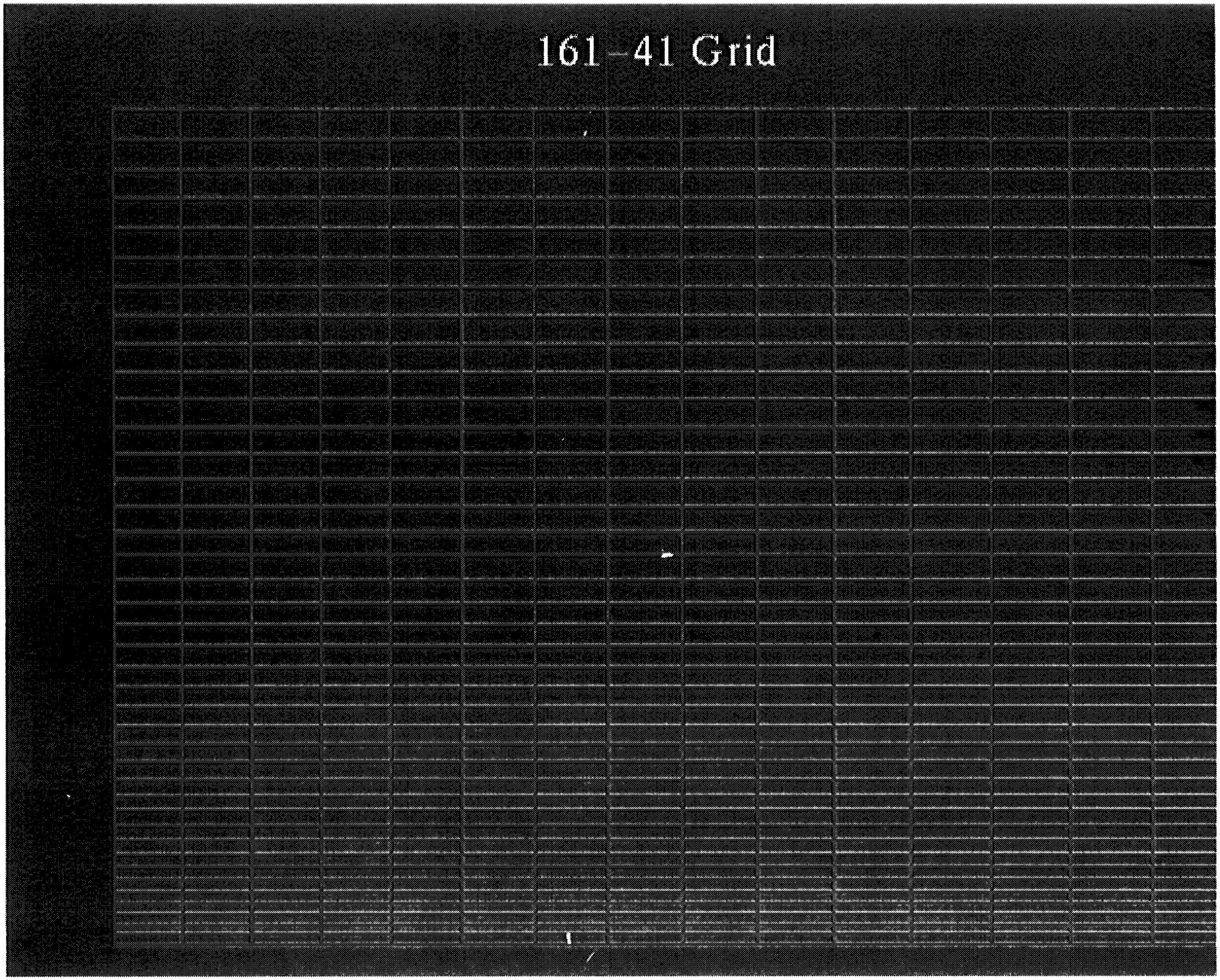


Figure 3

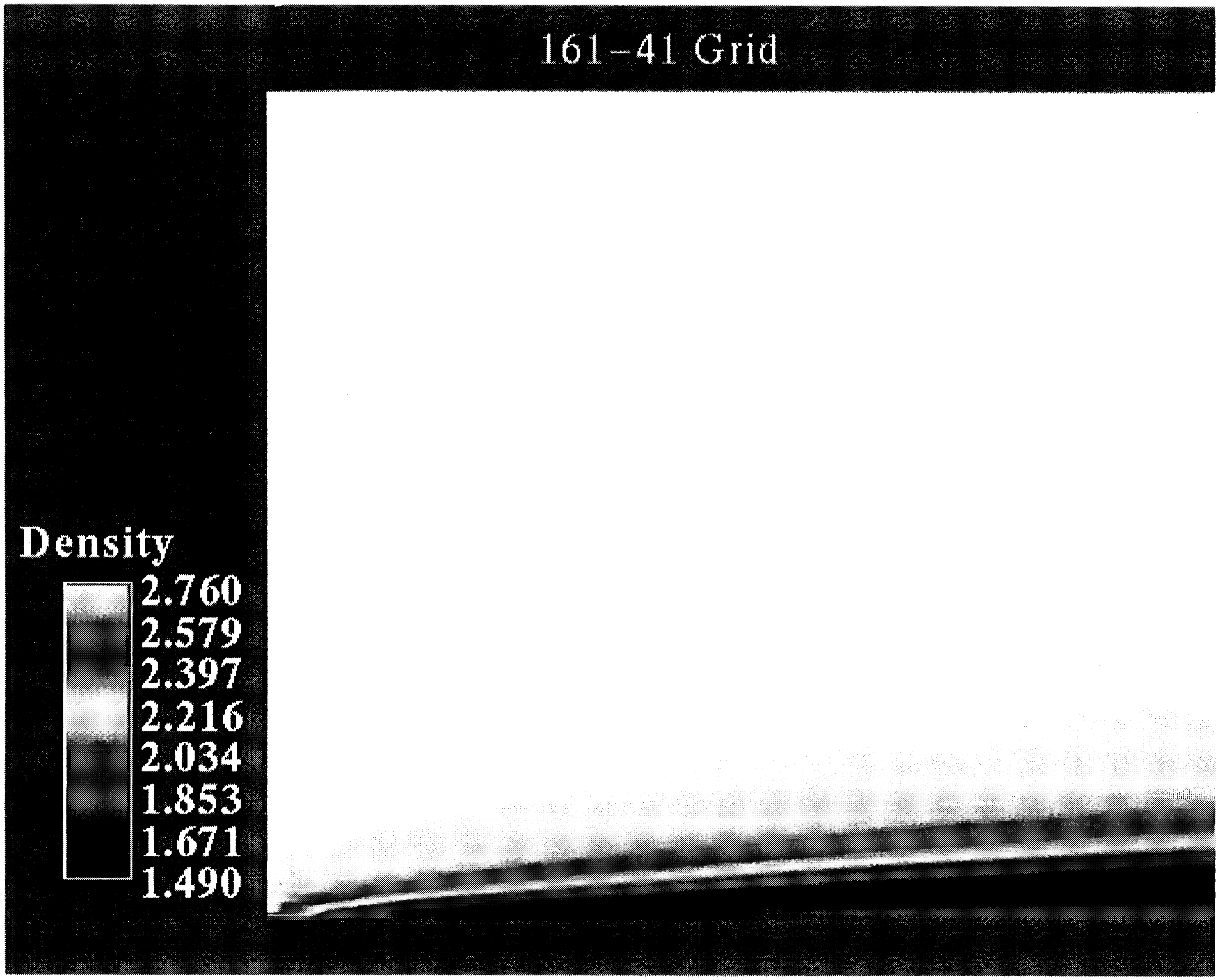


Figure 4

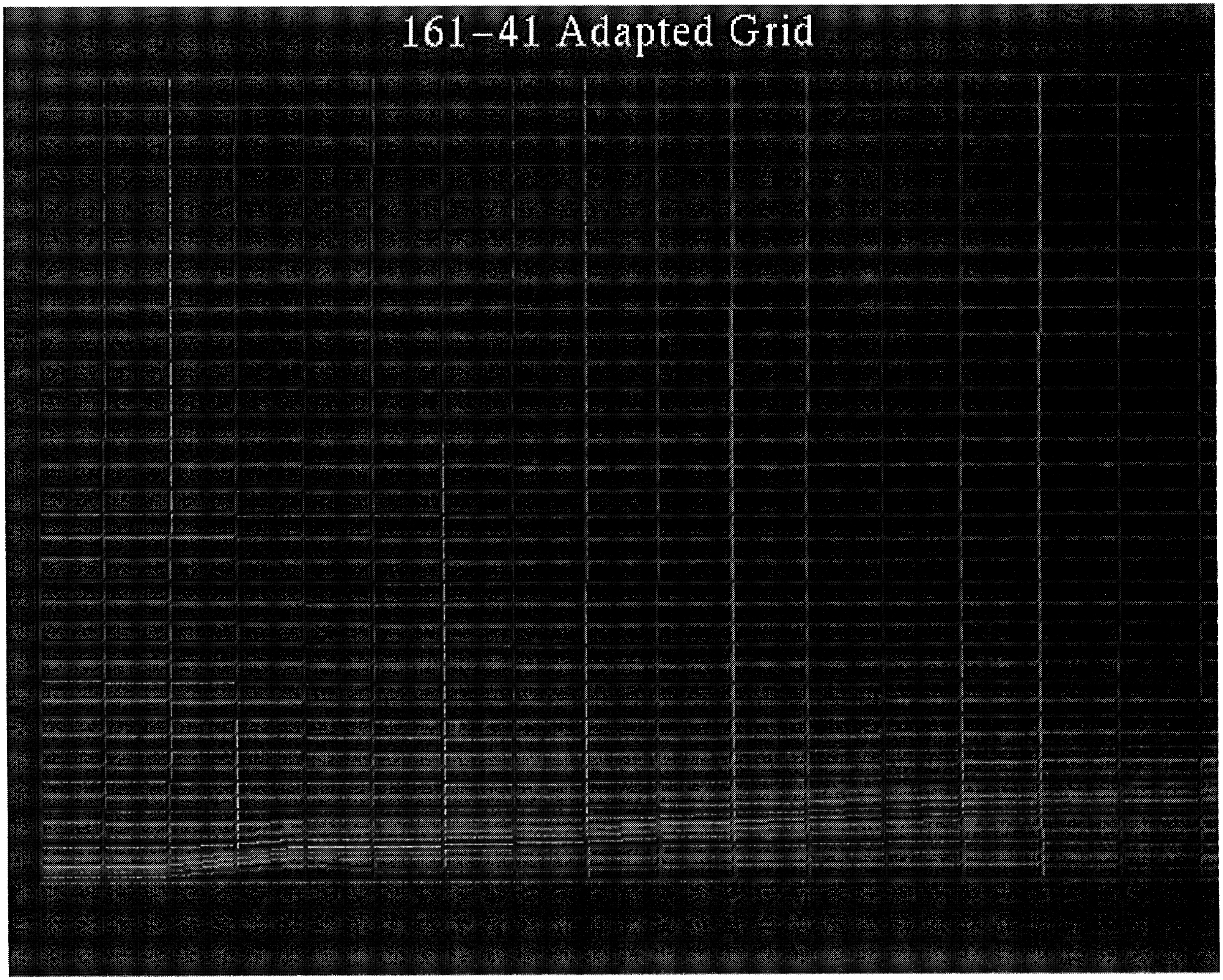


Figure 5

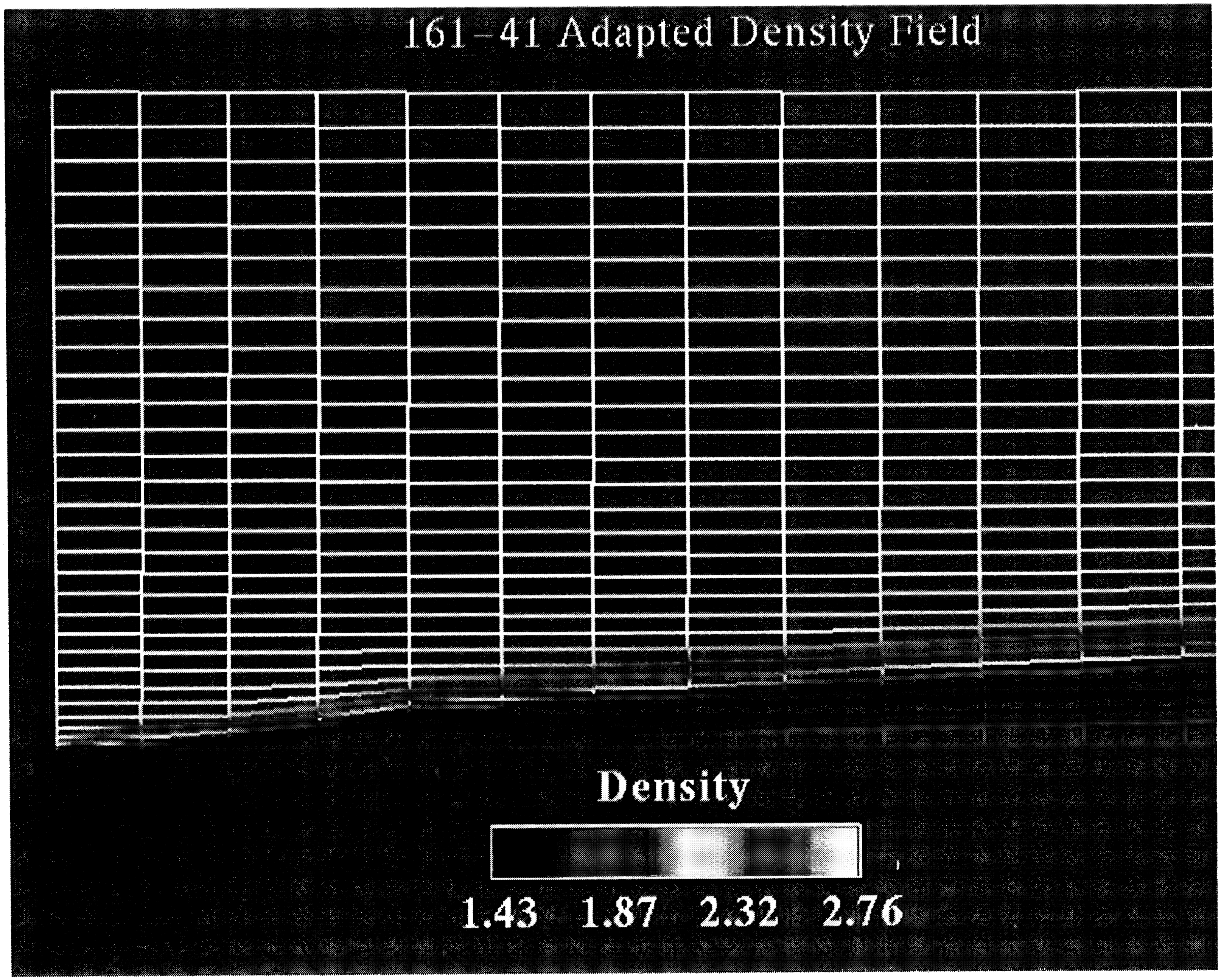


Figure 6

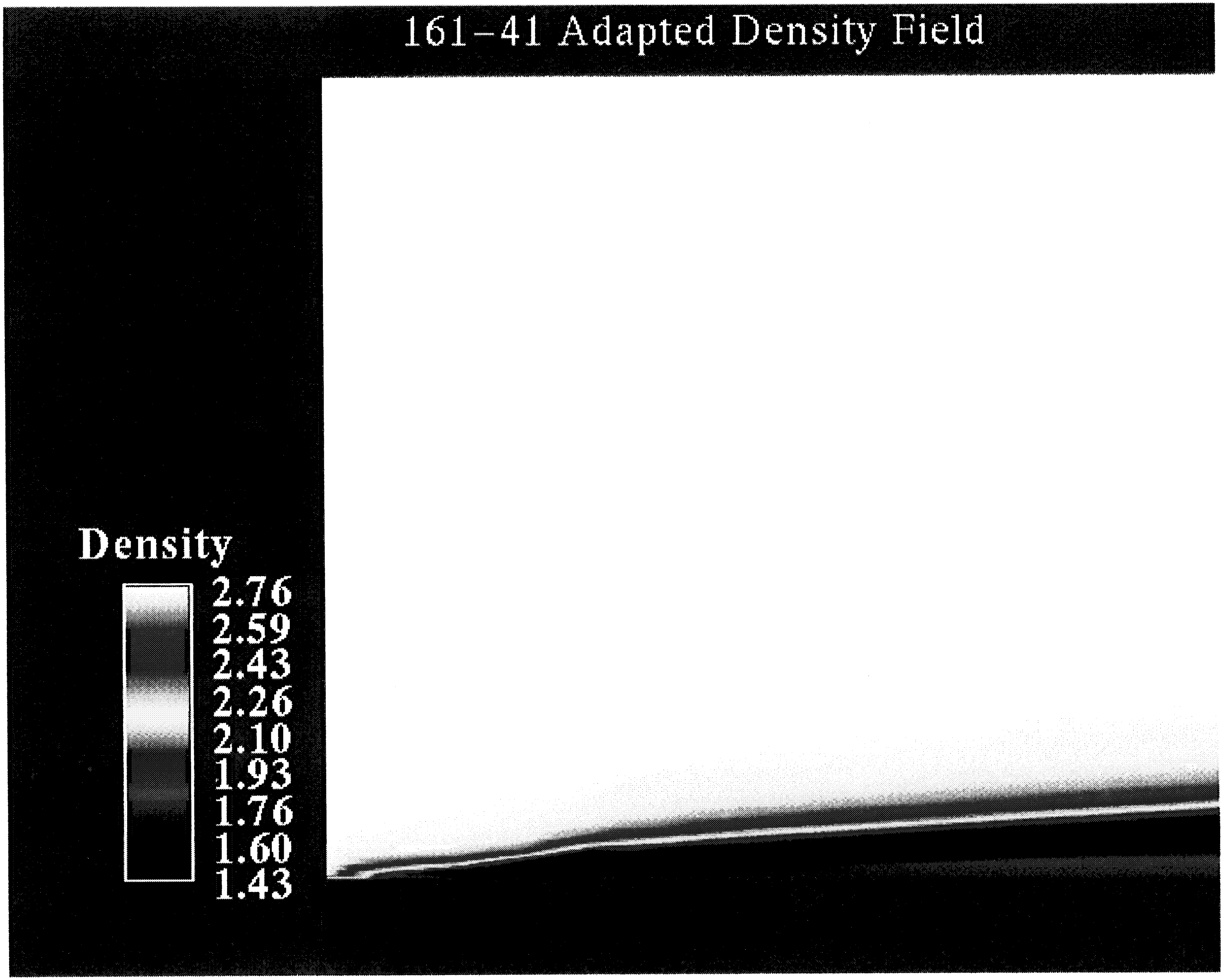

161–41 Adapted Density Field
Density
2.76
2.59
2.43
2.26
2.10
1.93
1.76
1.60
1.43

Figure 7

VI. Conclusions

An adaptive version of FDNS has been developed which allows the user to simulate flow fields in arbitrary geometries without code modifications. Adaptive grid capability was provided in the code by coupling it with the adaptive version of the EAGLE grid generation system. Periodic, multiple adaptations of the block–structured grids during computations demonstrate that solution–adaptive coarse grids give results as good as those obtained on finer grids, but at a reduced computational cost.

However, there are some cases in which adaptation may not be useful, since the labor required to determine the appropriate adaptation parameters will negate any benefit. This is probably always true if a single simulation is to be done. But in cases where multiple runs are to be made with similar flow conditions, the adaptation parameters will remain unchanged, so that the use of coarse or moderately fine grids and periodic adaptation will produce results equivalent or superior to those obtained on a grid fine enough to resolve the field without adaptation.

The adaptive results obtained in the present study show improved solution accuracy in the resolution of boundary and shear layers, as well as the interfaces between the mixing fluids of the cold flow case. The global convergence rate is not significantly impaired by the adaptation in spite of clear residual spikes caused by not interpolating the solution from the old grid to the new, adapted grid. The savings in CPU time which can be achieved by an adaptive grid capability using fewer grid points are another benefit expected in many flow field applications, but a benefit/cost decision must be made for each case.

References

1. Thompson. J. F., "A Composite Grid Generation Code for General 3–D Regions – the EAGLE Code," *AIAA Journal*, Vol. 26, No. 3, March 1988, p. 271.
2. Lijewski, L. E., Cipolla, J., et al., "Program EAGLE User's Manual, Volume I – Introduction and Grid Applications," AFATL–TR–88–117, September 1988.
3. Thompson, J. F. and Gatlin, B., "Program EAGLE User's Manual, Volume II – Surface Generation Code," AFATL–TR–88–117, September 1988.
4. Thompson, J. F. and Gatlin, B., "Program EAGLE User's Manual, Volume III – Grid Generation Code," AFATL–TR–88–117, September 1988.
5. .Chen, Y.–S., "User's Guide for FDNS general puprose CFD Code,", Seca, Inc., Huntsville, AL, May 1993.
6. Kwak, D., Chang, J. L. C., Shanks, S. P., and Chakravarthy, S., "A Three–Dimensional Incompressible Navier–Stokes Flow Solver Using Primitive Variables," *AIAA Journal*, Vol 24, No 3, March 1986, pp. 390–396.
7. Tu, J. and Thompson, J. F., "Three–Dimensional Solution–Adaptive Grid Generation on Composite Configurations," AIAA Paper 90–0329, AIAA 28th Aerospace Sciences Meeting, Reno, Nevada, January 1990.
8. Kim, H.–J. and Thompson, J. F., "Three–Dimensional Adaptive Grid Generation on a Composite–Block Grid," *AIAA Journal*, Vol 28, No 3, March 1990, pp. 470–477.
9. Warsi, Z. U. A., "Basic DIfferential Models for Coordinate Generation," Numerical Grid Generation, edited by J. F. Thompson, North–Holland, New York, 1982, pp. 41–77.
10. Saltzman, J. S., and Brackbill, J. U., "Applications and Generalizations of Variational Methods for Generating Adaptive Mesh," *Numerical Grid Generation*, edited by J. F. Thompson, North–Holland, New York, 1982, pp. 865–884.
11. Roache, P. J., and Steinberg, S., "A New Approach to Grid Generation Using a Variational Formulation,' AIAA Paper 85–12, June 1985.
12. Warsi, Z. U. A., and Thompson, J. F., "Application of Variational Methods in Fixed and Adaptive Grid Generation," *Computers and Mathematical Applications*, Vol. 19, No. 8/9, 1990, pp. 31–41.
13. Soni, B. K., and Mastin, C. W., "Variational Methods for Grid Optimization," to appear in *Journal of Applied Mathematics and Computers*.
14. Anderson, D. A., "Generating Adaptive Grids with a Conventional Grid Scheme," AIAA Paper 86–0427, AIAA 24th Aerospace Sciences Meeting, Reno, Nevada, January 1986.

NAME

VFILE – reads in the flow solution variables file, if specified.

SYNOPSIS

$'VFILE', FILNAM=\_\_\_\_\_,FORM=\_\_\_\_\_$

DESCRIPTION

The function of this command is to read in the file containing flow variables from an existing solution to perform static adaptation on and existing grid.

PARAMETER

FILNAM=the name of the solution file.

FORM='LIST', which indicates the file is formatted and that it has the same format as the *Q* file of PLOT3D.

Bugs:

To avoid errors in this case, this command line should appear right before the CUT command, or otherwise before the command END of input.

Appendix B: Adaptive EAGLE NAMELIST Names

PARAMETER

ITMAXA = The number of adaptive iterations.

PARAMETER

ADAPT = 'NO' or 'NONE' are default values: elliptic grid is produced.

ADAPT = 'YES' adaptive grid is produced.

PARAMETER

AWT = 'VAR', 'VAR', 'VAR'; adaptation to the variable with the weight function as

$$W = 1 + | V |,$$

where V is either a flow variable or a quality measure variable.

AWT = 'GRAD', 'GRAD', 'GRAD'; adaptation to the gradient of the variable with the weight function as

$$W = 1 + | \nabla V |,$$

where V is either a flow variable or a quality measure variable.

AWT = 'CURV', 'CURV', 'CURV'; adaptation to the curvature of the variable with the weight function as

$$W = (1 + \beta | K |) \sqrt{1 + \alpha | \nabla V |^2} \ ,$$

where

$$K = \frac{\nabla^2 V}{(1 + | \nabla V |^2)^{3/2}} \ ,$$

where V is either a flow variable or a quality measure variable.

Notes:

One may specify AWT = 'VAR', 'GRAD', 'CURV', which means the grid is adapted to the variable in the 1 direction, to the gradient in the 2 direction and to the curvature in the 3 direction. Any combination of this is also valid.

PARAMETER

CW = 1.0, 1.0, 1.0, are default values for the weight coefficients.

PARAMETER

ALPHA: Coefficient of the gradient, in the range from 0 to 1, default to 1.0.

PARAMETER

BETA: Curvature coefficient, in the range from 0 to 1, default to 1.0.

PARAMETER

ASKEW: Default value is 0.0, represents skew angle.

A value of 1.0 indicates the skew angle variable of the grid is used in the calculation of the weight function during the adaptive process.

PARAMETER

AASPE: Default value is 0.0, represents aspect ratio.

A value of 1.0 indicates the aspect ratio variable of the grid is used in the calculation of the weight function during the adaptive process.

PARAMETER

AARCL: Default value is 0.0, represents arc length.

A value of 1.0 indicates the arc length variable of the grid is used in the calculation of the weight function during the adaptive process.

PARAMETER

APLAC: Default value is 0.0, represents Laplacian.

A value of 1.0 indicates the Laplacian variable of the grid is used in the calculation of the weight function during the adaptive process.

Notes:

If all of these NAMELISTs are not 0.0, then the weight function is computed as a weighted average of the individual weight functions.

PARAMETER

RHO: Default value is 0.0, 0.0, 0.0, represents the density.

A value of 1.0 indicates the density variable of the solution is used in the calculation of the weight function during the adaptive process.

PARAMETER

RHOU: Default value is 0.0, represents the x–momentum.

A value of 1.0 indicates the x-momentum variable of the solution is used in the calculation of the weight function during the adaptive process.

PARAMETER

RHOV: Default value is 0.0, represents the y–momentum.

A value of 1.0 indicates the y-momentum variable of the solution is used in the calculation of the weight function during the adaptive process.

PARAMETER

RHOW: Default value is 0.0, represents the z–momentum.

A value of 1.0 indicates the z–momentum variable of the solution is used in the calculation of the weight function during the adaptive process.

PARAMETER

RHOE: Default value is 0.0, 0.0, 0.0, represents the energy.

A value of 1.0 indicates the energy variable of the solution is used in the calculation of the weight function during the adaptive process.

PARAMETER

VOMA: Default value is 0.0, 0.0, 0.0, represents the velocity magnitude.

A value of 1.0 indicates the velocity magnitude of the solution is used in the calculation of the weight function during the adaptive process.

PARAMETER

VORR: Default value is 0.0, 0.0, 0.0, represents the vorticity magnitude.

A value of 1.0 indicates the vorticity magnitude of the solution is used in the calculation of the weight function during the adaptive process.

PARAMETER

VARIN: Defaulted value is 'NO', VARIN = 'YES' indicates the restart file 'rsfile' contains adaptive variables array from the previous run.

PARAMETER

VAROUT: Defaulted value is 'NO', VAROUT = 'YES' indicates that in the current run adaptive variables array will be saved on the restart file 'rsfile'.

PARAMETER

RESTART: Defaulted value is 'NO', RESTART = 'YES' means a restart file namely 'rsfile', will be generated at the end of the current run.

PARAMETER

AFIXP: Defaulted value is 'YES', AFIXP = 'NO' means the control function is updated at every adaptive iteration.

PARAMETER

INTCYL: Defaulted value is 0, represents the number of time step at which the adaptation is performed.

PARAMETER

NUMCYL: Defaulted value is 999, represents the interval of time step between adaptations.

PARAMETER

MAXINT: Defaulted value is 9999, this parameter indicates the number of time step at which the last adaptation is performed.

PARAMETER

QUALITY: Defaulted value is 'NO'. QUALITY = 'MEASURE' means the grid is being adapted to the quality measure variables alone.

Appendix C: Example of Static User File

```
$'INITIAL',ITMAXA=1,ITMAX=5,CONTYP='INITIAL',ALL='YES',
 KSTORE='FILE',PROTYP='NONE',
 CONFAC=.3,TOL=0.,
 CHECK='NO',DFIRST='ONESIDE',INTERP='NO',
 RESTART='NO', ACCPAR='OPTIMUM',
 AWT='VAR','GRAD','VAR', CW=3.5,0.6,0.0,
 AFIXP='YES',ADAPT='YES',
 RHO=1.0,1.0,0.0$
C
C
$'BLOCK',SIZE=161,41,1$
C
$'FILE',FILE=11,FORM='LIST',FILNAM='plot.x',START=1,1,1,END=161,41,1$
C
$'FIX', START=1,1,1   , END=1,1,1$
$'FIX', START=161,1,1 , END=161,1,1$
$'FIX', START=1,41,1  , END=1,41,1$
$'FIX', START=161,41,1, END=161,41,1$
C
$'FIX', START=1,41,1    , END=161,41,1$
$'FIX', START=161,1,1   , END=161,41,1$
$'NEUMANN', START=1,1,1 , END=161,1,1$
$'NEUMANN', START=1,1,1 , END=1,41,1$
C
$'VFILE',FILNAM='plot.q',FORM='PLOT3D'$
C
$'END'$
$'ERROR'$
$'END'$
```

Appendix D: Incorporation Into FDNS

The coupling procedure is done in the main flow code by simply adding two sections of the adaptive grid code: One is added at the declaration section of the flow code and the other inside the time step loop of the flow code.

Addition of the first adaptive section **(#ifdef adapt-def)**, which is added at the declaration of the main flow code, involves several items, as follows:

20. The specification and declaration of adaptive grid variables.

    Character: AWT, RESIN, QUALITY, AFIXP

    Dimension:FACTOR(9,3),CW(3),AWT(3),RHO(3),RHOE(3),PRES(3),VOMA(3), VORR(3), ADAPTVAR(3, IIQMAX),TEMPS(3),Q(10=NSPM,IIQMAX)

    Here the array ADAPTVAR is added to store the adaptive variable and is peculiar to the flow code.

    Data: RESIN='YES', AWT = VAR', 'VAR', 'VAR', AFIXP = 'YES', QUALITY = 'NO', CW= 1,1,1, RHO = 0,0,0, RHOU = 0, RHOV = 0, RHOW = 0, RHOE = 0,0,0, PRES = 0,0,0, VOMA = 0,0,0, VORR = 0,0,0, ITMAX = 1, ITMAXA = 1, INTCYL= 0, NUMCYL= 1, KFILE = 99, ALPHAS = 0, BETAA = 0, ASKEW= 0, AARCL= 0, AASPE= 0, APLAC = 0, ISTAT = 0, MAXINT = 9999,ADAPT='YES',TEMPS=0,0,0.

21. The creation of the NAMELIST of adaptive variables.

    Namelist: RESIN, AWT, CW, RHO, RHOU, RHOV, RHOW, RHOE, PRES, VOMA, VORR, ITMAX, INTCYL, NUMCYL, ALPHA, BETA, ITMAXA, QUALITY, ASKEW, AARCL, AASPE, APLAC, AFIXP, MAXINT.

22. The opening of all the named scratch and input files used in the process.

23. The NAMELIST read statement to read in these variables from input file.

24. Store values of RHO, RHOU, RHOV, RHOW, RHOE, PRES, VOMA, VORR into array FACTOR. It should be verfied that the flow code does not contain variables with the same names.

The second adaptive section **(#ifdef adapt-io)**, which is added inside the time-step loop of the main flow code, involves several steps as follows:

1. Set up two IF statements inside the time step loop of the main flow code, one to check if current step is between the allowed start and end of adaption and the other to check if adaption is desired at the current iteration.

    ```
    IF (( ITO .GE. INTCYL) .AND. (ITO .LE. MAXINT) )THEN
    IF (MOD(ITO+NUMCLY,NUMCYL) >EQ. 0) THEN
    ```

SSDRA & SSDWA
When the NAMELIST KSTORE = 'FILE' and ADAPT = 'YES', these subroutines are called by SSD to store data not currently needed in files for later retrieval.

REDRES This subroutine is called by ELLGEN to read in the restart file rsfile.

SETIMP This subroutine sets control function values on boundaries. It is called by ELLGEN.

SETIMR This subroutine sets coordinate values at Neumann, image, and reflective points. It is called by ELLGEN.

SETIMV This subroutine sets the adaptive variable values at image points equal to the current values at the corresponding object points, sets values at Neumann and reflective boundaries and special points, and also extrapolates to other boundary points. It is called by ELLGEN.

SETIMW This subroutine has the same functions SETIMV, but with the weight values instead. Called by ELLGEN.

SMOOTHW
After the weight values are set on the field and boundaries, this subroutine smooths these weight values. It is called by ELLGEN.

STOREP When KSTORE = 'FILE' & ADAPT = 'YES', it is called by ELLGEN to store the original control function values in array for later use.

STOREPW This subroutine has the same function as STOREP for KSTORE = 'CORE' and ADAPT = 'YES' and is called by ELLGEN.

WEIT2D This subroutine is called by ELLGEN for computing the weight function values inside a 2–D field.

WEIT3D This subroutine is called by ELLGEN for computing the weight function values inside a 3 –D field.

WEITCON This subroutine is called by ELLGEN to perform the linear combination of the original control function values that have been saved by STOREP or STOREPW and the weight function values.

WAGRID This subroutine writes out the new grid into scratch file KFILE and it is called by ELLGEN.

Appendix E: Example of Dynamic User File

```
C
C                              START OF ADAPTIVE INPUT DECK
C $ADAPTS  RESIN='YES',AFIXP='NO',
      AWT='GRAD','GRAD','VAR',
      CW=0.8,0.8,0.0,ALPHAS=1.0,BETAA=1.0,
      RHOE=0,0,0,
      PRES=1,1,0,
      RHO= 0,0,0,
      RHOU=0,RHOV=0,RHOW=0,
      VOMA=0,0,0,VORR=0,0,0,
      TEMPS=0,0,0,
      ITMAXA=1,
      INTCYL=100,NUMCYL=50,MAXINT=1000,
      QUALITY='NO',
      ASKEW=0.0,AASPE=0.0,AARCL=0.0,APLAC=0.0  $
C
C              END OF THE ADAPTIVE INPUT DECK
C
```

Appendix F: Adaptive FDNS User's Guide

As computational fluid dynamics (CFD) methods are extended to simulate increasingly complex flowfields or geometries, solution accuracy and efficiency in obtaining these solutions have become paramount. It is well understood that increasing the number of grid points to obtain more accurate solutions results in increased computation. For many practical problems, however, consideration should be taken to provide adequate resolution in the flow field, for example, near shocks, boundary and shear layers, and wake regions. Adaptive grid generation as a means to redistribute grid points based on the nature of the flow is an answer to many such problems. Adaptive grids move in response to the evolving solution of the flow equations and concentrate in regions of large solution gradients without *a priori* knowledge of the solution. This yields an ability to better resolve the flow field. As a consequence, adaptive grid generation allows one to increase the accuracy of solutions and efficiency of the flow code. In principle, adaptive grids provide a more accurate simulation of the flow at a reduced computational cost, since fine grid resolution is not wasted in regions where it is not needed.

A principal idea of adaptive grid generation comes from the fact that the equidistribution of error over the field contributes toward the redistribution of the grid points so that the product of the spacing and some positive weight function is held constant. Thus if the weight function is the gradient of the flow solution, the grid points would be closely spaced in regions of large gradient, and widely spaced where the solution is smooth.

The adaptive EAGLE grid system was developed based upon the control function approach. Forcing functions in this system contain weights which cause grid points to migrate to regions of large solution gradients or some measure of error. Adaptive EAGLE grid generator provides for different weight functions in each coordinate direction. The gradient of a variable (such as grid quality measure variable or flow solution variable), the curvature of a variable, or the variable itself is considered as the source of different weight functions. Thus the weight functions can be expressed as follows.

$$\begin{aligned} W &= 1 + | V | && \textit{Variable} \\ W &= 1 + | \nabla V | && \textit{Gradient} \\ W &= (1 + \beta | K |)\sqrt{1 + \alpha | \nabla V |^2} && \textit{Curvature} \end{aligned} \tag{1}$$

where α and β are parameters to be specified, and V can be either a grid quality measure variable or a flow solution variable. The curvature of the variable curve is defined as

$$K = \frac{\nabla^2 V}{(1 + | \nabla V |^2)^{3/2}} \ . \tag{2}$$

The weight function given by the curvature of the variable curve in Eq. (1) provides the tendency toward point concentration both in regions of large gradient and high curvature of the variable curve, *e.g.*, near extrema. Clearly, concentration near large gradients can be achieved with large values of α, while the grid points will be concentrated near extrema by large β.

In addition to the weight functions mentioned, the system includes the ability to calculate the weight functions as weighted averages of weight functions from several grid quality measure variables and/or flow solution variables. Here, the grid quality measures available in the adaptive EAGLE grid system are skewness, aspect ratio, arc length, and smoothness of the grid. Provision is also made for the control function employed in this system so as to be based upon both the geometrical characteristics of the initial grid and the nature of the evolving flow solution, *i.e.*,

$$P_i = (P_i)_g + C_i(P_i)_w \qquad i = 1,2,3 \tag{3}$$

where $(P_i)_g$ is the control function based upon the geometry involved, $(P_i)_w$ is the control function based upon the weight function defined by Eq. (1), and C_i is the weight coefficient to be specified.

The adaptive EAGLE grid system has been used to generate static and multiple adaptive grids in simulating flows. The static adaptive grid is generated by adapting the initial grid only once in response to a weight function based on either grid quality measure variables or flow solution variables obtained by the flow code. Then, the newly generated grid by the adaptive EAGLE grid generator is supplied to the flow code for restart as a better grid with improved grid quality measures or providing adequate resolution in the flow field, depending upon the choice of adaptation. In general, static adaptation to flow solution variables provides helpful ideas regarding the appropriate source of weight function and input parameters to be determined before multiple adaptive grid generation procedure.

For multiple adaptation of grids, the adaptive EAGLE grid system has been incorporated into FDNS. The adaptive grid system consists of a set of #ifdef constructs- which are easily added to flow code that operates on a block–structured grid (or single–block grid). The flow code then calls the subroutine ellgen at each time step when a new grid is desired, passing the current flow solution via a scratch files and receiving the new adapted grid via the scratch files.

By adapting at multiple time steps during the flow evolution, grid points are moved gradually to the final grid of high local density to capture accurately the flow features.. The adaptive grid for the flows with strong gradients often has some local regions of increased grid point density and less dense grid distribution with no change of total grid points. This grid system, at present, has no interpolation procedures for the solutions used to take the movement of grid points into account. This interpolation procedure will be added in order to maintian the transient accuracy of the FDNS code.

Program Execution

The user of adaptive FDNS is assumed to be familiar with both the basic EAGLE grid generation code and FDNS.

1. Input Files

There are two extra input files which must always be prepared at the start of the run for dynamic adaption.

1. Adaptive restart file "rsfile", which is created by the adaptive EAGLE grid code, contains adaptive variables array. The rsfile at present must be generated in a preproessing step. This step will be incorporated into the code for the final version and will be transparent to the user.
2. Parameter NAMELIST file "adapt–input" contains the input parameters necessary to the execution of the adaptive FDNS code. This file consists of the NAMELIST: ADAPT for adaptive grid generator.

2. Output Files

Since the contact between the EAGLE adaptive code and FDNS code is done through the scratch files, the output of the adaptive grid is transparent to the user. The adaptive code writes and reads the scratch files and reloads the adaptive grid into the FDNS arrays. Run history of the adaptive code is displayed on the screen and also can be saved using redirection of output as follows:

```
FDNS.JCL > "run history filename" &.
```

The run history file shows the listing of input parameters specified, convergence history, and all the information associated with adaptive grid generation as well as the normal screen output of the FDNS code.

Note: After the adaptive grid is generated, grid points on the inflow boundary are also moved unless fixed or orthogonal boundary conditions are used. Therefore, if program changes are done in order to specify inlet velocity profil

Input Parameter NAMELIST (ADAPT)

❑ User–specified parameter list in the file "adapt–input"

Variable	Range	Default	Description
RESIN	YES	YES	RESIN="YES" allows the adaptive, multi-block INS3D to read in the restart file "rsfile" resulted from adaptive EAGLE grid code.
AFIXP	YES or NO	YES	AFIXP="NO" means the control function is updated at every adaptive iteration, giving stronger adaptation.
AWT	VAR, GRAD, or CURV	VAR	The grid is adapted to variable (VAR), gradient of variable (GRAD), or curvature of variable (CURV).
CW	any	1.0	This is the weight coefficient defined in Eq. (3).
ALPHA	0.0 to 1.0	1.0	This is the coefficient of gradient defined in Eq. (1).
BETAA	0.0 to 1.0	1.0	This is the coefficient of curvature defined in Eq. (1).
RHOE	0 or 1	0	RHOE=1 indicates the energy variable of the solution is used in the calculation of the weight function during the adaptive process.
RHO	0 or 1	0	RHO=1 indicates the density variable of the solution is used in the calculation of the weight function during the adaptive process.
RHOU	0 or 1	0	RHOU=1 indicates the x–component momentum variable of the solution is used in the calculation of the weight function during the adaptive process.

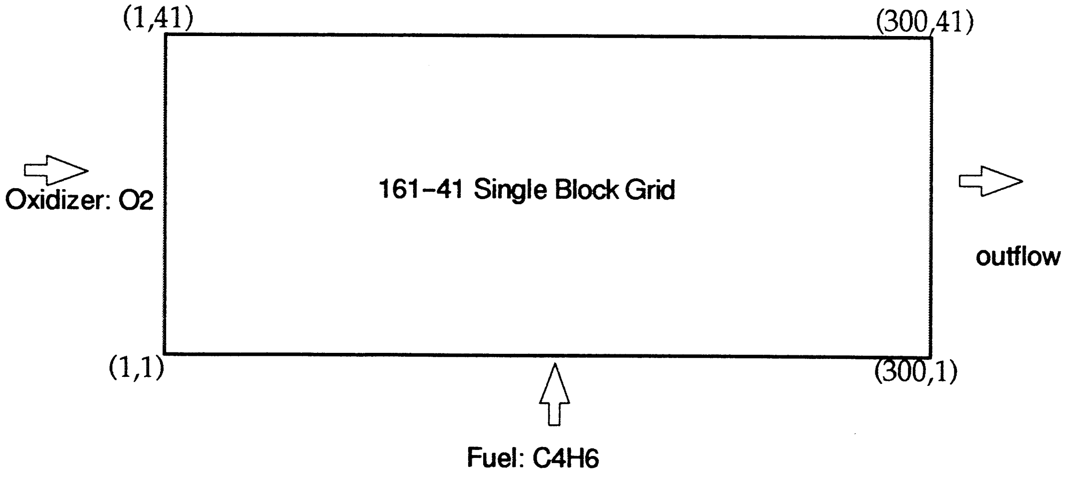
AASPE	0 or 1	0	AASPE=1 indicates the aspect ratio variable of the grid is used in the calculation of the weight function during the adaptive process.
AARCL	0 or 1	0	AARCL=1 indicates the arc length variable of the grid is used in the calculation of the weight function during the adaptive process.
APLAC	0 or 1	0	APLAC=1 indicates the Laplacian variable of the grid is used in the calculation of the weight function during the adaptive process.

Example

1. 2D Flat plate with Blown Combustion

□ Parameter NAMELIST file “adapt-input”

```
C
C          START OF ADAPTIVE INPUT DECK
C $ADAPTS   RESIN='YES',AFIXP='NO',
         AWT='GRAD','GRAD','VAR',
         CW=0.8,0.8,0.0,ALPHAS=1.0,BETAA=1.0,
         RHOE=0,0,0,
         PRES=1,1,0,
         RHO= 0,0,0,
         RHOU=0,RHOV=0,RHOW=0,
         VOMA=0,0,0,VORR=0,0,0,
         TEMPS=0,0,0,
         ITMAXA=1,
         INTCYL=100,NUMCYL=50,MAXINT=1000,
         QUALITY='NO',
         ASKEW=0.0,AASPE=0.0,AARCL=0.0,APLAC=0.0  $
C
C        END OF THE ADAPTIVE INPUT DECK
C
```



Appendix G: Example of #ifdef Constructs

```
cnvx*DECK FDNS
   PROGRAM FDNS
C*******************************************************************
C
C   FDNS — FINITE       (VERSION 3.0, Date: 05–1993)
C         DIFFERENCE
C         NAVIER–
C         STOKES FLOW SOLVER IN
C         3–D OR 2–D SPACE (INCOMPRESSIBLE/COMPRESSIBLE;
C        LAMINAR/TURBULENT AND STEADY/UNSTEADY FLOW PROBLEMS)
C
C**** MULTI–ZONE, CHEMISTRY, PARTICLE & POROSITY MODELS **************
C
C   BY:  Y. S. CHEN          TEL: (205) 721–0660
C        Engineering Sciences, Inc. (ESI)
C        4920 Corporate Dr., Suite K
C        Huntsville, Alabama 35805
C
C——(INPUT DATA DESCRIPTIONS INCLUDED IN THE READ.ME FILE)
C**** CHANGE IIQMAX, IWP & ISLMAX: FLOW, WALL & INTERFACE
DIMENSIONS**
C*****——— ISPMAX & NSPM FOR CHEMSTRY DIMENSIONS *******************
C*****——— IJKPMX & NPMAX FOR PARTICLE DIMENSIONS ******************
C*****——— IJKVMX & NPOROX FOR POROSITY DIMENSIONS *****************
cnvx*CALL fdns01
   include 'fdns01'
cnvx*CALL fdns02
   include 'fdns02'
cnvx*CALL fdns03
   include 'fdns03'
cnvx*CALL fdns04
   include 'fdns04'
cnvx*CALL fdns05
   include 'fdns05'
cnvx*CALL fdns06
   include 'fdns06'
cnvx*CALL fdns07
   include 'fdns07'
cnvx*CALL fdns08
   include 'fdns08'
cnvx*CALL fdns09
   include 'fdns09'
```

```
cnvx*CALL fdns10
   include 'fdns10'
cnvx*CALL fdns11
   include 'fdns11'
cnvx*CALL fdns12
   include 'fdns12'
cnvx*CALL fdns13
   include 'fdns13'
cnvx*CALL fdns14
   include 'fdns14'
cnvx*CALL fdns15
   include 'fdns15'
cnvx*CALL fdns16
   include 'fdns16'
cnvx*CALL fdns17
   include 'fdns17'
   COMMON /UNSTDY/ SPP(IIQMAX,6),SUP(IIQMAX),FFG(900),FFH(900),
  &          FFI(900),FFJ(900),FFK(900),FFL(900),IIW(900)
C
C
#ifdef ADAPT
   include 'adapt-def'
#endif
C
C
C*******************************************************************
C I/O UNITS ASSIGNMENTS:
C    IR1: PROBLEM CONTROL INPUT
C    IR2: READ IN GRID FILE
C    IR3: RESTART FLOW FILE
C    IR4: FUTURE EXPANSION
C    IW1: ——————————
C    IW2: PRINT OUT GRID FILE
C    IW3: PRINT OUT FLOW FILE
C    IW4: FUTURE EXPANSION
C
C———ASSIGN FILE UNITS——————————(YS — 04/03/91)
C
   CALL IVA4(IR1,IR2,IR3,IR4, 11, 12, 13, 14)
   CALL IVA4(IW1,IW2,IW3,IW4, 21, 22, 23, 24)
C
   IDEBUG = 0
C
C*****INITIALIZE CONVERGENCE PARAMETERS
```

```
C
   CALL RVA4(ERRU,ERRV,ERRW,ERRM,  0.0, 0.0, 0.0, 0.0)
   CALL RVA4(ERRT,ERRK,ERRE,ERRFM, 0.0, 0.0, 0.0, 0.0)
C
C*****CODE INITIALIZATION (INPUT & CONSTANTS)
C
   CALL INIT(1)
   IHTOTL = 0
C*****READ IN INITIAL FLOW FIELDS FROM RESTART FILE (FROM IR2 & IR3)
C
   IF(IDATA.LE.1) THEN
C
C———RESTART FILE
C
    CALL DATAIO(1)
C
C———GRID REDISTRIBUTION AND/OR PROBLEM MODIFICATION
C     USE I/O UNITS 50 - 89 AND STATEMENT NUMBERS 7000 - 7900
C———PUT MODIFICATIONS AFTER THIS LINE (*I fdns.85)
    INCLUDE 'fmain01'
C
   ELSE
C
C*****EXAMPLE START
C
    CALL INIT(2)
    CALL EXAMP
   ENDIF
C
C*****LIMIT THE DIMENSION PARAMETER IDIM
C
   IDIM = MAX0(2,MIN0(3,IDIM))
   IF(IDIM.EQ.3) IAX = 1
C
C*****GET BOUNDARY CONTROL PARAMETERS
C
   CALL DIRCOS(0)
C
C*****CALCULATE GRID TRANSFORMATION COEFF. & TRANSFORMED
VELOCITIES
C
   CALL TRANF
C
C*****INITIALIZE TURBULENCE, SOLVER FLAGS & THERMOD. PROPERTY
```

```
C
    CALL INIT(3)
C
C*****INITIALIZE ITERATION CONTROL PARAMETERS
C
    ITO   = 0
    ISTOP = 0
C
C*****TURBULENT VISCOSITY AND THERMODYNAMIC PROPERTY
C
    IF(INSO(10).EQ.1) CALL NEWVIS
    IF(INSO(12).EQ.1) CALL PROPTY
C
C*****INITIALIZE PARTICLE AND POROSITY MODULES
CCCCC IDPTCL = 0
    IFQPT = 100
    IF(IJKPMX.EQ.IIQMAX) CALL LPTSD(1)
    IF(IJKVMX.EQ.IIQMAX.AND.NPOROX.GT.1) CALL POROST(1)
C
C*****FOR SPECIAL INTERFACE B. C. INITIALIZATION (SEE INFACE)
C
    VRO = 0.0
CCCCC CALL INFACE(1,0,0,AP)
C
C*****INITIALIZE INLET MASS FLOW RATE
C
    CALL INIT(4)
    CALL USUBIO(1,0)
C
C———ASSIGN DEN, U, V, W AND P
    DO 4002 IJK=1,IGDMAX
     SPP(IJK,1) = DEN(IJK)
     SPP(IJK,2) =   U(IJK)
     SPP(IJK,3) =   V(IJK)
     SPP(IJK,4) =   W(IJK)
     SPP(IJK,5) =   P(IJK)
     SPP(IJK,6) =   P(IJK)
 4002 CONTINUE
    IF(IDEBUG.EQ.1) PRINT *, '*** START TIME–MARCHING ***'
C
C*****START TIME–MARCHING PROCEDURE**********************************
C============================================================
==========
  2   CONTINUE
```

```
C
C*****FOR MULTI–ZONE GRID MOVEMENT SETUP (SEE INFACE)
C
      IF(IZON.GE.2.AND.THETA.LT.0.8) THEN
       CALL INFACE(2,0,IFMV,AP)
       IF(IFMV.EQ.1) CALL INFACE(1,0,1,AP)
       CALL DIRCOS(1)
      ENDIF
C
C*****CHECK TIME STEP SIZE (CFL OR CHEMISTRY LIMITS)
C
      DTT0 = 1.0/DTT
C
C*****CALCULATE TIME & START SOLUTION PROCEDURE
C
      TIMT  = (ITO+1)/DTT0
      IF(NLIMT.EQ.0) THEN
       ISTOP = 1
       GO TO 99
      ENDIF
C
C*****UPDATE PREVIOUS TIME LEVEL VARIABLES
C
      IF(DTT.LE.0.0) NLIMT = 1
      CALL BCCOND(6,1)
C
C*****VISCOSITY FUNCTION OF TEMPERATURE
C — SUTHERLAND’S CORRELATION FOR AIR AT LOW PRESSURE
C
      IF(IG.EQ.1.AND.AMC.GT.0.0) THEN
       IF(IGEO.EQ.6) THEN
        TFREE = 95.89
        DO 68 I=1,IGDMAX
           TM(I) = AMAX1(1.0E–04,TM(I))
            TEMP = TM(I)*TFREE
          VISE(I) = VISC*0.312986*TEMP**1.5/(TEMP+198.0)
  68   CONTINUE
       ELSE
        DO 69 I=1,IGDMAX
           TM(I) = AMAX1(1.0E–04,TM(I))
          VISE(I) = VISC*TM(I)**0.667
  69   CONTINUE
       ENDIF
      ENDIF
```

```
C
C*******************************************************
C*****SOLUTION PROCEDURES START
C
C——FOR PARTICLE AND POROSITY MODULES
C
    IF(MOD(ITO+1,IFQPT).EQ.1) THEN
     IF(IJKPMX.EQ.IIQMAX.AND.IDPTCL.GT.0) CALL LPTSD(2)
     IF(IJKVMX.EQ.IIQMAX.AND.IDPORO.GT.0
   &              .AND.NPOROX.GT.1) CALL POROST(2)
    ENDIF
C
C*****TIME–MARCHING SUBITERATION (NLIMT ITERATIONS)
C
    DO 3333 III = 1,1
C
C——SOLVE MOMENTUM, ENERGY AND PRESSURE CORRECTION
EQUATIONS
C
     IF(IDEBUG.EQ.1) PRINT *, '*** START SOLVEU ***'
     IF(INSO( 1).EQ.1) CALL SOLVEU( 1,ISWU,ERRUVW,ERRT)
C
C——PRESSURE CORRECTION EQUATION
     IF(IDEBUG.EQ.1) PRINT *, '*** START SOLVEP ***'
     IF(INSO( 1).EQ.1) CALL SOLVEP( 0,ISWP,ERRM)
 3333 CONTINUE
C
C——FOR WALL HEAT–CONDUCTION EQUATION
C
     IF(IDEBUG.EQ.1) PRINT *, '*** START SOLVET ***'
     I1 = 0
     DO 3339 II=1,ID
      IF(IWALL(II).EQ.1) I1 = 1
 3339  CONTINUE
     IF(INSO( 4).EQ.1.AND.I1.EQ.1) CALL SOLVET( 4,ISWU,ERRT)
C
C——SOLVE TURBULENCE EQUATIONS
     IF(IDEBUG.EQ.1) PRINT *, '*** START SOLVEK ***'
     IF(INSO( 5).EQ.1) CALL SOLVEQ( 5,ISWK,ERRK)
     IF(INSO( 6).EQ.1) CALL SOLVEQ( 6,ISWK,ERRE)
C
C——FOR SPECIES EQUATIONS AND PRESSURE CORRECTION EQUATION
     ISOPRO = 1
```

```
      IF(INSO(11).EQ.1.AND.(NREACT.EQ.0.OR.ISOPRO.EQ.1)) THEN
       IF(IDEBUG.EQ.1) PRINT *, '*** START SOLVES ***'
       CALL SOLVES(11,ISWK,ERRFM)
      ENDIF
C———FOR THERMODYNAMICS PROPERTIES
C
      IF(IDEBUG.EQ.1) PRINT *, '*** START PROPTY ***'
      IF(INSO(12).EQ.1.AND.ISOPRO.EQ.1) CALL PROPTY
      CALL BCCOND(1,1)
C
C———CONVERGENCE CHECK
C
      ERRMAX = AMAX1(ERRM,ERRUVW,ERRT)
      IJK    = IMN+IZS(JMN)
      IJK1   = IPC+IZS(JPC)
      DMON   = DEN(IJK)
      UMON   =  U(IJK)
      VMON   =  V(IJK)
      WMON   =  W(IJK)
      PMON   =  P(IJK)-P(IJK1)
      HMON   = TM(IJK)
      YPLS   = YPLN(MAX0(1,IJLO(IJK)))
      TWNN   = TWN(MAX0(1,IJLO(IJK)))
      ITO    = ITO+1
      CALL AINDEX(IMER,IZZ,I,J,K)
      WRITE( 6,9350) ITO,IZZ,I,J,K,ERRUVW,ERRM,ERRT,ERRK,UMON
      WRITE(21,9350) ITO,IZZ,I,J,K,ERRUVW,ERRM,ERRT,ERRK,UMON
ccccc                                                   WRITE(IW1,9351)
ERRMAX,DMON,UMON,VMON,WMON,PMON,FLOTIN,FLOTEX
 9350  FORMAT(2X,I5,4I4,5(1P,E11.4))
 9351  FORMAT(8(1P,E10.2))
C
      IF(ERRMAX.LE.EREXT) ISTOP = 1
C
C———UPDATE TURBULENCE EDDY VISCOSITY
     IF(INSO(10).EQ.1) CALL NEWVIS
C
C*****END OF SOLUTION PROCEDURE FOR ONE TIME STEP*****
C
C
#ifdef ADAPT
      include 'adapt-io'
#endif
C
```

```
C
 99 CONTINUE
    IOUT = 0
    IF(ISTOP.EQ.1.OR.(ITO.GT.1.AND.MOD(ITO+ITPNT,ITPNT).EQ.0))
   & IOUT = 1
    IF(AMC.GT.0.0) CALL USUBIO(2,IOUT)
C
C*****PRINT OUT SOLUTIONS
    IF(IOUT.EQ.1) CALL DATAIO(2)
C*****USER PROFILE DATA OUTPUT CAN BE PRINTED AS SHOWN BELOW*****
C    USE I/O UNITS 50 – 89 AND STATEMENT NUMBERS 8000 – 8900
C———PUT PROFILE DATA OUTPUT AFTER THIS LINE——(*I fdns.267)
      INCLUDE 'fmain02'
C
C———FLOWRATES
      ARETIN = 0.0
      FLOTIN = 0.0
      CALL AREAIO(0,INB)
      CALL FLOWIO(0,INB)
      DO 9995 IB=1,INB
       FLOTIN = FLOTIN+FLOWX(IB)
 9995   ARETIN = ARETIN+AREAX(IB)
      CALL AREAIO(1,INB)
      CALL FLOWIO(1,INB)
      DO 9996 IB=1,INB
       FLOTIN = FLOTIN+FLOWX(IB)
 9996   ARETIN = ARETIN+AREAX(IB)
      CALL AREAIO(–1,INB)
      CALL FLOWIO(–1,INB)
      DO 9997 IB=1,INB
       FLOTIN = FLOTIN+FLOWX(IB)
 9997   ARETIN = ARETIN+AREAX(IB)
      ARETEX = 0.0
      FLOTEX = 0.0
      CALL AREAIO(2,INB)
      CALL FLOWIO(2,INB)
      DO 9998 IB=1,INB
       FLOTEX = FLOTEX+FLOWX(IB)
 9998   ARETEX = ARETEX+AREAX(IB)
C
C———CHECK INPUT & PRINT OUT GLOBAL MASS–FLOW CONDITIONS
C
    IF(MOD(ITO,10).EQ.0) THEN
C———INPUT MODIFICATION
```

```
 9600  FORMAT(1X)
       REWIND(IR1)
       ISKIP = 11+IZON+2*IZFACE+IBND+ID+ISNGL
       DO 9601 II=1,ISKIP
        READ(IR1,9600)
 9601  CONTINUE
       READ(IR1,*) I1,I2,ITT,ITPNT,ICOUP,I3,I4,I5
       READ(IR1,9600)
       READ(IR1,*) DTT,IREC,REC,THETA,BETAP,IEXX,PRAT
        IF(DTT.EQ.0.0) DTT = –1.00
       READ(IR1,9600)
       READ(IR1,*) I1,I2,I3,I4,IMN,JMN
       READ(IR1,9600)
       READ(IR1,*) P1,I1,I2,P2,P3,P4,CBH,EREXT
       READ(IR1,9600)
       READ(IR1,*) ISWU,ISWP,ISWK,ISKEW
C
C———PRINT OUT FLOWRATES
       FACT = 32.174*DNREF1*UREF1*XREF1*XREF1
       IF(IAX.EQ.2) FACT = FACT*8.0*ATAN(1.0)
       WRITE(6,9999) FLOTIN*FACT,FLOTEX*FACT
 9999  FORMAT(1X,’ FLOWIN =’,E13.6,2X,’FLOWEX =’,E13.6)
      ENDIF
C
C*****PROGRAM TERMINATION CHECK
C
      IF(ITO.GE.ITT.OR.ISTOP.EQ.1) GO TO 999
      IF(ERRMAX.GT.1.0E+10) GO TO 999
      GO TO 2
C
C*****END OF TIME–MARCHING PROCEDURE**********************************
C
 999  CONTINUE

C
C*****END OF fdns MAIN PROGRAM
C
      STOP
      END
C
C
#ifdef ADAPT
      include ’adapt–subs’
#endif
```

NASA
National Aeronautics and Space Administration

Report Documentation Page

1. Report No.

2. Government Accession No.

3. Recipient's Catalog No.

4. Title and Subtitle

HEAT TRANSFER IN ROCKET ENGINE COMBUSTION CHAMBERS AND REGENERATIVELY COOLED NOZZLES

5. Report Date

November 16, 1993

6. Performing Organization Code

7. Author(s)

8. Performing Organization Report No.

9. Performing Organization Name and Address

SECA, Inc.
3313 Bob Wallace Avenue, Suite 202
Huntsville, AL 35805

10. Work Unit No.

11. Contract or Grant No.

NAS8-38961

12. Sponsoring Agency Name and Address

George C. Marshall Space Flight Center
Marshall Space Flight Center, AL 35812

13. Type of Report and Period Covered

Final Report

14. Sponsoring Agency Code

15. Supplementary Notes

16. Abstract

An improved heat transfer analysis for liquid rocket engine design will be developed by using state-of-the-art computational fluid dynamics (CFD) methodology to accurately describe the combusting flow and boundary conditions at and within combustion chamber walls and regeneratively cooled nozzles. Because current heat transfer technology describes the governing processes separately as: inviscid flow, boundary layer flow, and structural heat transfer, a new solution procedure was developed which couples all of these processes into a unified computational tool. SECA proposes an innovative approach of using a CFD analysis which places node points in the flowfield and the structure simultaneously to obtain the required solution. Regenerative cooling was described with boundary conditions external to the wall. The detail and rigor of this high-speed, turbulent flow analysis and the unique treatment of the wall heating will provide a more accurate thermal analysis of the engine than has been previously possible. In addition, realistic fluid properties which more accurately describe the conjugate heat transfer were analyzed. Finally, the overall rocket engine heat transfer and flowfields were modeled and analyzed.

17. Key Words (Suggested by Author(s))

regenerative cooling,

rocket engine heat transfer

18. Distribution Statement

unclassified - unlimited

19. Security Classif. (of this report)

unclassified

20. Security Classif. (of this page)

unclassified

21. No. of pages

166

22. Price