(2)

AD-A206 323

TECHNICAL REPORT RD-ST-88-7

A NUMERICAL ROUTINE FOR DEFINING ROCKET EXHAUST CHARACTERISTICS AND IMPINGEMENT EFFECTS

Roswell W. Nourse
Structures Directorate
Research, Development, and Engineering Center

SEPTEMBER 1988

U.S. ARMY MISSILE COMMAND
Redstone Arsenal, Alabama 35898-5000

Approved for public release; distribution is unlimited.

DISPOSITION INSTRUCTIONS

DESTROY THIS REPORT WHEN IT IS NO LONGER NEEDED. DO NOT RETURN IT TO THE ORIGINATOR.

DISCLAIMER

THE FINDINGS IN THIS REPORT ARE NOT TO BE CONSTRUED AS AN OFFICIAL DEPARTMENT OF THE ARMY POSITION UNLESS SO DESIGNATED BY OTHER AUTHORIZED DOCUMENTS.

TRADE NAMES

USE OF TRADE NAMES OR MANUFACTURERS IN THIS REPORT DOES NOT CONSTITUTE AN OFFICIAL INDORSEMENT OR APPROVAL OF THE USE OF SUCH COMMERCIAL HARDWARE OR SOFTWARE.

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION: Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION / AVAILABILITY OF REPORT: Approved for public release; distribution is unlimited
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S): TR-RD-ST-88-7	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION: Structures Directorate, RD&E Center; 6b. OFFICE SYMBOL (If applicable): AMSMI-RD-ST	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State, and ZIP Code): Commander, U.S. Army Missile Command, Redstone Arsenal, AL 35898-5247	7b. ADDRESS (City, State, and ZIP Code)
8a. NAME OF FUNDING / SPONSORING ORGANIZATION; 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)
A Numerical Routine for Defining Rocket Exhaust Characteristics and Impingement Effects

12. PERSONAL AUTHOR(S)
Roswell W. Nourse

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Technical	FROM _____ TO June 88	September 1988	90

16. SUPPLEMENTARY NOTATION

17. COSATI CODES: FIELD	GROUP	SUB-GROUP

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
Momentum Integral Method Impingement Effects
Rocket Exhaust
Rocket Plumes

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

This report presents a computer method for predicting the thermodynamic properties of decaying rocket exhaust plumes. A momentum integral technique is the chosen method for the program. This method describes accurately the plume parameters of pitot pressure, total temperature, and velocity at discrete axial and radial locations.

Included in the report are:

1. The development of the analytical method

2. An empirical determination of the mixing parameter K

(CONTINUED ON BACK)

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION
☐ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL: Roswell W. Nourse	22b. TELEPHONE (Include Area Code): (205) 876-7384; 22c. OFFICE SYMBOL: AMSMI-RD-ST

DD Form 1473, JUN 86 Previous editions are obsolete. SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

19. Continued

3. The computer code that was developed

4. Typical results as compared to a similar method and actual measured data

Program listings are provided with a description of their use. Also included is a plot program for graphically presenting constant contour lines of pressure, temperature, and velocity within the plume flow field.

PREFACE

The author wishes to acknowledge the encouragement and support of Richard A. Reynolds, who actually conceived this project. Special thanks are also extended to Grady Patrick for writing the computer codes necessary for plotting the results and for teaching the author the programing language "C". Without the encouragement and patience of these men, the computer code developed would not have been a success.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

I. INTRODUCTION

Currently, there is a need within the Structures Directorate for a simple method to predict the thermodynamic properties of decaying rocket exhaust plumes. This is needed primarily to determine the effects of exhaust plume heating and loading on adjacent structures during missile firings. In order to accomplish this, the method must describe accurately the plume parameters of pressure, temperature, and velocity at discrete axial and radial locations.

A simple momentum integral method was chosen to accomplish the above stated goal. Therefore, the purpose of this paper is to present:

1. the analytical method chosen,

2. an empirical determination of the mixing rate parameter, K

3. the computer code that was developed,

4. and typical results as compared to a similar method and measured test data [1].

The momentum integral method used is based on the results of work done in the mid-1960's by Donaldson and Gray [2]. The method predicts the turbulent mixing and decay of axially symmetric, compressible, free jets, where the mixing gases are dissimilar. This is an extension of previous work done by Warren [3] where the mixing gases were both air. In both cases, it is assumed that there is parallel flow, no radial or axial pressure gradients, no shocks, and an optimally expanded nozzle.

II. DISCUSSION

The integral method of Donaldson and Gray [2] solves the equation

$$\frac{d}{dx}\int_0^{r^*} \rho u^2 r\,dr = u^* \frac{d}{dx}\int_0^{r^*} \rho u r\,dr + \tau^* r^* \qquad \text{(II.1)}$$

This equation states that the net rate of change in the axial direction of the momentum in a jet from the centerline out to some radius r* is equal to the change in momentum due to mass addition with velocity u* at radius r* plus the change in momentum due to the shear taking place at radius r*. Letting $r^* = r_s$, where r_s is the radius at which the local velocity is one half the value of the centerline velocity, and $\tau^* = \tau_s$, which

is the shear stress at r_s, equation (II.1) becomes

$$\frac{d}{dx}\int_0^{r_s} \rho u^2 r dr = u_s \frac{d}{dx}\int_0^{r_s} \rho u r dr - \tau_s r_s \qquad \text{(II.2)}$$

Also letting r* equal some very large radius, the two terms on the right hand side of equation (II.1) vanish. That is, u* and τ* approach zero as r* approaches infinity. This implies that the net change in total momentum passing through a plane normal to the axis is zero or that the total axial momentum does not change. This leads to the following equation

$$\int_0^{\infty} \rho u^2 r dr = \text{const} = \rho_1 u_1^2 r_1/2 \qquad \text{(II.3)}$$

A. Velocity Profiles

The method assumes a mixed viscid-inviscid plume model as shown in Figure 1. As can be seen, the jet consists of two distinct regions. The first being the core region, where there is embedded a core of flow moving at the nozzle exit velocity. The core decreases in radial extent as one moves axially down the jet to a point where it disappears. Outside the core turbulent mixing is occurring. The mixing increases in radial extent as the core decreases and continues to increase until the core disappears.

The second region, referred to as the developed region, begins once the core has disappeared. In this region, the centerline velocity decreases with axial distance and the radial extent of the plume continues to increase.

Two velocity profiles are needed to describe the radial velocity distribution. The velocity profile for the core region is given by

$$u = u_1 \times \exp[-\lambda (r^2 - r_i^2)/(r_s^2 - r_i^2)] \qquad r \geq r_i \qquad \text{(II.4)}$$

and

$$u = u_1 \qquad r \leq r_i$$

The velocity profile for the developed region is given by

$$u = u_c \times \exp[-\lambda (r/r_s)^2] \qquad \text{(II.5)}$$

In the above equations, r_i refers to the outer radius of the core, r_s is the radius at which the velocity is one half the

centerline velocity, u_1 is the exit plane velocity, u_c is the velocity along the centerline, and λ is equal to ln2.

B. Turbulent Shear Stress

As was assumed by Donaldson and Gray [2], the turbulent shear stress is given by

$$\tau = K\rho b U_S (\partial u/\partial r) \qquad \text{(II.6)}$$

where K is the mixing rate parameter, b is a local scale length, ρ is the local density, and U_S is an arbitrary local velocity difference. In the core region b is equal to $r_s - r_i$ and U_S is equal to $u_1/2$, which yields

$$\tau = K\rho (r_1 - r_i)(u_1/2)(\partial u/\partial r) \qquad \text{(II.7)}$$

For the developed region b is equal to r_s and U_S is equal to $u_c/2$, yielding

$$\tau = K\rho r_s (u_c/2)(\partial u/\partial r) \qquad \text{(II.8)}$$

C. Spreading and Decay

Equations (II.2) and (II.3) are the two basic equations describing the flow. Also, expressions have been developed for

1. u in terms of r, r_s, and r_i in the core region [Eq.(II.4)], and u in terms of r, r_s, and u_c in the developed region [Eq.(II.5)]

2. τ in terms of K, r_s, and r_i in the core region [Eq.(II.6)], and τ in terms of K, r_s, and u_c in the developed region [Eq.(II.7)]

3. and ρ in terms of u (derived in Appendix A).

Substituting these expressions into the two flow equations, only three unknowns will remain in each region. In the core region, K, r_s, and r_i are unknown and in the developed region K, r_s, and u_c are unknown.

Values of K have been determined empirically from experimental test data. A series of free-jet mixing experiments were performed by both Donaldson and Gray [2] and Warren [3], the results of which were analyzed by use of the momentum integral technique to determine local turbulent mixing rates and the local mixing rate parameter, K. The results of these tests are shown in Figure 2 [2]. This figure gives the mixing rate parameter, K, as a function of local Mach number, M_s. It was concluded from these tests [2] that a general relationship exists at any given

axial location between a local mixing rate parameter and the local Mach number. It was also shown that this parameter was independent of the physical properties of the mixing gases.

A determination of the parameter K was also made using the computer code developed and measured test data from the MK 56 rocket motor [1]. The curve developed for K from this correlation is also given in Figure 2. It differs slightly from the results of Donaldson and Gray, but the same basic trend is apparent.

Knowing K as a function of M_s, only two unknowns and two equations remain for both the core and developed regions. This, therefore, provides a means to solve for the decay and spreading behavior of the plume flowfield. The solution to these equations is given in Appendices B and C. This solution allows the determination of the velocity at any given point in the plume flowfield, which enables the parameters of total temperature, Mach number, pitot pressure, and static temperature to be calculated and plotted.

III. COMPUTER PROGRAM

The method outlined above was implemented with a "C" language computer program. The program included a slight modification to account for nozzle underexpansion, which is common in many Army missiles. A circular arc curve fit from the nozzle exit to the core boundary was used, where the core boundary is calculated assuming an ideally expanded nozzle. This curve fit was first developed for use with the momentum integral method by Reardon and Word [4]. The details of implementing this curve fit are provided in Appendix D.

The program requires an input knowledge of the

1. ambient temperature and pressure,
2. chamber temperature and pressure,
3. molecular weight of exhaust gas,
4. specific heat ratio of exhaust gas,
5. nozzle exit Mach number,
6. nozzle exit radius,
7. and nozzle half angle.

The program then calculates local flowfield parameters of Mach number, pitot pressure, and total temperature at discrete axial

and radial locations. Other parameters such as velocity, molecular weight, specific heat, and the specific heat ratio are also calculated and can be printed with slight modifications to the program.

The program is written in "C" and a listing is provided in Appendix E. A sample printout is provided in Appendix F. Also included in this report is a plot program which allows graphical presentation of the results. This plot program will plot lines of constant Mach number, pitot pressure, and total temperature within the plume flowfield. The user is prompted to choose one of these parameters for plotting. Having chosen the desired parameter, the program automatically chooses ten constant contour lines for plotting. Any number of these can be overwritten, giving the user the ability to plot any number of constant contour lines of his choosing up to ten. Figures 3, 4, and 5 are example plots for each of these parameters. A listing of the plot program is provided in Appendix G.

IV. TYPICAL RESULTS AND COMPARISONS

A similar semi-empirical method was developed in 1970 by Piesik [1]. In the presentation of his method, a comparison was made with measured data obtained by the Naval Surface Weapons Center, Dahlgren, VA, for the MK56 DTRM [5]. The Table lists the pertinent rocket motor parameters and geometry for the MK56 DTRM.

Predictions from the momentum integral method were compared to both the method of Piesik [1] and the measured data for the MK56 DTRM [1]. Comparisons were made for both the axial and radial directions of the plume flowfield.

Figures 6 and 7 give a comparison of the centerline total temperature and pitot pressure. As can be seen, the momentum integral method agrees very well with the method of Piesik and the measured data.

Radial comparisons of pitot pressure are presented in Figures 8 thru 12. Again, agreement with the predictions of Piesik and the test data is quite good radially from the centerline. Figures 11 and 12 actually indicate that the momentum integral method may actually predict pitot pressures more accurately than the method of Piesik far downstream. As mentioned previously, in order to predict exhaust plume heating or loading, an accurate model of the flowfield plume is necessary. As is indicated by this comparison, the momentum integral method is an effective procedure for modeling an exhaust plume.

V. CONCLUSIONS

From the MK56 DTRM total temperature and pitot pressure comparison, the momentum integral method showed good agreement with measured data and the predictions of Piesik's method [5]. Therefore, it is concluded that this technique is a valid design tool that can be used to predict flowfield plume parameters necessary in determining heating and pressure loadings due to exhaust gas impingement from rocket motors.

TABLE. MK56 DTRM Rocket Motor Parameters[a]

Rocket Motor	Chamber Pressure psia	Chamber Temp. °R	Specffic Heat Ratio	Molecular Weight	Exit Radius in.	Nozzle Half Angle	Exit Mach Number
MK56 DTRM	2550	6580	1.20	28.09	3.605	0.262	3.05

[a] Ambient pressure = 14.7 psia. Ambient temperature = 70 °F.

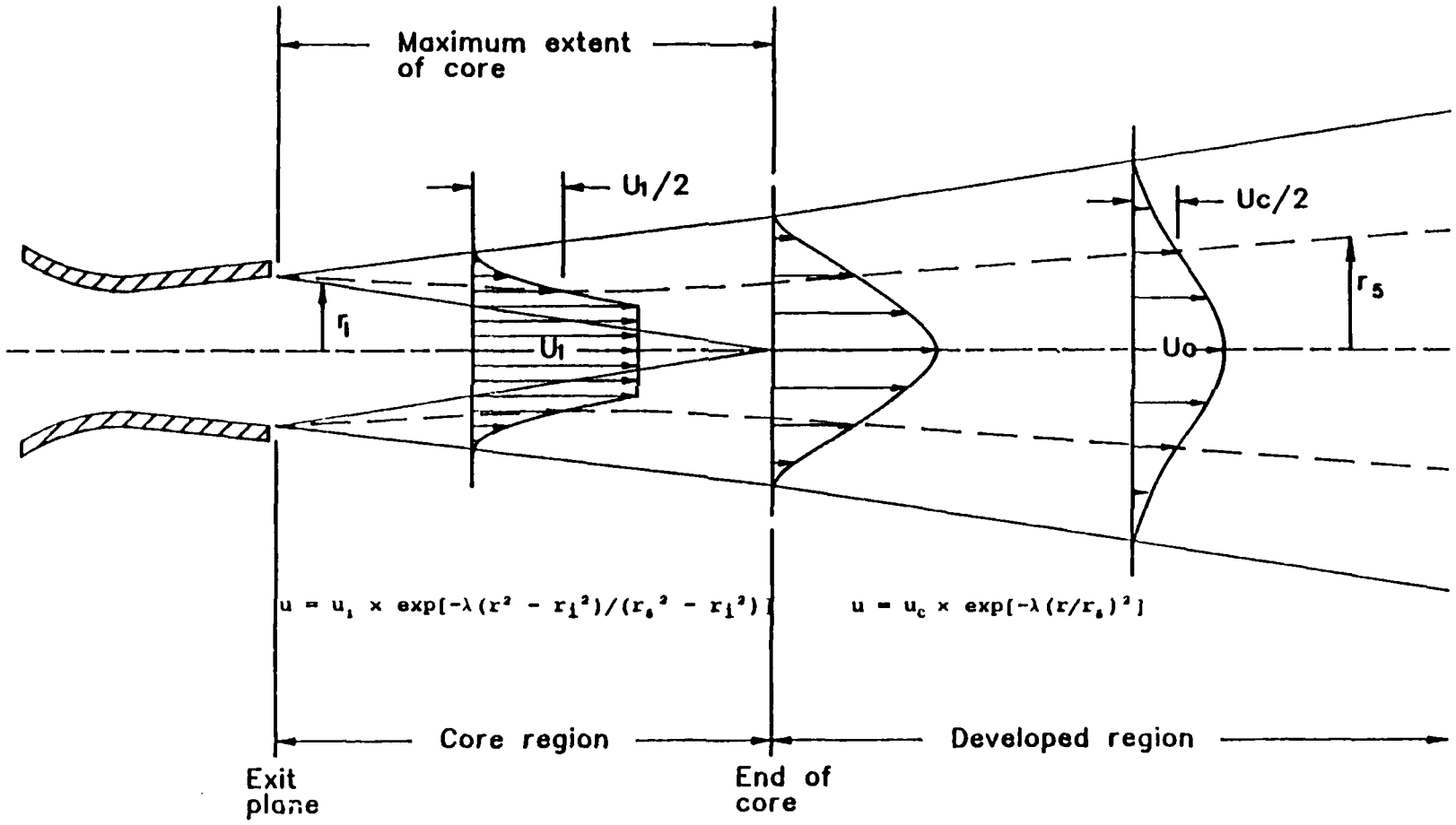

Figure 1. Fully expanded exhaust plume structure.

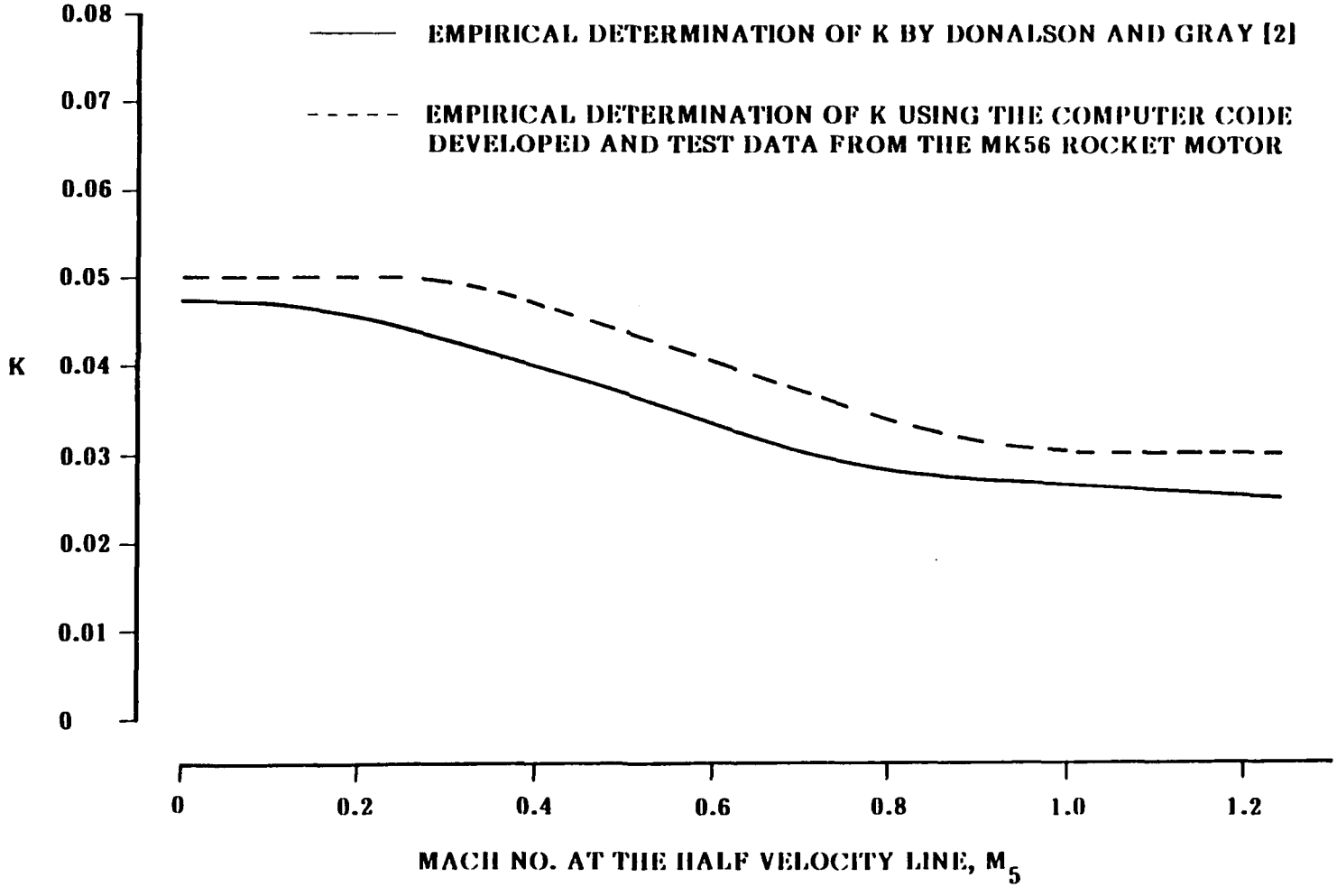

Figure 2. Empirical determinations of the mixing parameter K.

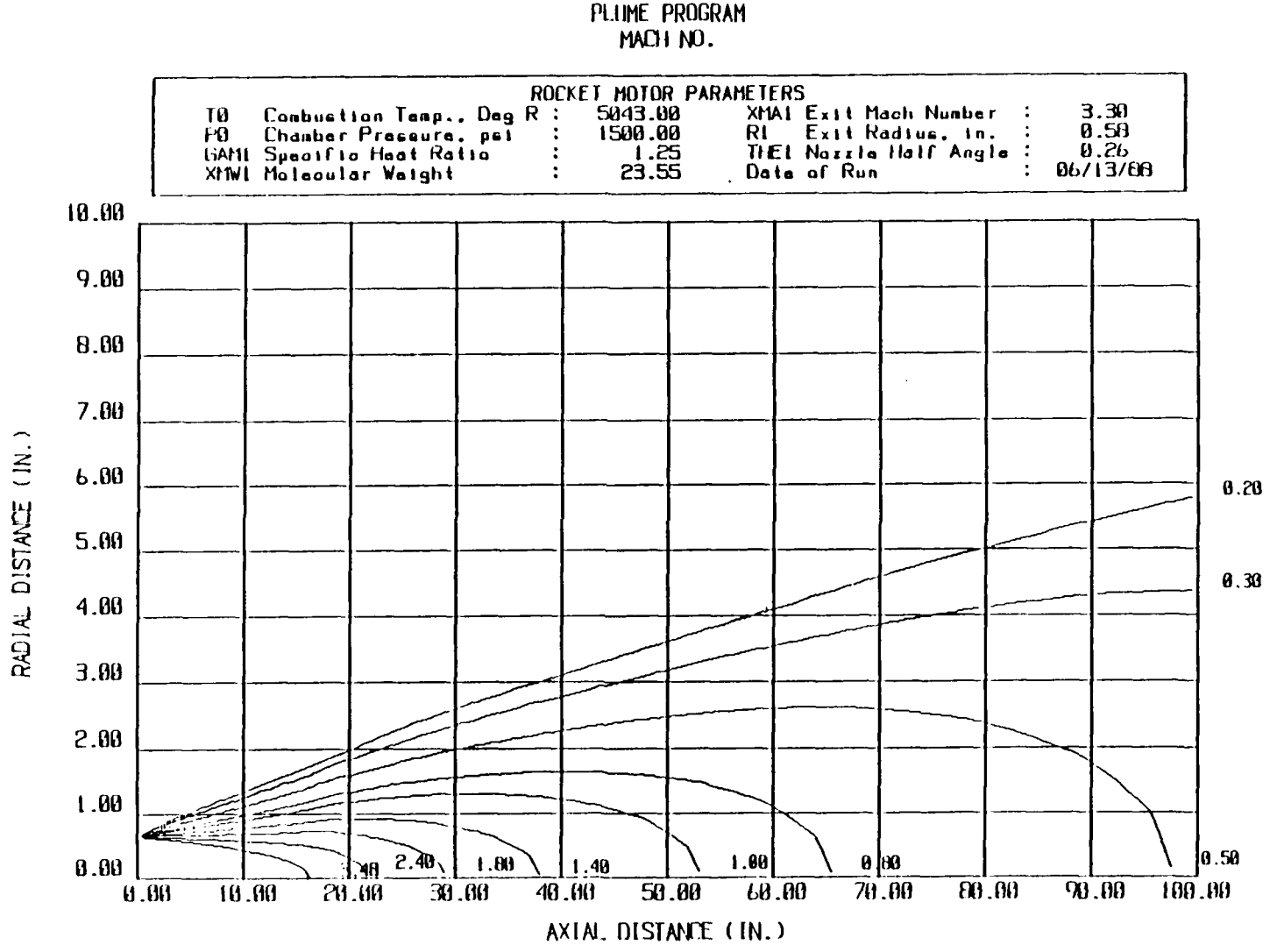

Figure 3. Typical plot of mach number distribution within plume flowfield.

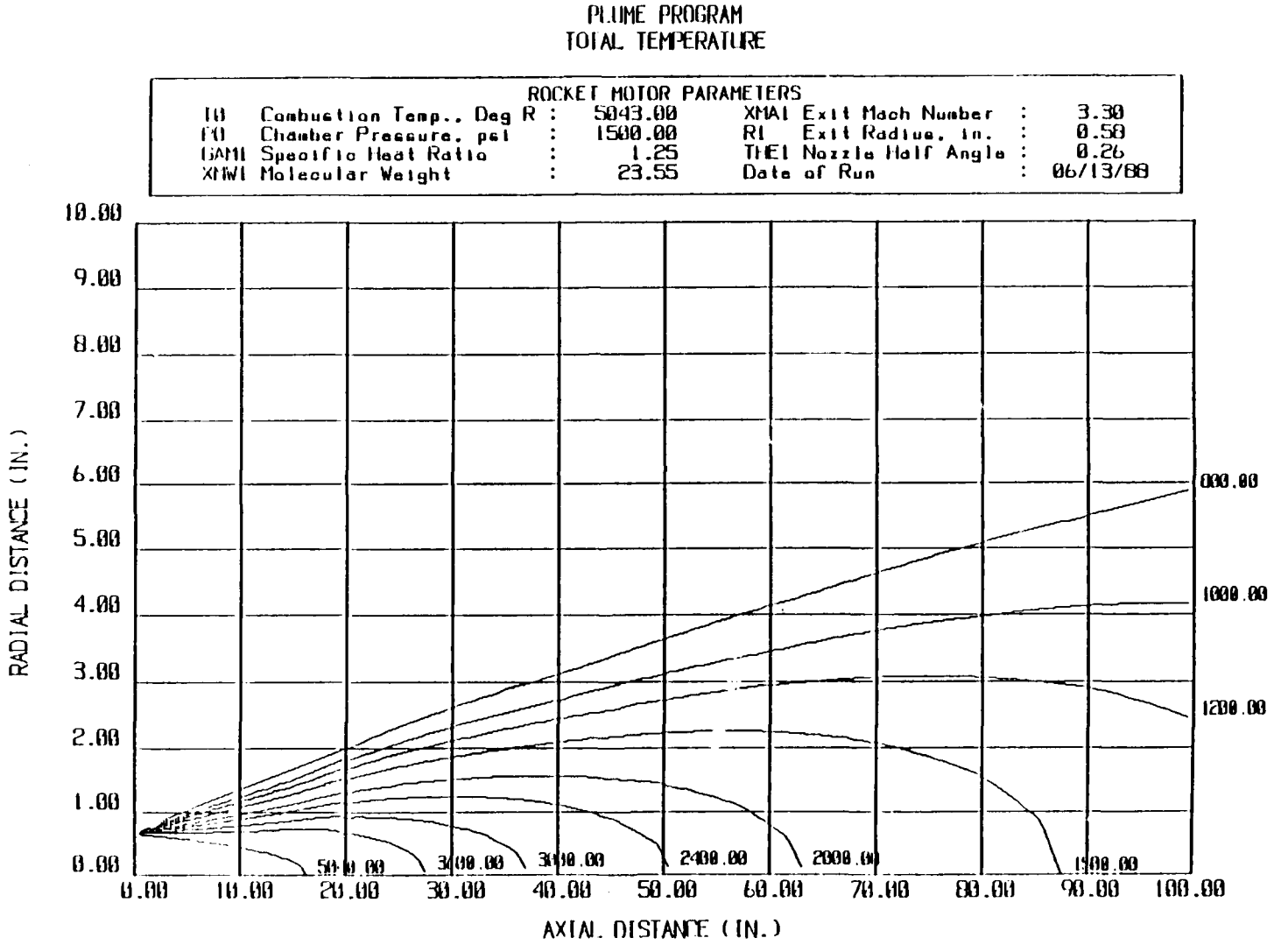

Figure 4. Typical plot of total temperature distribution within plume flowfield.

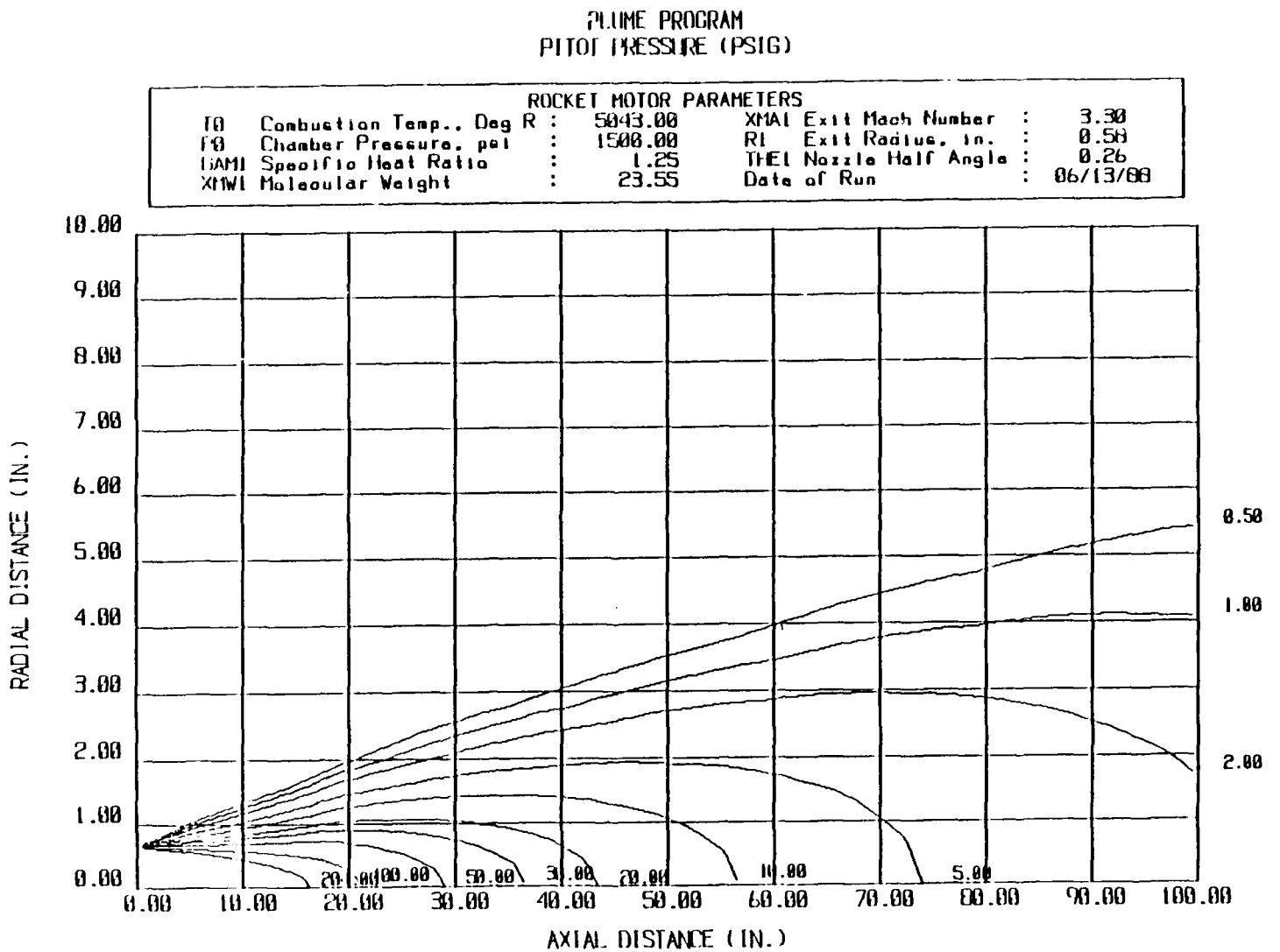

Figure 5. Typical plot of pitot pressure distribution within plume flowfield.

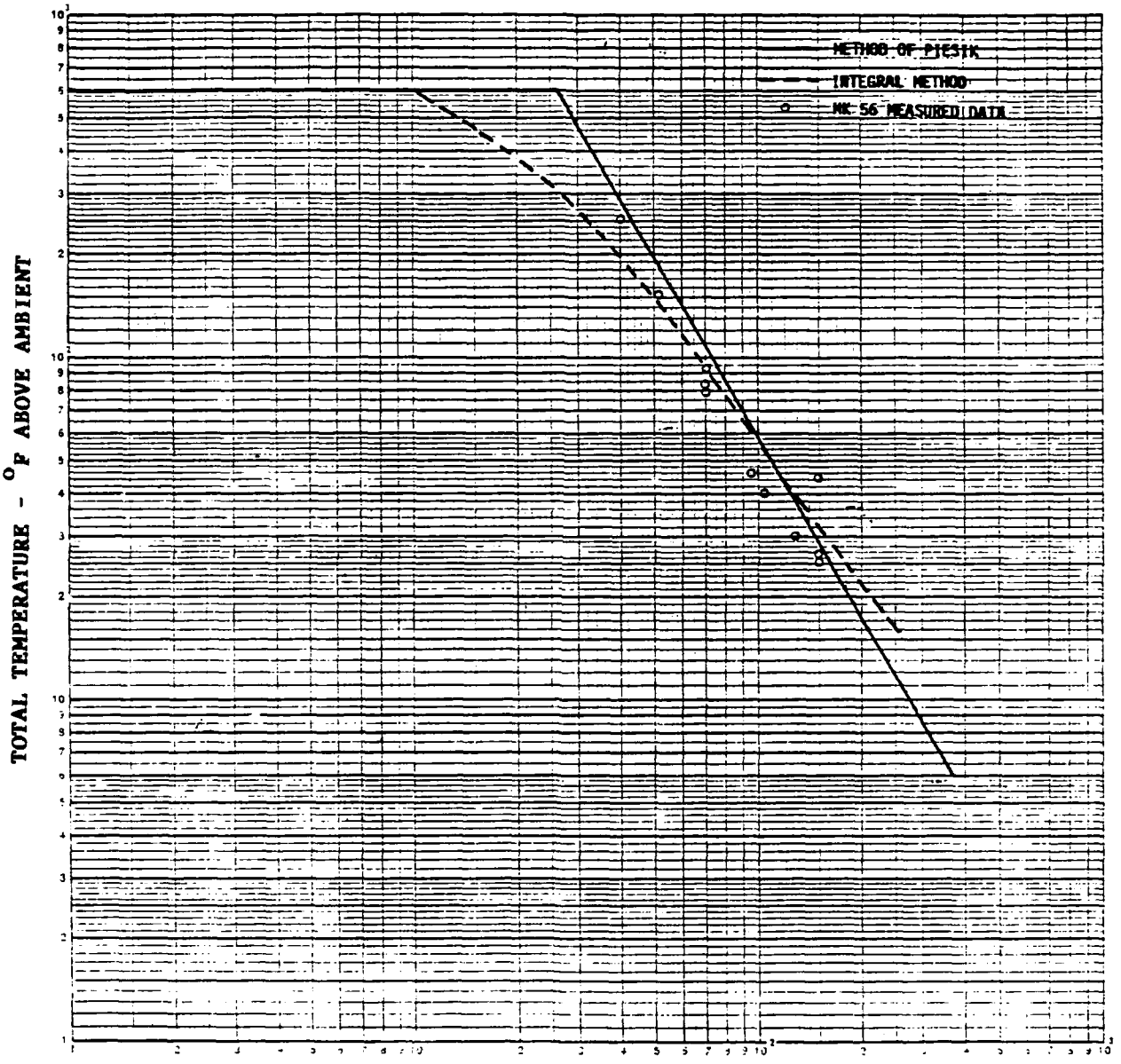

Figure 6. Centerline total temperature comparison.

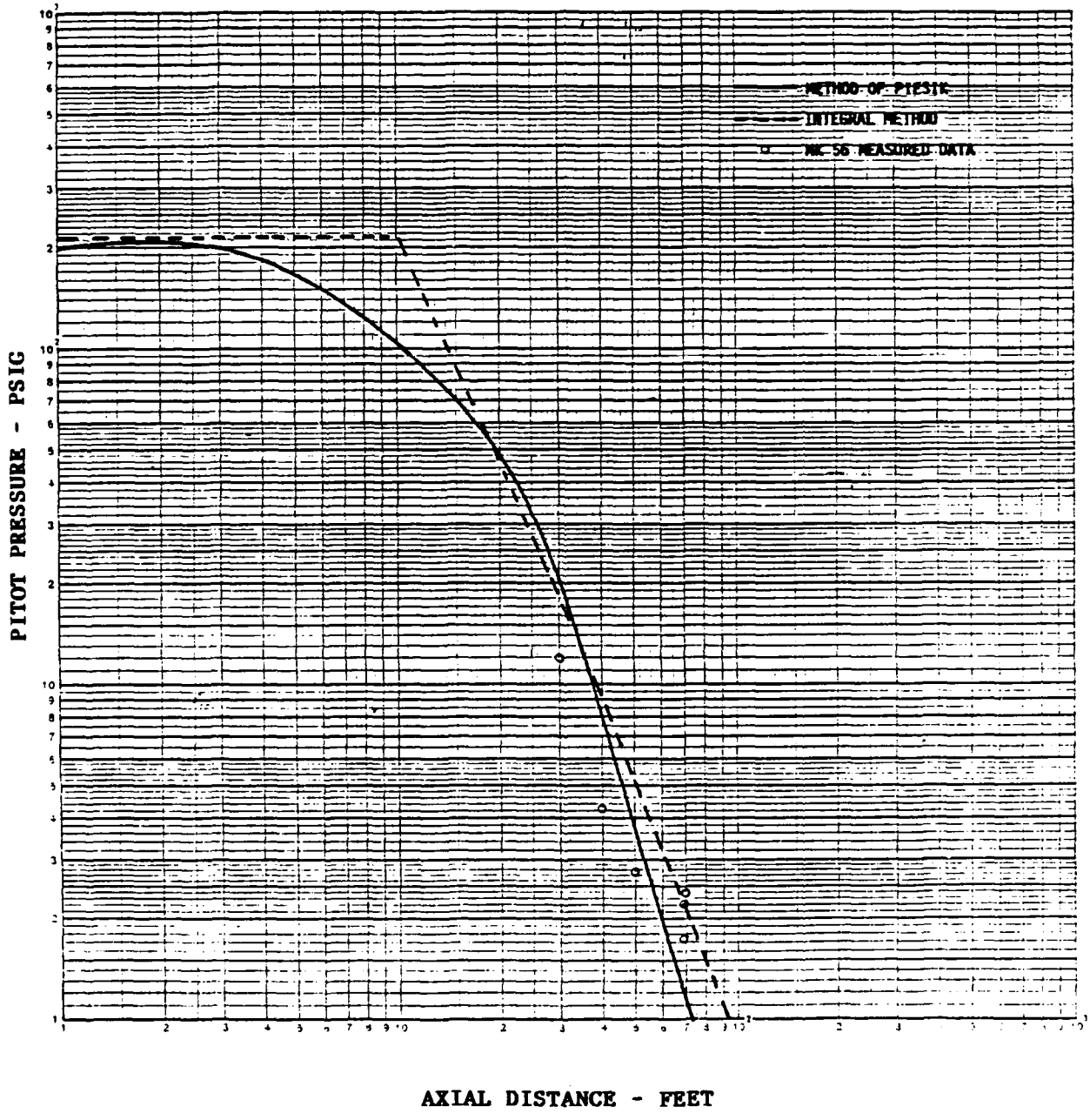

Figure 7. Centerline pitot pressure comparison.

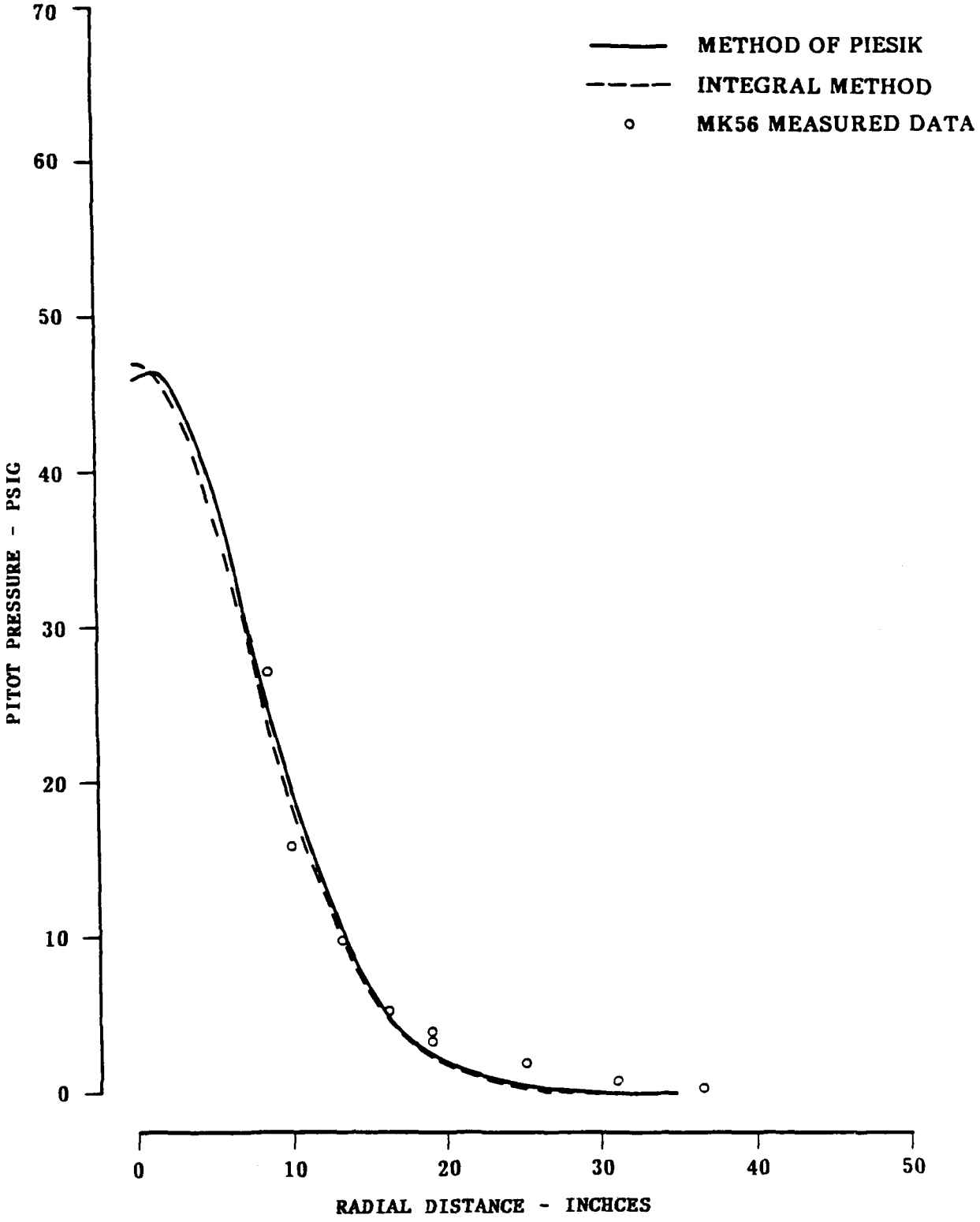

Figure 8. Radial pitot pressure comparison at x = 20.4 ft.

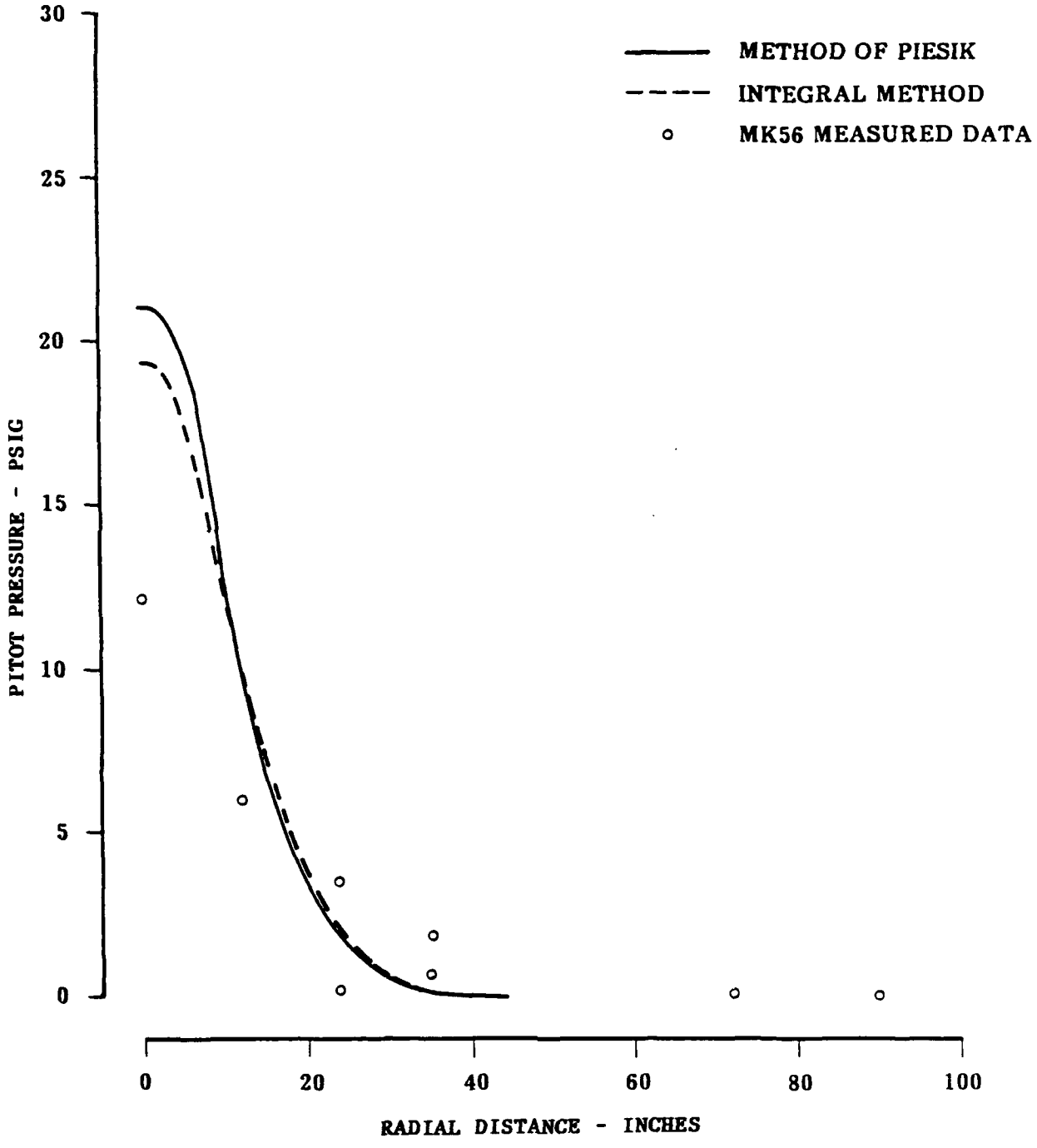

Figure 9. Radial pitot pressure comparison at x = 30 ft.

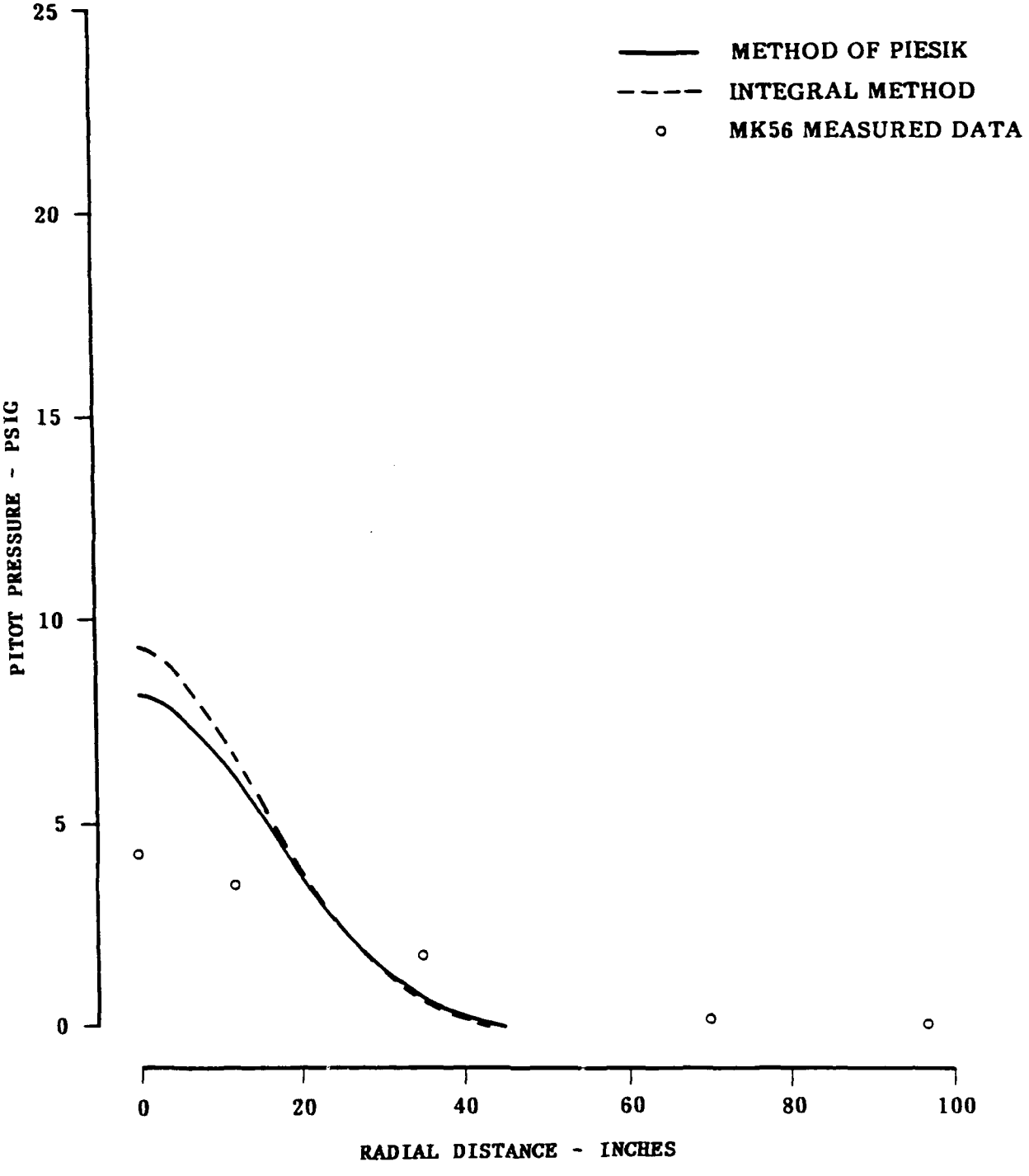

Figure 10. Radial pitot pressure comparison at x = 40 ft.

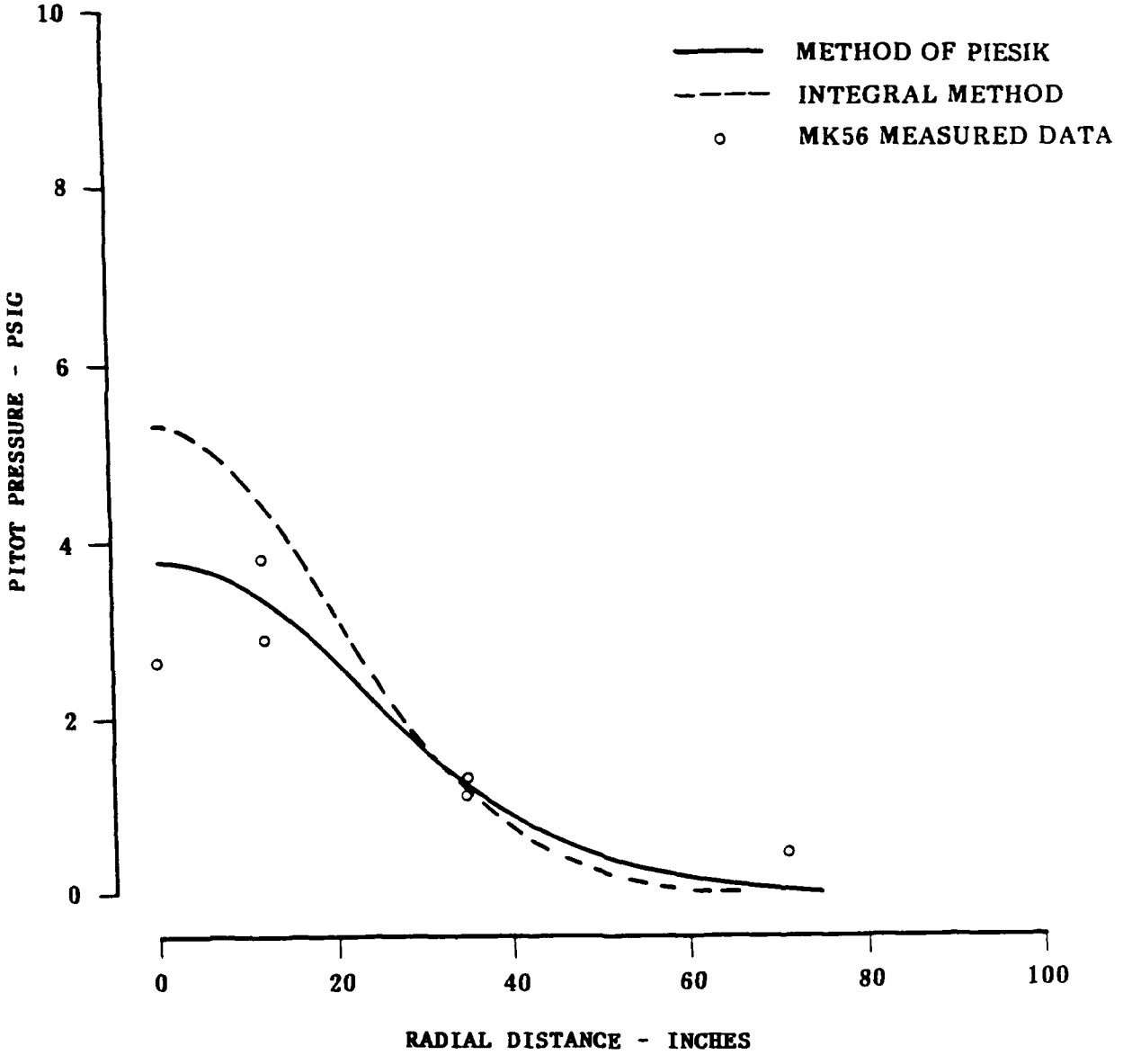

Figure 11. Radial pitot pressure comparison at x = 50 ft.

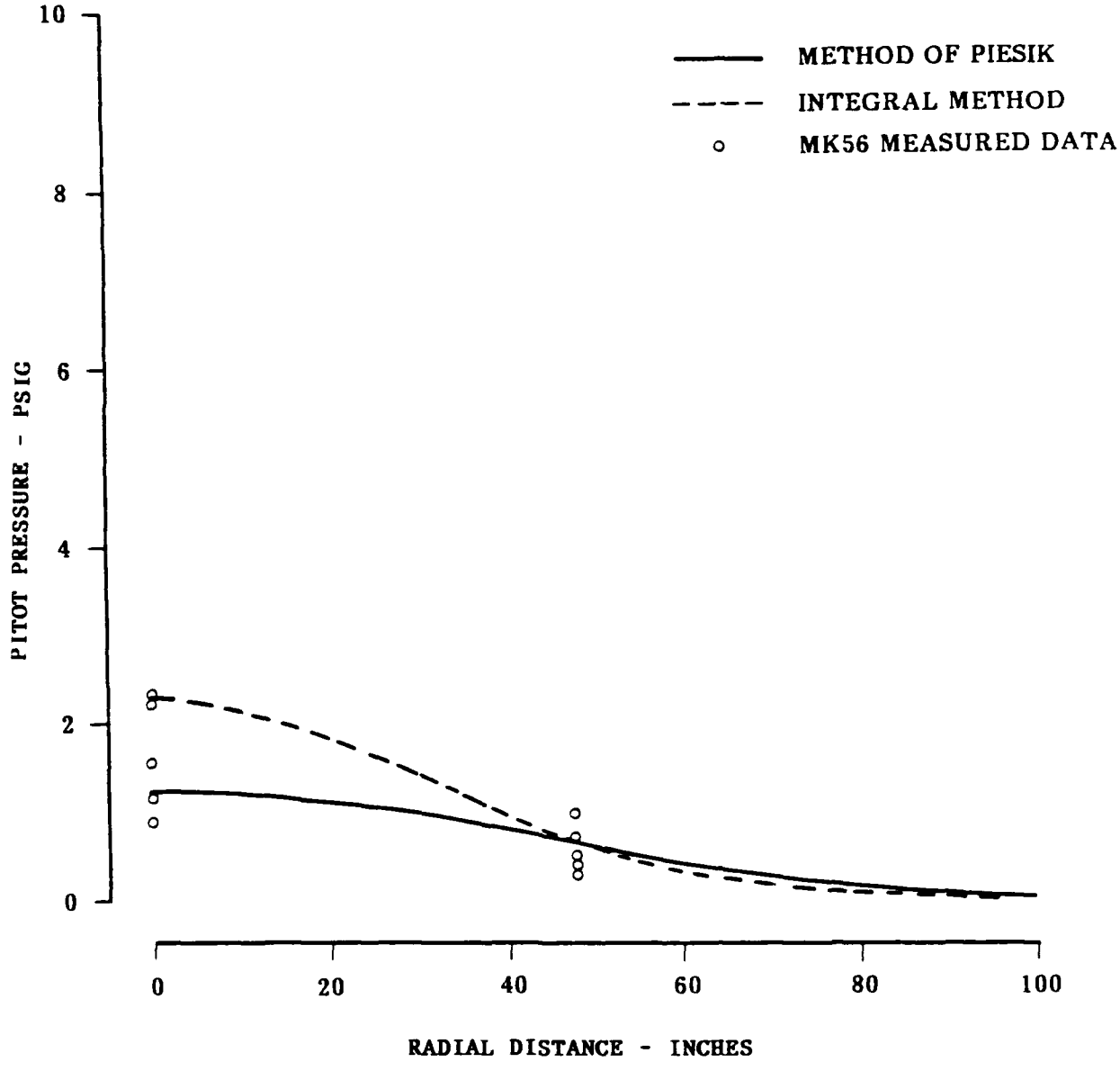

Figure 12. Radial pitot pressure comparison at x = 70.4 ft.

REFERENCES

1. Piesik, E.T., "Aluminized Propellants and a Method Defining Altitude Exhaust Plumes." Journal of Spacecraft and Rockets, Vol. 23, No. 2, pp. 215-21, March-April 1986.

2. Donaldson, C. and Gray, K., "Theoretical and Experimental Investigation of the Compressible Free Mixing of Two Dissimilar Gases." AIAA Journal, Vol. 4, No. 11, pp. 2017-25, November 1966.

3. Warren, W.R., "An Analytical and Experimental Study of Compressible Free Jets." Princeton University Department of Aeronautical Engineering, Rept. 381, 1957.

4. Reardon, J.E. and Word, E.C., "Rocket Exhaust Plume Analysis for Launchers." Report prepared under Contract DDAH01-76-C-0520 for the U.S. Army Missile Command, Redstone Arsenal, Huntsville, Alabama, March 1977.

5. Miller Jr., R.E., "Pressure and Temperature Measurements in Subsonic Region of the Standard Mark 56 Rocket Motor Exhaust." U.S. Naval Weapons Laboratory, Dahlgren, VA, NWS Tech. Rept. TR-2800, August 1972.

LIST OF SYMBOLS

c	Mass fraction
C_p	Specific heat at constant pressure
h	Enthalpy
I,J,F,G	Integrals defined in Appendix C
K	Mixing rate parameter
m	Molecular Weight
M	Mach Number
P	Pressure
r	Radial distance
R	Nondimensional radius, r/r_1
R_g	Universal gas constant
T	Temperature
u	Velocity
U	Nondimensional velocity, u/u_1
x	Axial distance
X	Nondimensional distance, x/r_1
γ	Specific heat ratio
θ	Nozzle half angle
λ	ln 2
ρ	Density
τ	Shear stress
ω	Prandtl-Meyer expansion angle

Superscripts

$^\circ$	Stagnation condition
$'$	Value at the exit of an underexpanded nozzle

Subscripts

$_1$	Exit plane conditions of fully expanded nozzle
$_\infty$	Ambient conditions
$_5$	Value where $u = \frac{1}{2}u_c$
c	Centerline conditions
d	Downstream of normal shock
i	At viscid-inviscid boundary (core boundary)
t	Chamber conditions
u	Upstream of normal shock
*	Variable value at some arbitrary radial location, r

APPENDIX A

DEVELOPMENT OF AN EXPRESSION FOR DENSITY AS A FUNCTION OF VELOCITY

APPENDIX A. Development of an Expression for Density as a Function of Velocity

In the following discussion, it is assumed that free mixing is taking place between an initially irrotational and axially symmetric stream of an ideal gas given by the subscript 1 and another ideal ambient gas given by subscript ∞. The gases are assumed to be inert chemically.

Taking the usual Crocco integral (and assuming it holds throughout the mixing region)

$$h + (u^2/2) = Au + B \tag{A.1}$$

and applying the boundary conditions $h = h_\infty$ when $u = 0$ and $h = h_1$ when $u = u_1$, the above equation becomes

$$h = h_\infty + (h_1^\circ - h_\infty)(u/u_1) - (u_1^2/2)(u/u_1)^2 \tag{A.2}$$

where

$$h_1^\circ = h_1 + u_1^2/2 \tag{A.3}$$

Also taking the concentration integral

$$c_\alpha + a_\alpha u = b \tag{A.4}$$

and considering the mixing of a rocket exhaust with the air surrounding it, there exist two mass fractions (c_α), c_1 and c_∞, where c_1 is the local mass fraction of the exhaust gas and c_∞ is the local mass fraction of the ambient or surrounding gas. Applying the boundary conditions, $c_1 = 0$ when $u = 0$ and $c_1 = 1$ when $u = u_1$, equation (A.4) becomes

$$c_1 = u/u_1 \tag{A.5}$$

Also applying the boundary conditions $c_\infty = 1$ when $u = 0$ and $c_\infty = 0$ when $u = u_1$, equation (A.4) becomes

$$c_\infty = 1 - (u/u_1) \tag{A.6}$$

Knowing the mass fractions, expressions for molecular weight and specific heat become

$$m = 1/[(1/m_1 - 1/m_\infty)(u/u_1) + (1/m_\infty)] \tag{A.7}$$

$$C_{p_1} = (C_{p_1} + C_{p_\infty})(u/u_1) + C_{p_\infty} \tag{A.8}$$

Assuming $h = C_pT$ with a constant pressure throughout the plume flowfield equal to the ambient pressure, and using the perfect

gas law, density can be written as

$$\rho = PmC_p/(Rh) \tag{A.9}$$

Substituting expressions (A.2), (A.7), and (A.8) into (A.9) yields

$$\rho = \frac{P_\infty[(C_{p_1}+C_{p\infty})(u/u_1)+C_{p\infty}]}{[(1/m_1-1/m_\infty)(u/u_1)+(1/m_\infty)][h_\infty+(h_1{}^\circ-h_\infty)(u/u_1)-(u_1{}^2/2)(u/u_1)^2]} \tag{A.10}$$

where density is now a function of known quantities and velocity.

APPENDIX B

DEVELOPMENT OF THE EQUATIONS FOR THE PLUME FLOWFIELD FOR BOTH THE CORE AND DEVELOPED REGIONS

APPENDIX B. Development of the Equations for the Plume Flowfield for both the Core and Developed Regions

B1. General Equations

The first step in solving equations (II.2) and (II.3) is to non-dimensionalize these equations and the expressions for u, Eqs.(II.4 and 5), τ, Eq.(II.8 and 9), and ρ, Eq.(A.10). Letting capital letters represent non-dimensionalized parameters such that $R = r/r_1$, $U = u/u_1$, and $X = x/r_1$, these expressions become

$$\frac{d}{dX}\int_0^{R_5} (\rho/\rho_\infty)U^2 R\,dR - (U_C/2)\frac{d}{dX}\int_0^{R_5} (\rho/\rho_\infty)UR\,dR = [\tau_5/(\rho_\infty u_1^2)]R_1 \qquad (B.1)$$

$$\int_0^{\infty} (\rho/\rho_\infty)U^2 R\,dR = 1/2(\rho_1/\rho_\infty) \qquad (B.2)$$

where

$$U = U_1 \times \exp[-\lambda(R^2-R_i^2)/(R_5^2-R_i^2)] \qquad (B.3)$$

$$(\tau_5/\rho_\infty u_1^2) = (\rho_5/\rho_\infty)[1/2K_C(R_5-R_i)](\partial U/\partial R) \qquad (B.4)$$

for the core region,

$$U = U_C \times \exp[-\lambda(R/R_5)^2] \qquad (B.5)$$

$$(\tau_5/\rho_\infty u_1^2) = (\rho_5/\rho_\infty)1/2KR_5U_C(\partial U/\partial R) \qquad (B.6)$$

for the developed region, and

$$(\rho/\rho_\infty) = [(\Phi U+1)/(\Gamma U+1)(1+AU-BU^2)] \qquad (B.7)$$

$$\Phi = (C_{p_1}/C_{p\infty})-1 \qquad \Gamma = (m_\infty/m_1)-1$$

$$A = (h_1^\circ/h_\infty)-1 \qquad B = 1/2(u_1^2/h_\infty)$$

B2. Development of Equations for the Developed Region

From Eq.(B.5) it can be shown that

$$(\partial U/\partial R) = -(\lambda U_C/R_5) \qquad (B.8)$$

and

$$RdR = -(R_s^2 dU/2\lambda U) \tag{B.9}$$

Expressions (B.5), (B.6), (B.7), (B.8), and (B.9) can be substituted into the two flow equations (B.1) and (B.2). (Note: For compactness, the density ratio is carried as ρ/ρ_∞, though it is a known function of u given by Eq.(B.7).) These substitutions and the necessary change in the limits yield

$$\frac{d}{dX}\left[R_s^2 \int_{U_s}^{U_c} (\rho/\rho_\infty) U dU\right] - (U_c/2)\frac{d}{dX}\left[R_s^2 \int_{U_s}^{U_c} (\rho/\rho_\infty) dU\right] =$$

$$-\lambda^2 K R_s U_c^2 (\rho/\rho_\infty) \tag{B.10}$$

and

$$\int_0^{U_c} (\rho/\rho_\infty) U dU = \lambda/R_s^2 (\rho_1/\rho_\infty) \tag{B.11}$$

Differentiating by parts, Eq.(B.10) becomes

$$R_s^2 \frac{d}{dX}\int_{U_s}^{U_c} (\rho/\rho_\infty) U dU - (U_c/2) R_s^2 \frac{d}{dX}\int_{U_s}^{U_c} (\rho/\rho_\infty) dU +$$

$$\int_{U_s}^{U_c} (\rho/\rho_\infty) U dU \frac{dR_s^2}{dX} - (U_c/2)\int_{U_s}^{U_c} (\rho/\rho_\infty) dU \frac{dR_s^2}{dX} =$$

$$-\lambda^2 K R_s U_c^2 (\rho_s/\rho_\infty) \tag{B.12}$$

The five indicated integrations in Eqs.(B.10 and B.11) can be solved closed form and are represented by the following notation

$$\int_{U_s}^{U_c} (\rho/\rho_\infty) U dU = I_c - I_s \tag{B.13}$$

$$\int_{U_s}^{U_c} (\rho/\rho_\infty) dU = J_c - J_s \tag{B.14}$$

$$\frac{d}{dX}\int_{U_s}^{U_c} (\rho/\rho_\infty) U dU = F_c \frac{dU_c}{dX} - F_s \frac{dU_s}{dX} \tag{B.15}$$

$$\frac{d}{dX}\int_{U_s}^{U_c}(\rho/\rho_\infty)\,dU = G_c\frac{dU_c}{dX} - G_s\frac{dU_s}{dX} \tag{B.16}$$

$$\int_{0}^{U_c}(\rho/\rho_\infty)\,U\,dU = I_c - I_0 \tag{B.17}$$

where I, J, F, and G are functions of U and are solved in Appendix C.

Substituting expressions (B.13-B.17) in Eqs.(B.11) and (B.12), yields

$$I_c - I_0 = \lambda(\rho_1/\rho_\infty)/R_s^2 \tag{B.18}$$

$$\left[\frac{U_c}{2}(G_c - \frac{1}{2}G_s) - (F_c - \frac{1}{2}F_s)\right]R_s^2\frac{dU_c}{dX} + \left[\frac{U_c}{2}(J_c - J_s) - (I_c - I_s)\right]\frac{dR_s^2}{dU_c}\frac{dU_c}{dX} =$$

$$\lambda^2 K R_s U_c^2(\rho_s/\rho_\infty) \tag{B.19}$$

Equation (B.18) can be rearranged to solve for R_s^2 as follows

$$R_s^2 = \lambda(\rho_1/\rho_\infty)/(I_c - I_0) \tag{B.20}$$

and

$$\frac{dR_s^2}{dU_c} = -\frac{\lambda(\rho_1/\rho_\infty)}{(I_c - I_0)}\frac{dI_c}{dU_c} = \frac{R_s^2 F_c}{(I_c - I_0)} \tag{B.21}$$

where

$$F_c = dI_c/dU_c$$

Substituting Eqs.(B.20) and (B.21) into Eq.(B.19), provides

$$\left\{\frac{(I_C-I_0)\left[\frac{U_C}{2}(G_C-\frac{1}{2}G_5)-(F_C-\frac{1}{2}F_5)\right]-F_C\left[\frac{U_C}{2}(J_C-J_5)-(I_C-I_5)\right]}{\lambda^2(\rho_5/\rho_\infty)U_C^2(I_C-I_0)}\right\} X$$

$$R_5\frac{dU_C}{dX} = K \qquad (B.22)$$

which is of the form

$$f(U_C)(dU_C/dX) = K \qquad (B.23)$$

Due to the complex nature of $f(U_C)$ and the fact K is a function of M_5 and therefore X, an equation of this form must be solved numerically. Since it is a first order, non-linear, ordinary differential equation, it was solved using the Runge-Kutta method in conjunction with a Taylor series approximation.

B3. Development of Equations for the Core Region

In the core region, Eqs.(B.8) and (B.9) become

$$(\partial U/\partial R) = -\lambda R_5/(R_5^2 - R_i^2) \qquad (B.24)$$

and

$$RdR = [(R_5^2 - R_i^2)dU/2\lambda U] \qquad (B.25)$$

As was done previously, expressions (B.3), (B.4), (B.7), (B.24), and (B.25) can be substituted into equations (B.1) and (B.2). Performing the necessary integration and using the integral notation of Eqs.(B.13-B.17), two expressions similar to Eqs.(B.18) and (B.19) can be obtained. (Note: The integration in the core region must be broken into two parts. These are $0 \le R \le R_i$, where conditions are constant, and $R_i \le R < \infty$.)

$$I_1 - I_0 = \frac{\lambda(\rho_1/\rho_\infty)(1 - R_i^2)}{R_5^2 - R_i^2} \qquad (B.26)$$

$$\left[\begin{matrix}1\\-(J_1-J_5)-(I_1-I_5)\\2\end{matrix}\right]\frac{d(R_5^2 - R_i^2)}{dX} =$$

$$\frac{K_C\lambda^2(\rho_1/\rho_\infty)R_5^2}{R_5 - R_i} + \frac{\lambda(\rho_1/\rho_\infty)}{2}\frac{dR_i^2}{dX} \qquad (B.27)$$

Equation (B.26) can be rearranged to provide R_5^2 as a function of R_i, so that

$$R_5^2 = \frac{\lambda(\rho_1/\rho_\infty)(1 - R_i^2)}{I_1 - I_0} + R_i^2 \qquad (B.28)$$

and

$$\frac{d(R_5^2 - R_i^2)}{dX} = -\frac{\lambda(\rho_1/\rho_\infty)}{I_1 - I_0}\frac{dR_i^2}{dX} \qquad (B.29)$$

Substituting Eqs.(B.28) and (B.29) into Eq.(B.27) and letting

$$\phi = \lambda(\rho_1/\rho_\infty)/(I_1 - I_0) \qquad (B.30)$$

$$\psi = \frac{(\rho_1/\rho_5)}{\lambda}\left\{\frac{1}{(I_1-I_0)}\left[(J_5-J_1)-2(I_5-I_1)\right]- 1\right\} \qquad (B.31)$$

equation (B.27) becomes

$$\frac{\psi R_i(R_5 + R_i)}{\phi + (1 - \phi)R_i^2}dR_i = K_C dX \qquad (B.32)$$

Since ψ and K are constant in the core region Eq.(B.33) can be solved in closed form, yielding

$$\psi\int_1^{R_i^*}\left[\frac{R_i}{[\phi+(1-\phi)R_i^2]^{\frac{1}{2}}} + \frac{R_i^2}{\phi+(1-\phi)R_i^2}\right]dR_i = K\int_0^{X^*} dX \qquad (B.33)$$

This integration will yield two solutions, depending on the value of ϕ. When ϕ is greater than zero the integration yields

$$X = \frac{\psi}{K}\left\{\frac{2-[\phi-(\phi-1)R_i^2]^{\frac{1}{2}}-R_i}{\phi - 1} + \frac{\phi}{2T(\phi-1)} \times \ln\left[\frac{(\phi+R_iT)(\phi-T)}{(\phi-R_iT)(\phi+T)}\right]\right\} \qquad (B.35)$$

where

$$T = [\phi(\phi-1)]^{\frac{1}{2}}$$

and when ϕ is less than zero the integration yields

$$X = \frac{\psi}{K}\left\{\frac{[\phi+(1-\phi)R_i^2]^{\frac{1}{2}}+R_i-2}{1 - \phi} + \frac{\phi}{T(1-\phi)} \times \left[\tan^{-1}\left(\frac{T}{\phi}\right) - \tan^{-1}\left(\frac{R_iT}{\phi}\right)\right]\right\} \qquad (B.36)$$

where

$$T = [\phi(1-\phi)]^{\frac{1}{2}}$$

B3. Solving for Local Flowfield Parameters

Having solved the necessary equations to determine the velocity at any point within the flowfield, flowfield parameters of interest can be calculated as follows

$$C_p = C_{p\infty}(\Phi U'+1) \qquad (B.37)$$

$$m = m_\infty/(\Gamma U'+1) \qquad (B.38)$$

$$\gamma = 1/(1-R_g/mC_p) \qquad (B.39)$$

$$T = T_\infty(1+AU-BU^2)/(\Phi U+1) \qquad (B.40)$$

$$M = (U)(u_1)/(\gamma R_g T/m)^{-\frac{1}{2}} \qquad (B.41)$$

$$P^\circ = P\left[1 + \frac{\gamma-1}{2} M^2\right]^{\zeta} \qquad (B.42)$$

$$P_d^\circ = P_u^\circ \left[\frac{(\gamma+1)M_u^2}{(\gamma-1)M_u^2+2}\right]^{\zeta} \left[\frac{\gamma+1}{2\gamma M_u^2-\gamma+1}\right]^{\mu} \tag{B.43}$$

$$T^\circ = T\left[1 + \frac{\gamma-1}{2} M^2\right] \tag{B.44}$$

where

$U' = u/u_c$

$\zeta = \gamma/(\gamma-1)$

$\mu = 1/(\gamma-1)$

APPENDIX C

DERIVATION OF I, J, F, and G

APPENDIX C. Derivation of I, J, F, and G

From Eqs.(B.13) and (B.17)

$$I = \int (\rho/\rho_\infty) U dU \tag{C.1}$$

Substituting Eq.(B.7) in for ρ/ρ_∞ in Eq.(C.1), yields

$$I = \int \frac{(\Phi U^2+U)\ dU}{(\Gamma U+1)(1+AU-BU^2)} \tag{C.2}$$

This integral can be solved in closed form by utilizing partial fractions. The integrand can be expressed as

$$\frac{\xi_1}{\Gamma U+1} + \frac{\xi_2 U+\xi_3}{(1+AU-BU^2)} \tag{C.3}$$

where ξ_1, ξ_2, and ξ_3 are constants. Equation (C.2) now becomes

$$I = \xi_1 \int \frac{dU}{\Gamma U+1} + \xi_2 \int \frac{UdU}{(1+AU-BU^2)} + \xi_3 \int \frac{dU}{(1+AU-BU^2)} \tag{C.4}$$

Carrying out the integration yields

$$I = \frac{\xi_1}{\Gamma} \ln(\Gamma U+1) - \frac{\xi_2}{2B} \ln(1+AU-BU^2) - \left[\frac{\xi_2 A}{B} + 2\xi_3\right] \frac{1}{(4B+A^2)^{\frac{1}{2}}} \tanh^{-1}\left[\frac{-2BU+A}{(4B+A^2)^{\frac{1}{2}}}\right] \tag{C.5}$$

The constants ξ_1, ξ_2, and ξ_3 can be evaluated by equating like terms

$$\frac{\Phi U^2+U}{(\Gamma U+1)(1+AU-BU^2)} = \frac{\xi_1}{\Gamma U+1} + \frac{\xi_2 U+\xi_3}{(1+AU-BU^2)} \tag{C.6}$$

such that

$$\Phi = -\xi_1 B + \xi_2 \Gamma$$

$$1 = \xi_1 A + \xi_2 + \xi_3 \Gamma$$

$$0 = \xi_1 + \xi_3$$

yielding

$$\xi_1 = \frac{-(\Phi-\Gamma)}{B+\Gamma A-\Gamma^2} \tag{C.7}$$

$$\xi_2 = \frac{B+\Phi(A-\Gamma)}{B+\Gamma A-\Gamma^2} \tag{C.8}$$

$$\xi_3 = \frac{(\Phi-\Gamma)}{B+\Gamma A-\Gamma^2} \tag{C.9}$$

From Eq.(B.14)

$$J = \int (\rho/\rho_\infty)\,dU \tag{C.10}$$

Substituting Eq.(B.7) for ρ/ρ_∞, Eq.(C.10) becomes

$$J = \int \frac{(\Phi U+1)\,dU}{(\Gamma U+1)(1+AU-BU^2)} \tag{C.11}$$

As above the integral can be solved in closed form using partial fractions. Equation (C.11) becomes

$$J = \delta_1 \int \frac{dU}{\Gamma U+1} + \delta_2 \int \frac{U\,dU}{1+AU-BU^2} + \delta_3 \int \frac{dU}{1+AU-BU^2} \tag{C.13}$$

and integrating yields

$$J = \frac{\delta_1}{\Gamma} \ln(\Gamma U+1) - \frac{\delta_2}{2B} \ln(1+AU-BU^2) - \left[\frac{\delta_2 A}{B} + 2\delta_3\right] \frac{1}{(4B+A^2)^{\frac{1}{2}}} \tanh^{-1}\left[\frac{-2BU+A}{(4B+A^2)^{\frac{1}{2}}}\right] \tag{C.14}$$

where

$$\delta_1 = \frac{\Gamma(\Phi-\Gamma)}{B+\Gamma A-\Gamma^2} \tag{C.15}$$

$$\delta_2 = \frac{B(\Phi-\Gamma)}{B+\Gamma A-\Gamma^2} \tag{C.16}$$

$$\delta_3 = \frac{B+\Gamma(A-\Phi)}{B+\Gamma A-\Gamma^2} \qquad (C.17)$$

From Eq.(B.15) F can be defined as follows

$$F = dI/dU$$

which is identical to Eq.(C.3), yielding

$$F = \frac{\xi_1}{\Gamma U+1} + \frac{\xi_2 U+\xi_3}{(1+AU-BU^2)} \qquad (C.18)$$

where ξ_1, ξ_2, and ξ_3 have been previously defined in Eqs.(C.7-9).

Also from Eq.(B.16), G can be defined as

$$G = dJ/dU$$

which leads to

$$J = \frac{\delta_1}{\Gamma U+1} + \frac{\delta_2 U+\delta_3}{(1+AU-BU^2)} \qquad (C.18)$$

where δ_1, δ_2, and δ_3 have been defined in Eqs.(C.15-17).

APPENDIX D

MODIFICATIONS TO ORIGINAL METHOD TO ALLOW FOR NOZZLE UNDEREXPANSION

APPENDIX D. Modifications to Original Method to Allow for Nozzle Underexpansion

The integral method outlined in Appendices A, B, and C applies to ideally or fully expanded nozzles. Many Army missiles have rocket nozzles that are slightly underexpanded. To allow for some underexpansion, the method was modified with a circular arc curve fit [4]. The details of this modification are given below.

The radius R in the momentum integral method is nondimensionalized with respect to r_1, where r_1 is the radius at the exit of a fully expanded nozzle. In most cases, the rocket motor to be analyzed will not be fully expanded. Given an underexpanded rocket motor with a nozzle exit radius given by r_1' and an exit Mach number given by M_1', the flow must be expanded to ambient pressure to obtain the necessary parameters of r_1 and M_1. The Mach number at ambient pressure is given by

$$M_1 = \sqrt{\frac{2}{\gamma_1 - 1}\left[\left(\frac{P_t}{P_\infty}\right)^{\eta} - 1\right]} \qquad (D.1)$$

where

$$\eta = (\gamma_1 - 1)/\gamma_1$$

and the associated exit radius for a fully expanded nozzle is

$$r_1 = \sqrt{\frac{M_1'}{M_1}}\left[\frac{2+(\gamma_1-1)M_1^2}{2+(\gamma_1-1)M_1'^2}\right]^{\vartheta} r_1' \qquad (D.2)$$

where

$$\vartheta = (\gamma_1 + 1)/[4(\gamma_1 - 1)]$$

Using the fully expanded parameters of r_1 and M_1, the core region boundary R_i can be calculated according to the momentum integral relations developed in Appendix B. Then using the real radius r_1' a circular arc can be drawn from the exit of the underexpanded nozzle out to the point of full expansion where, $r = r_1$. The circular arc from the real (underexpaned) nozzle and the core boundary for the ideal nozzle will overlap as is shown in Figure D.1. Merging these curves will create a smooth curve describing the core region, which expands from the lip of the exit of the underexpanded nozzle out to ambient pressure and then behaves as per the momentum integral equations.

The slope of the circular arc at the exit of the underexpanded nozzle is assumed to be the Prantl-Meyer expansion angle given by

$$\omega = \frac{1}{\beta}\tan^{-1}\left[\sqrt{\beta^2(M_1^2-1)}\right] - \frac{1}{\beta}\tan^{-1}\left[\sqrt{\beta^2(M_1'^2-1)}\right] - \tan^{-1}\left[\sqrt{M_1^2-1}\right] + \tan^{-1}\left[\sqrt{M_1'^2-1}\right] + \theta_1 \qquad (D.3)$$

where

$$\beta = \sqrt{(\gamma_1-1)/(\gamma_1+1)}$$

Knowing the expansion angle, the axial extent, b, of the expansion region is

$$b = \tan(\omega)(r_1'-a) \qquad (D.4)$$

and the radius r of the circular arc can be determined as a function of x, given by

$$r = \sqrt{(r_1'-a)^2+\tan^2(\omega)(r_1'-a)^2-[x-\tan(\omega)(r_1'-a)]^2}+a \qquad (D.5)$$

where

$$a = (r_1-r_1'\sec(\omega))/(1-\sec(\omega))$$

The local Mach number can be calculated from the area Mach number relationship

$$\frac{r}{r_1'} = \sqrt{\frac{M_1'}{M}}\left[\frac{2+(\gamma_1-1)M^2}{2+(\gamma_1-1)M_1'^2}\right]^{\vartheta} \qquad (D.6)$$

which provides Mach number as a function of x. This equation for Mach number was solved using a Newton iterative technique. Knowing the Mach number, the velocity, pressure, and temperature within the expansion region ($x < b$) can be calculated from basic isentropic one-dimensional relations. The values of γ, C_p, amd m are assumed constant throughout the core in this region and equal to their values at the nozzle exit.

Since, with the momentum integral method all linear values

are nondimensionalized with respect to a fully expanded nozzle, r_i and r_s begin at the ideal nozzle lip. In order to conform to the physical geometry of an underexpanded nozzle, r_i and r_s must begin at the underexpanded nozzle lip. This is achieved by using the relation

$$\Delta r = r_1 - r$$

where r is given by Eq.(D.5). The Δr is subtracted from r_i and r_s within the expansion region ($x < b$), forcing these curves to begin at the exit of the underexpanded nozzle. Outside the expansion region ($x > b$) Δr is equal to zero.

APPENDIX E

PROGRAM LISTING

24 June 1988 - 01:08 PM (Pages E-1 thru E-13)

24 June 1988 - 01:13 PM (Pages E-14 thru E-18)

```
/*      EXHAUST PLUME PROGRAM                                          */
/*                                                                     */
/*      Prepared by                                                    */
/*      R.W.NOURSE                                                     */
/*      STRUCTURES DIRECTORATE                                         */
/*      RESEARCH, DEVELOPMENT, AND ENGINEEERING CENTER                 */
/*      U.S.ARMY MISSILE COMMAND                                       */
/*      REDSTONE ARSENAL, AL                                           */
/*                                                                     */
/*                                                                     */
/*      This program utilizes the Momentum Integral Technique          */
/*      to predict the thermodynamic properties of decaying            */
/*      rocket exhaust plumes.  This program will accurately           */
/*      predict the plume parameters of pressure, velocity,            */
/*      and temperature at discrete axial and radial locations.        */
/*                                                                     */
/*                                                                     */
/*      The user is prompted by the program for the required           */
/*      input.  This input includes;                                   */
/*                                                                     */
/*              1. Ambient pressure and temperature                    */
/*              2. Chamber pressure and temperature                    */
/*              3. Specific heat ratio of exhaust gases                */
/*              4. Molecular wieght of exhaust gases                   */
/*              5. Mach number at the exit of the nozzle               */
/*              6. Exit nozzle radius                                  */
/*              7. Nozzle half angle                                   */
/*                                                                     */
/*                                                                     */
/*                                                                     */
#include <stdio.h>
#include <math.h>
#include <fcntl.h>
double t2,p2;                   /* Prop of air temp, pres             */
double t0,p0,gam1,xmw1,xma1;    /* Prop of propellant                 */
double r1,the1;                 /* Exit plane geom                    */
double gam2 = (double) 1.4;     /* Specific heat ratio                */
double rgas = (double) 49720.;  /* Real gas constant                  */
double xmw2 = (double) 28.97;   /* Molecular wt of air                */
double xk = (double) 0.025;     /* Mixing factor                      */
double cp1,cp2;                 /* Specific heat                      */
double h1,h2;                   /* Enthalpy                           */
double rho0,rho1,rho2,rho5;     /* Density                            */
double t1;                      /* Temperature at nozl exit           */
double a1;                      /* Speed of sound at nozl exit        */
double cons1 = (double) 144.0 * (double) 32.174;
double xmb;                     /* Reference nozl mach number         */
double rb;                      /* Reference nozl radius              */
double omga;                    /* Expansion angle                    */
double aa;
double xb;                      /* Axial length of Region I           */
double t;                       /* Temperature                        */
double asound;                  /* Speed of Sound                     */
double u1;                      /* Exit Plane Velocity                */
```

```
double uc;                      /* Centerline velocity                */
double u5;                      /* Half velocity  u5 = .5 ( uc)       */
double a,b,xm,p,xlam;           /* Constants of integration           */
double uux,uu;                  /* Non-dimensionalized velocity ratio */
                                /* U = u/u1                           */
double xi,xj,xi1,xj1,xi0;
double xi5,xj5;
double psi,phi,tau;
double xf,xg;
double xf5,xfc,xg5,xgc,xic,xjc;
double z1,z2,z3,z4,z5,z6;       /* Parameters for D.E.                */
double xxmax;                   /* Non-dimensionalized max length of  */
                                                                      /* c
double xmax;                    /* Maximum length of core region      */
double x;                       /* Axial distance                     */
double rr,r;                    /* Radius of circular arc, Region I   */
double dr;                      /* DR = RB - R                        */
double xmai;                    /* Mach number Region I               */
double rri;                     /* Non-dimensionalized radius of core */
                                                                      /* r
double ri;                      /* Radius of core region              */
double rro;                     /* Non-dimensionalized radius of plume */
                                                                      /* b
double ro;                      /* Radius of plume boundary           */
double rr5;                     /* Non-dimensionalized half velocity  */
                                /* radius, RR5 = R5 / RB              */
double r5;                      /* Half Velocity Radius               */
double uuc,uucc;
double duc;
double del,delx;
double cp,xmw,gam,mach,mach5,presu,presd;
double xlimit, xprt, cnt;
int fh;
struct _dat
{
        double dis;
        double rad[11];
        double tem[11];
        double mac[11];
        double pre[11];
} dat;
FILE *prt;

main()
{
        int i;
    int byte;
        int j;
        int ic;
        double temp;
        double v[4];
        double w[3];
        void bound();
        void calc();
```

```
        void region1();
        void viscid();
        void solvde();
        void mix();
        void factor();
        double desub();
        double tstep;
        double tstep1 = (double)30.0;
        double tstep2 = (double)0.1;
        double tstep3 = (double)0.5;
        if ((fh=creat("PLUME.DAT",0666)) == -1)
        {
                printf("COULD NOT OPEN DATA FILE");
                exit(1);
        }
        if ((prt = fopen("PLUME.PRT","w")) == NULL)
        {
                printf("\n\nCOULD NOT OPEN PRINT FILE\n\n");
                exit(1);
        }
        printf("\nPROPERTIES OF AMBIENT AIR (TEMP/PRES) : ");
/*    scanf("%f%f",&t2,&p2); */
        t2 = 540.; p2 = 14.7;
        printf("\nPROPERTIES OF PROPELLANT GAS (T0,P0,GAM1,XMW1,XMA1) : ");
        scanf("%f%f%f%f%f",&t0,&p0,&gam1,&xmw1,&xma1);
        printf("\nEXIT PLANE GEOMETRY (R1,THE1) : ");
    scanf("%f%f",&r1,&the1);
    printf("\nMAXIMUM AXIAL DISTANCE : ");
    scanf("%f",&xlimit);
    printf("\nPRINT INTERVAL : ");
    scanf("%f",&xprt);
    cnt=xprt;
    printf("XCOUNT = %10.4f\n",cnt);
/*
        PROPERTIES OF AMBIENT AIR
*/
    dat.rad[0]=t0;
    dat.rad[1]=p0;
    dat.rad[2]=gam1;
    dat.rad[3]=xmw1;
    dat.rad[4]=xma1;
    dat.rad[5]=r1;
    dat.rad[6]=the1;

        if ((byte = write(fh,(char*)&dat,(unsigned int)sizeof(dat))) == -1)
        perror("");
    fprintf(prt,"\n     PROPERTIES OF AMBIENT AIR\n");
    fprintf(prt,"\n     TEMP                = %10.4f   PRES              = %10.4f\n\n",t2,p2);
    fprintf(prt,"\n     ROCKET MOTOR PARAMETERS\n");
    fprintf(prt,"\n     COMBUSTION TEMP     = %10.4f   CHAMBER PRES      = %10.4f\n",t0,p0);
    fprintf(prt,"     SPECIFIC HEAT RATIO = %10.4f   MOLECULAR WEIGHT   = %10.4f\n",gam1,xmw1);
    fprintf(prt,"     RADIUS OF EXIT      = %10.4f   NOZZLE HALF ANGLE  = %10.4f\n",r1,the1);
    fprintf(prt,"     EXIT MACH NUMBER    = %10.4f\n\n",xma1);
```

```
/*  Calculate ambient specific heat                                    */

        cp2 = (gam2*(rgas/xmw2))/(gam2 - 1.0);

/*  Calculate ambient air enthalpy      */

        h2 = cp2 * t2;

/*  Calculate ambient air density       */

        rho2 = (p2 * cons1 * xmw2) / (rgas * t2);


/*
        CALCULATE PROPERTIES OF PROPELLANT GAS
        SPECIFIC HEAT
*/

        cp1 = (gam1*(rgas/xmw1))/(gam1 - 1.0);

/*  Stagnation entalphy */

        h1 = cp1 * t0;

/* Density  */

        rho0 = (cons1 * p0 * xmw1) / (rgas * t0);

/* Speed of sound   */

        t1 = t0 / ( (double) 1.0 + (double) 0.5 * (gam1 - (double) 1.0)
                        * (xma1 * xma1));
        a1 = sqrt(gam1*(rgas/xmw1)*t1);
        bound();

/* Calculate local velocity for reference nozl  */

        t = t0 / ( (double) 1.0 + (double) 0.5 *(gam1 - (double) 1.0)
                        * (xmb*xmb));
        asound = sqrt(gam1*(rgas/xmw1)*t);
        uc = xmb * asound;
        u1=uc;
        uuc=uc/u1;
/* Calculate constants used for integration  */

        a = ( h1 / h2 ) - (double) 1.0;
        b = ( u1 * u1 ) / ( (double) 2.0 * h2);
        xm = ( xmw2 / xmw1 ) - (double) 1.0;
        p = ( cp1 / cp2 ) - (double) 1.0;
        xlam = log((double) 2.0);

        uux=(double)1.0;
```

>>>> plmrpt.c 06/24/88 01:08p page: 004

```
        rho1 = rho2*(p*uux+(double)1.0)/((xm*uux+(double)1.0)
                        *((double)1.0+a*uux-b*uux*uux));
        calc(0,0);
        xi1=xi;
        xj1=xj;
        uux=(double)0.;
        calc(0,0);
        xi0=xi;
        u5=(double)0.5*u1;
        uux=u5/u1;
        calc(1,0);
        if (phi < (double)1.0)
        {
                xxmax=(psi/xk)*((sqrt(phi)-(double)2.0)/((double)1.0-phi)
                        +(phi/(tau*((double)1.0-phi)))*atan(tau/phi));
        } else
        {
                xxmax=(psi/xk)*(((double)2.0-sqrt(phi))/(phi-(double)1.0)
                 +(phi/((double)2.0*tau*(phi-(double)1.0)))*log((phi-tau)/(phi+tau)));
        }
        xmax=xxmax*rb;
        fprintf(prt,"     MAXIMUM EXTENT OF CORE REGION = %10.4f\n\n\n\n\n\n",xmax);
    del = tstep2;
        x = (double)0.0;
    while(x <= xlimit)
        {
                if (x <= xb)
                {
                        temp=r1-aa;
                        temp*=temp;
                        r=sqrt(temp+tan(omga)*tan(omga)*temp-(x-tan(omga)*(r1-aa))*
                   (x-tan(omga)*(r1-aa)))+aa;
                        region1();
             asound = sqrt(gam1*(rgas/xmw1)*t);
             uc=xmai*asound;
             uuc=uc/u1;                        */
                        viscid();
                        dr=rb-r;
                        ri=rri*rb-dr;
                        rr5=sqrt(phi+((double)1.0-phi)*rri*rri);
                        r5=rr5*rb-dr;
                        temp=rri-dr/rb;
                        temp*=temp;
                        rro=sqrt((((pow(rr5-dr/rb,(double)2.0)-temp)*log((double).02))
                           /-xlam)+temp);
                        ro=rro*rb;
                } else
                if (x <= xmax)
                {
                        viscid();
                        ri=rri*rb;
                        rr5=sqrt(phi+((double)1.0-phi)*rri*rri);
                        r5=rr5*rb;
                        rro=sqrt(((rr5*rr5-rri*rri)*log((double)0.02))/-xlam+rri*rri);
```

```
                ro=rro*rb;
                mix();
            } else
            if (x > xmax)
            {
                uc=uuc*u1;
                u5=(double)0.5*uc;
                uux=u5/u1;
                calc(1,1);
                xi5=xi;
                xj5=xj;
                xf5=xf;
                xg5=xg;
                uux=uc/u1;
                calc(1,1);
                xic=xi;
                xjc=xj;
                xfc=xf;
                xgc=xg;
                rr5=sqrt(xlam*(rho1/rho2)/(xic-xi0));
                r5=rr5*rb;

                uu=uux/uuc;
                cp=cp2*(p*uu+(double)1.0);
                xmw=xmw2/(xm*uu+(double)1.0);
                gam=(double)1.0/((double)1.0-rgas/(xmw*cp));
                t=t2*((double)1.0+a*uux-b*uux*uux)/(p*uux+(double)1.0);
                mach=uux*u1/sqrt(gam*rgas*t/xmw);
                mach5=mach;
                factor();
                z1=xlam*xlam*(rho5/rho2)*(xic-xi0);
                z2=xgc-(double).5*xg5;
                z3=xfc-(double).5*xf5;
                z4=xjc-xj5;
                z5=xic-xi5;
                z6=xic-xi0;
                ic=0;
                v[0]=(double)0.0;
                v[1]=uuc;
                v[2]=(double)0.0;
                v[3]=(double)0.0;
                solvde(v,w,del,.05,1,&ic);
                duc=v[2];
                delx=del/rb;
                uuc+=duc*delx;
                rro=rr5*sqrt(log((double).02)/-xlam);
                ro=rro*rb;
                mix();
            }
            if (x < xmax)
            {
               if (fabs(x-xmax) < 0.1)
        x=xmax;
        else
```

```
            x+=xmax/(double)60.0;
         }else
                {
            if (x < xmax*(double)1.2)
            {
               x+=tstep2;
                      del=tstep2;
            }else
            {
               x+=tstep3;
               del=tstep3;
            }
         }
        }
        close (fh);
 fclose(prt);
 }

double desub(v)
double v[];
{
        extern double z1,z2,z3,z4,z5,z6,rr5,xk,xfc;
        return (xk/rr5)*((v[1]*v[1]*z1)/(z6*((v[1]/(double)2.0)*z2-z3)
                        -xfc*((v[1]/(double)2.0)*z4-z5)));
}
void bound()
{
        double b0,b1,b2,b3,b4,b5,b6,b7,b8;
        double rb0,rb1;
        double bet;
/*
        Calculate reference nozl mach number
*/
        b0=gam1+(double)1.0;
        b1=gam1 - (double) 1.0;
        xmb=sqrt( ( (double) 2.0/b1) * (pow( p0/p2, b1/gam1) - (double) 1.0));
        b2=b0/( (double) 4.0*b1);
        b3=xmb*xmb - (double) 1.0;
        b4=xma1*xma1 - (double) 1.0;
        bet=sqrt(b1/b0);
        rb0= (double) 2.0+b1 * xmb*xmb;
        rb1= (double) 2.0+b1 * xma1*xma1;
        rb=sqrt(xma1/xmb)*pow( rb0/rb1, b2) * r1;
        b5= (double) 1.0/bet;
        b8=bet*bet;
        b6=sqrt(b8*b3);
        b7=sqrt(b8*b4);
        omga=b5*(atan(b6)-atan(b7))-atan(b3)+atan(b4)+the1;
        aa=(rb-r1*( (double) 1.0/cos(omga)))/( (double) 1.0- (double) 1.0/cos(omga));
        xb=tan(omga)*(r1-aa);
}

void calc(opt1,opt2)
int opt1;                               /* opt = 0  initial conditions  */
```

>>>> plmrpt.c 06/24/88 01:08p page: 007

```
int opt2;
{
        double c1,c2;
        double e1,e2,e3;
        double g1,g2,g3;
        double alpha,beta,gamma,delta,epsn,zeta;
        double v1,v2,v3;
        c1 = b + a * xm - xm * xm;
        c2 = sqrt ( a * a + (double) 4.0 * b );
        e1 = ( b * ( p -xm )) / c1;
        e2 = xm * (( b + p * ( a - xm)) / c1 );
        e3 = ( xm * ( p - xm )) / c1;
        g1 = ( b * xm * (xm - p)) / c1;
        g2 = -g1;
        g3 = xm * (( xm * ( a - p ) + b) / c1 );
        alpha = -e1 / ( b * xm );
        beta = -e2 / ( (double) 2.0 * b * xm );
        gamma = ( e3 / xm + ( a * e2) / ((double) 2.0 * b * xm ))
                                * ((double)1.0/c2);
        delta = -g1/(b*xm);
        epsn = -g2/((double)2.0*b*xm);
        zeta = (g3/xm+(a*g2)/((double)2.0*b*xm))*((double)1.0/c2);
        v1 = xm*uux+(double)1.0;
        v2 = (double)1.0 + a*uux - b*uux*uux;
        v3 = (c2+(double)2.0*b*uux-a)/(c2-(double)2.0*b*uux+a);

        xi = alpha*log(v1)+beta*log(v2)+gamma*log(v3);
        xj = delta*log(v1)+epsn*log(v2)+zeta*log(v3);

        if (opt1 == 0)
                return;
        if (opt2 == 0)
        {
                xi5=xi;
                xj5=xj;
                rho5=rho2*(p*uux+(double)1.0)/((xm*uux+(double)1.0)
                        *((double)1.0+a*uux-b*uux*uux));
                psi=((rho1/rho5)/xlam)*(((double)1.0/(xi1-xi0))*((xj5-xj1)
                        -(double)2.0*(xi5-xi1))-(double)1.0);
                phi=xlam*(rho1/rho2)/(xi1-xi0);
                if (phi > (double)1.0) tau=sqrt(phi*(phi-(double)1.0));
                        else tau=sqrt(phi*((double)1.0-phi));
                return;
        }
        xf=(-e1)/(b*v1)+(e2*uux+e3)/(xm*v2);
        xg=(-g1)/(b*v1)+(g2*uux+g3)/(xm*v2);
        rho5=rho2*(p*uux+(double)1.0)/((xm*uux+(double)1.0)
                        *((double)1.0+a*uux-b*uux*uux));

}

void region1()
{
```

>>>> plmrpt.c 06/24/88 01:08p page: 008

```
        double xo;
        double dfo;
        double xn;
        double tol = (double)1.e-6;
        double tolx;
        double test;
        double d1,d2,d3,d4;
        int j;
/*
        This subroutine utilizes Newtons method to solve for the mach
        number in Region I at a specific location x less than b.
*/

        xo=xma1;
        d1=xma1/((r/r1)*(r/r1));
        d3=gam1-(double)1.0;
        d2=(double)2.0+d3*xma1*xma1;
        d4=(gam1+(double)1.0)/((double)4.0*d3);
        for (j=0;j<20;j++)
        {
                dfo=(double)2.*d4*d1*pow(((double)2.+d3*xo*xo)/d2,
                   (double)2.*d4-(double)1.)*(((double)2.*d3*xo)/d2)-(double)1.;
                xn=xo-(d1*pow(((double)2.+d3*xo*xo)/d2,(double)2.*d4)-xo)/dfo;
                test=fabs(xn-xo);
                tolx=tol*fabs(xn);
                xmai=xn;
                if (test < tolx)
            {
                        return;
        }
                xo=xn;
        }

}

void viscid()
{
        double xo;
        double dfo;
        double xx;
        double xn;
        double test;
        double tolx;
        double tol = (double)1.e-4;
        double w1,w2,w3,w4,w5,w6,w7,w8,w9;
        int j;
/*
        This subroutine calculates the core boundary using Newton's method.
*/
        xo=(double)1.0;
        xx=x/rb;
    if (phi > 1.0)
    {
                for(j=0;j<20;j++)
```

>>>> plmrpt.c 06/24/88 01:08p page: 009

```
                {
                        w1=((xk*xx)/psi)*(phi-(double)1.0);
                        w2=sqrt(phi-(phi-(double)1.0)*xo*xo);
                        w3=phi/((double)2.*tau);
                        w4=(phi+tau*xo)*(phi-tau);
                        w5=(phi-tau*xo)*(phi+tau);
                        w6=(double).5*pow(phi-(phi-(double)1.0)*xo*xo,(double)-0.5);
                        w7=(double)-2.0*(phi-(double)1.0)*xo;
                        w8=tau*(phi-tau);
                        w9=(-tau)*(phi+tau);
                        dfo=(-w6)*w7-(double)1.0+w3*(w8/w4-w9/w5);
                        xn=xo-((double)2.-w2-xo+w3*(log(w4)-log(w5))-w1)/dfo;
                        test=fabs(xn-xo);
                        tolx=tol*fabs(xn);
                        rri=xn;
                        if (test <= tolx)
                                return;
                        xo=xn;

                }
        }else
    {
        for(j=0;j<20;j++)
        {
            w1=((xk*xx)/psi)*((double)1.0-phi);
            w2=sqrt(phi+((double)1.0-phi)*xo*xo);
            w3=phi/tau;
            w4=atan(tau/phi);
            w5=atan((xo*tau)/phi);
            w6=(double)0.5*pow(phi+((double)1.0-phi)*xo*xo,(double)-0.5);
            w7=(double)2.0*((double)1.0-phi)*xo;
            w8=(double)1.0/((double)1.0+(xo*tau)/phi);
            dfo=w6*w7+(double)1.0-w8;
            xn=xo-(w2+xo-(double)2.0+w3*(w4-w5)-w1)/dfo;
            test=fabs(xn-xo);
            tolx=tol*fabs(xn);
            rri=xn;
            if (test <= tolx)
                return;
            xo=xn;
        }
        }


}

void mix()
{
        double temp;
        int i;
        int byte;
        double mac2;
        dat.dis=x;
```

>>>> plmrpt.c 06/24/88 01:08p page: 010

```
                if ((x > xb) && (x < xmax))
                {
                        temp=(ro-ri)/(double)10.;
                        dat.rad[0]=ri - temp;
                }else
                {
                        temp=ro/(double)10.;
                        dat.rad[0]=(double)0.0 - temp;

                }

                for (i=0;i<10;i++)
                {
                        if (i > 0) dat.rad[i] = dat.rad[i-1] + temp;
                                else dat.rad[0] += temp;
                if ((x > xb) && (x < xmax))
                {
                        rr=dat.rad[i]/rb;
                        uux=uuc*exp(-xlam*(rr*rr-rri*rri)/(rr5*rr5-rri*rri));
                }else
                {
                        rr=dat.rad[i]/rb;
                        uux=uuc*exp(-xlam*((rr*rr)/(rr5*rr5)));
                }
                uu=uux/uuc;
                cp=cp2*(p*uu+(double)1.0);
                xmw=xmw2/(xm*uu+(double)1.0);
                gam=(double)1.0/((double)1.0-rgas/(xmw*cp));
                t=t2*((double)1.0+a*uux-b*uux*uux)/(p*uux+(double)1.0);
                dat.mac[i]=uux*u1/sqrt(gam*rgas*t/xmw);
                mac2=dat.mac[i]*dat.mac[i];
            dat.tem[i]=t*((double)1.0+((gam-(double)1.0)/(double)2.0)*mac2);
        presu=p2*pow((double)1.0+((gam-(double)1.0)/(double)2.0)*
                        mac2,gam/(gam-(double)1.0));
                if (dat.mac[i] < (double)1.0)
                {
                        dat.pre[i]=presu;
                }else
                {
                        dat.pre[i]=presu*pow(((gam+(double)1.0)*mac2)/
                           ((gam-(double)1.0)*mac2+(double)2.0),gam/(gam-(double)1.0))
                                *pow((gam+(double)1.0)/((double)2.0*gam*mac2-gam+(double)1.0),
                                (double)1.0/(gam-(double)1.0));
                }
                dat.pre[i]-=p2;
                }
/*          if ((x < xmax ) || (x > 20.))   */

                if ((byte = write(fh,(char*)&dat,(unsigned int)sizeof(dat))) == -1)
                        perror("");
        if (dat.dis >= xprt)
                {
                fprintf(prt,"AXIAL DIST     ");
                fprintf(prt,"%10.4f\n",dat.dis);
```

```
                fprintf(prt,"RADIUS         ");
                for (i=0;i<10;i++)
                                fprintf(prt,"%10.4f",dat.rad[i]);
                                fprintf(prt,"\n");
                fprintf(prt,"MACH NO.       ");
                        for (i=0;i<10;i++)
                                fprintf(prt,"%10.4f",dat.mac[i]);
                                fprintf(prt,"\n");
                fprintf(prt,"TOTAL TEMP     ");
                        for (i=0;i<10;i++)
                                fprintf(prt,"%10.4f",dat.tem[i]);
                                fprintf(prt,"\n");
                fprintf(prt,"TOTAL PRES     ");
                        for (i=0;i<10;i++)
                                fprintf(prt,"%10.4f",dat.pre[i]);
                                fprintf(prt,"\n\n");
            xprt+ = cnt;
        }
}

void factor()
{
        int m;
        int n;
        double dep[5];
        double ind[5];

        dep[0]=(double)0.042;
        dep[1]=(double)0.042;
        dep[2]=(double)0.03;
        dep[3]=(double)0.025;
        dep[4]=(double)0.025;

        ind[0]=(double)0.0;
        ind[1]=(double)0.3;
        ind[2]=(double)0.73;
        ind[3]=(double)1.0;
        ind[4]=(double)100.0;

        if ((mach5 > ind[0]) && (mach5 <= ind[1]))
        {
                m=1; n=0;
        }
        if ((mach5 > ind[1]) && (mach5 <= ind[2]))
        {
                m=2; n=1;
        }
        if ((mach5 > ind[2]) && (mach5 <= ind[3]))
        {
                m=3; n=2;
        }
        if ((mach5 > ind[3]) && (mach5 <= ind[4]))
        {
                m=4; n=3;
```

```
        }

        xk=dep[n] + ((dep[m] - dep[n])/(ind[m] - ind[n]))*(mach5 - ind[n]);
}
```

>>>> plmrpt.c 06/24/88 01:08p page: 013

```
/*      This program solves a differential equation using          */
/*      using a Runge-Kutta solution.                              */
#include <stdio.h>
#include <math.h>


#define TRUE 1;
#define FALSE 0;
double desub();

void solvde(v,w,del,delmin,n,ic)
double v[];
double w[];         /* 3 component working array    */
double del;         /* Integration step size        */
double delmin;      /* Min value to use when decreasing step size   */
int n;              /* Order of equation to solve   */
int *ic;            /* Used for internal control    */
{
        int DBG = 0;
        int np1;
        int np2;
        int nh;
        int loop1;
        int loop2;
        int i,j;
        double tstop;
        double h;
        double h2;
        double vs[22];
        double c[4][20];
        double f;
        double tmp1;
        double tmp2;


/*       INITIALIZE     */

        np1 = n+1;
        np2 = n+2;

/*       COMPUTE STOP FOR THIS INTEGRATION STEP     */

        tstop = v[0]+del;
if (DBG)
printf("NP1 = %d NP2 = %d TSTOP = %lf\n",np1,np2,tstop);

/*       CHECK FOR FIRST TIME INTO ROUTINE      */
if (DBG)
printf("DEL = %lf W[1] = %lf  IC = %d\n",del,w[1],*ic);

        if (*ic <= 0)    /*      FIRST TIME IN       */
        {
/*       RESTART IF INPUT DEL HAS CHANGED       */
```

>>>> solvde.c 06/24/88 01:13p page: 001

```
                if (del != w[1])
                {
                        *ic = 0;
                        h = del;
                        nh = 0;
                        w[1] = del;
                } else
                {
                        h = w[0];
                        nh = (int)w[1];

/*      TEST FOR DOUBLING H. INCREASE TEST AS NUMBER OF HALVINGS
                INCREASES.                                              */

                        if (nh != 0)
                        {
                                if (*ic >= (10 * (nh + 1)))
                                {
                                        *ic += 1;
                                        h2 = 2.0 * h;
                                        if (modf((tstop + 0.01 * h - v[0]), &h2) < (0.011 * h))
                                        {
                                                h = h2;
                                                nh -= 1;
                                        }
                                }
                        }
                }

        }

/*  SET LOOP COUNTERS TO TRUE       */

        loop1 = TRUE;
        loop2 = TRUE;

/*  INCREASE COUNTER        */
        while (loop1)
        {
if (DBG)
{
 printf("DEL = %lf W[1] = %lf  IC = %d  H = %lf\n",del,w[1],*ic,h);
 printf("V[0] = %lf V[1] = %lf  V[2] = %lf  V[3] = %lf\n",v[0],v[1],v[2],v[3]);
}
                *ic += 1;                       /* label 60     */

/*      INTEGRATE USING R-K
                SAVE V TABLE VALUES IN VS ARRAY         */
                for (i=0;i<np2;i++)             /* label 100    */
                        vs[i] = v[i];

/*      FIRST PASS THRU R-K             */
                while (loop2)
                {
```

>>>> solvde.c 06/24/88 01:13p page: 002

```
                        for (j=0;j<n;j++)               /* label 110    */
                        {
                                if (j == (n-1))
                                {
                                        f = desub(v);
                                } else
                                {
                                        f = v[j+2];
                                }
                                c[0][j] = f * h;
if (DBG)
  printf("1  F = %lf C[0][%d] = %lf\n",f,j,c[0][j]);
                        }


/*      SECOND AND THIRD PASS THRU R-K          */

                        v[0] = vs[0] + 0.5 * h;
                        for (i=1;i<3;i++)                       /* do 129       */
                        {
                                for (j=1;j<np1;j++)                 /* do 122       */
                                {
                                        v[j] = vs[j] + 0.5 * c[i-1][j-1];
                                }
                                for (j=0;j<n;j++)                   /* do 127       */
                                {
                                        if (j == (n-1))
                                        {
                                                f = desub(v);
                                        } else
                                        {
                                                f = v[j+2];
                                        }
                                        c[i][j] = f * h;
if (DBG)
printf("2  F = %lf C[%d][%d] = %lf\n",f,i,j,c[i][j]);
                                }
if (DBG)
printf("V[0] = %lf V[1] = %lf  V[2] = %lf  V[3] = %lf\n",v[0],v[1],v[2],v[3]);
                        }
if (DBG)
{
printf("DEL = %lf W[1] = %lf  IC = %d  H = %lf\n",del,w[1],*ic,h);
printf("0.5 * H = %lf  DELMIN = %lf\n",0.5*h,delmin);
}
                        if ((0.5 * h) >= delmin)
                        {
/*      TEST FOR HALVING H      */

                                tmp2 = 0.0;
                                for (j=0;j<n;j++)               /* do 153       */
                                {
                                        tmp1 = c[0][j] - c[1][j];
                                        if (tmp1 != 0.0) tmp2 = (c[1][j] - c[2][j]) / tmp1;
```

```
if (DBG)
printf("TMP2 = %lf\n",tmp2);
                                        if (fabs(tmp2) > 0.025)
                                        {
                                                h = 0.5 * h;                    /* label 155
                                                nh += 1;
                                                *ic = 1;
                                                for (j=0;j<np2;j++)
                                                        v[j] = vs[j];
                                                break;
                                        } else
                                        {
                                                loop2 = FALSE;
                                        }
                                }
if (DBG)
printf("1/2 H  DEL = %lf W[1] = %lf  IC = %d  H = %lf\n",del,w[1],*ic,h);
                        } else
                        {
                                loop2 = FALSE;
                        }
                }  /* while (loop2)        */

/*        FOURTH PASS THRU R-K            */

                v[0] = vs[0] + h;                       /* label 160    */
                for (j=1;j<np1;j++)
                {
                        v[j] = vs[j] + c[2][j-1];       /* label 161    */
                }
if (DBG)
printf("4th pass V[0] = %lf V[1] = %lf  V[2] = %lf  V[3] = %lf\n",v[0],v[1],v[2],v[3]);
                for (j=0;j<n;j++)
                {
                        if (j == (n-1))
                        {
                                f = desub(v);
                        } else
                        {
                                f = v[j+2];
                        }
                        c[3][j] = f * h;
                }

/*       UPDATE V TABLE             */

                for (j=0;j<n;j++)                       /* label 180    */
                {
                        v[j+1] = vs[j+1] + (c[0][j] +2.0 * (c[1][j]+c[2][j]) + c[3][j])/6.0;
                }
if (DBG)
printf("UPDATE V[0] = %lf V[1] = %lf  V[2] = %lf  V[3] = %lf\n",v[0],v[1],v[2],v[3]);
/*      TEST FOR END OF INTEGRATION STEP        */
```

```
/* label 200    */
                if ((v[0] + 0.1*h) > tstop)
                {
                        loop1 = FALSE;
                } else
                {
                        if (*ic >= (10 * (nh + 1)))
                        {
                                *ic += 1;
                                h2 = 2.0 * h;
                                if (modf((tstop + 0.01 * h - v[0]), &h2) < (0.011 * h))
                                {
                                        h = h2;
                                        nh -= 1;
                                }
                        }
                        loop2 = TRUE;
                }
        }

/*       END OF REQUIRED INTEGRATION        */

        v[0] = tstop;                                   /* label 220    */
        v[np2 - 1] = desub(v);
        w[0] = h;
        w[1] = del;
        w[2] = (double)nh;

        return;
}
```

APPENDIX F

EXAMPLE PROGRAM PRINTOUT

PROPERTIES OF AMBIENT AIR

TEMP = 540.0000 PRES = 14.7000

ROCKET MOTOR PARAMETERS

COMBUSTION TEMP = 4500.0000 CHAMBER PRES = 1004.0000
SPECIFIC HEAT RATIO = 1.2300 MOLECULAR WEIGHT = 28.0900
RADIUS OF EXIT = .5000 NOZZLE HALF ANGLE = 0.2620
EXIT MACH NUMBER = 2.7800

MAXIMUM EXTENT OF CORE REGION = 149.7483

AXIAL DIST	102.328									
RADIUS	4.666	5.883	7.100	8.317	9.535	10.752	11.969	13.186	14.403	15.620
MACH NO.	3.234	2.535	1.965	1.507	1.142	0.851	0.618	0.434	0.290	0.183
TOTAL TEMP	4499.997	4031.254	3497.186	2933.653	2382.740	1884.600	1468.362	1146.530	915.468	760.668
TOTAL PRES	172.074	104.157	60.616	33.542	17.301	8.521	4.198	1.991	0.875	0.346
AXIAL DIST	200.250									
RADIUS	0.000	2.669	5.339	8.008	10.677	13.347	16.016	18.686	21.355	24.025
MACH NO.	2.347	2.249	1.990	1.648	1.293	0.969	0.692	0.467	0.292	0.166
TOTAL TEMP	3724.767	3647.135	3417.333	3049.904	2581.146	2071.737	1592.597	1199.945	917.533	737.549
TOTAL PRES	87.231	79.709	61.362	40.494	23.085	11.429	5.316	2.306	0.884	0.284
AXIAL DIST	300.250									
RADIUS	0.000	3.681	7.362	11.044	14.725	18.407	22.088	25.769	29.451	33.132
MACH NO.	1.617	1.560	1.406	1.191	0.953	0.723	0.518	0.346	0.212	0.118
TOTAL TEMP	2906.058	2844.216	2662.110	2374.272	2013.681	1630.873	1279.786	998.713	800.299	675.496
TOTAL PRES	37.906	34.973	27.516	18.505	10.775	5.752	2.823	1.236	0.463	0.143
AXIAL DIST	400.250									
RADIUS	0.000	4.745	9.491	14.237	18.983	23.729	28.475	33.221	37.967	42.713
MACH NO.	1.231	1.191	1.082	0.925	0.746	0.567	0.404	0.266	0.161	0.088
TOTAL TEMP	2377.585	2325.544	2173.988	1939.045	1651.627	1353.817	1086.588	876.391	729.902	638.535
TOTAL PRES	19.624	18.136	14.334	9.828	6.041	3.360	1.672	0.724	0.264	0.079
AXIAL DIST	500.250									
RADIUS	0.000	5.930	11.860	17.790	23.720	29.651	35.581	41.511	47.441	53.371
MACH NO.	0.980	0.950	0.866	0.743	0.600	0.455	0.321	0.209	0.124	0.067
TOTAL TEMP	1993.479	1949.517	1822.758	1629.594	1398.033	1162.822	955.334	794.256	683.023	614.050
TOTAL PRES	11.050	10.262	8.275	5.878	3.722	2.101	1.043	0.444	0.158	0.046
AXIAL DIST	600.250									
RADIUS	0.000	7.239	14.479	21.719	28.959	36.199	43.439	50.679	57.919	65.159
MACH NO.	0.804	0.780	0.712	0.611	0.492	0.371	0.260	0.167	0.098	0.052
TOTAL TEMP	1708.905	1671.755	1565.462	1405.626	1216.998	1028.281	863.918	737.541	650.842	597.302
TOTAL PRES	6.882	6.431	5.265	3.806	2.439	1.380	0.678	0.283	0.098	0.028

AXIAL DIST	700.250									
RADIUS	0.000	8.663	17.326	25.989	34.653	43.316	51.979	60.642	69.306	77.969
MACH NO.	0.676	0.655	0.598	0.513	0.412	0.309	0.214	0.137	0.079	0.042
TOTAL TEMP	1496.904	1465.362	1375.637	1242.050	1086.240	932.107	799.128	697.596	628.271	585.584
TOTAL PRES	4.634	4.344	3.583	2.609	1.677	0.945	0.459	0.188	0.064	0.0183
AXIAL DIST	800.250									
RADIUS	0.000	10.190	20.380	30.570	40.760	50.951	61.141	71.331	81.521	91.711
MACH NO.	0.578	0.561	0.511	0.438	0.350	0.261	0.180	0.114	0.065	0.034
TOTAL TEMP	1337.040	1310.050	1233.601	1120.626	990.009	861.886	752.122	668.749	612.020	577.164
TOTAL PRES	3.295	3.093	2.559	1.868	1.199	0.670	0.321	0.130	0.043	0.012
AXIAL DIST	900.250									
RADIUS	0.000	11.808	23.617	35.426	47.234	59.043	70.852	82.660	94.469	106.278
MACH NO.	0.503	0.488	0.444	0.379	0.302	0.224	0.153	0.096	0.055	0.029
TOTAL TEMP	1214.804	1191.486	1125.653	1028.910	917.802	809.513	717.230	647.410	600.025	570.956
TOTAL PRES	2.440	2.291	1.898	1.385	0.885	0.491	0.233	0.093	0.031	0.008

>>>> plume.prt 06/24/88 01:24p page: 002

APPENDIX G

PLOT PROGRAM LISTING

24 June 1988 - 02:05 PM (Pages G-1 thru G-11)

24 June 1988 - 01:07 PM (Pages G-12 thru G-18)

```
#include <gl.h>
#include <device.h>
#include <stdio.h>
#include <fcntl.h>
#include <math.h>
#include <time.h>

int xmin = 100;
int ymin = 100;
int xmax = 900;
int ymax = 600;
int choice;
long stat;
short val;
double maxv;
double minv;
double maxh;
double minh;
double tmp;
double incv,inch;
int DEBUG = FALSE;              /* Change to TRUE for DEBUG printout          */
struct _dat
{
    double dis;
    double rad[11];
    double tem[11];
    double mac[11];
    double pre[11];
};
struct _dat dat;
struct _dat in;

FILE *plt;
FILE *prt;

main()
{
    int fh;
    int ch;
    int i,j;
    int inc;
    int rbyte;
    double maxval = (double)-10000000.0;
    double minval = (double)10000000.0;
    double xmax = (double)-10000000.0;
    double xmin = (double)10000000.0;
    double ymax = (double)-10000000.0;
    double ymin = (double)10000000.0;
    double nval;
    double incr;
    int num;
    char tmp[80];
    char st[3];
    double x,r,z[40];
```

>>>> plumeplt.c 06/24/88 02:05p page: 001

```
    void plplt();
    ginit();
    cursoff();
    qdevice(MOUSE1);
    qdevice(KEYBD);
/*              */
    color(BLUE);
    clear();
    color(YELLOW);

    if ((fh=open("PLUME.DAT",(O_CREAT|O_RDWR),(0666))) == -1)
    {
        printf("COULD NOT OPEN DATA FILE");
        exit(1);
    }
    if ((plt = fopen("PLUME.PLT","wt")) == NULL)
    {
        printf("\n\nCOULD NOT OPEN PRINT FILE\n\n");
        exit(1);
    }
    if (DEBUG)
    {
        prt = fopen("PLUMEPLT.PRT","wt");
    }
    choice = 0;
    while ((choice != 'T') && (choice != 'P') && (choice != 'M'))
    {
        cmov2i(100,740);
        charstr("Enter Item to plot < T)emp, P)ress, M)ach > ");
        stat = qread(&val);
        tmp[0] = val;
        tmp[1] = '\0';
        charstr(tmp);
        choice = toupper(val);
    }
    cmov2i(100,720);
    charstr("Choice = ");
    charstr(tmp);
    rbyte = read(fh,(char*)&in,sizeof(in));
    while ((rbyte = read(fh,(char*)&dat,sizeof(dat))) == sizeof(dat))
    {
        switch(choice)
        {
        case 'T' :  for (i=0;i<10;i++)
                    {
                       if (dat.tem[i] > maxval) maxval = dat.tem[i];
                       if (dat.tem[i] < minval) minval = dat.tem[i];
                       if (dat.rad[i] > ymax) ymax = dat.rad[i];
                       if (dat.rad[i] < ymin) ymin = dat.rad[i];
                    }
                    break;
        case 'P' :  for (i=0;i<10;i++)
                    {
                       if (dat.pre[i] > maxval) maxval = dat.pre[i];
```

```
                    if (dat.pre[i] < minval) minval = dat.pre[i];
                    if (dat.rad[i] > ymax) ymax = dat.rad[i];
                    if (dat.rad[i] < ymin) ymin = dat.rad[i];
                }
                break;
        case 'M' : for (i=0;i<10;i++)
                {
                    if (dat.mac[i] > maxval) maxval = dat.mac[i];
                    if (dat.mac[i] < minval) minval = dat.mac[i];
                    if (dat.rad[i] > ymax) ymax = dat.rad[i];
                    if (dat.rad[i] < ymin) ymin = dat.rad[i];
                }
                break;
        }
        if (dat.dis > xmax) xmax = dat.dis;
        if (dat.dis < xmin) xmin = dat.dis;

    }
    switch(choice)
    {
        case 'T' : maxval = (double)((long)maxval);
                   minval = (double)((long)minval);
                   break;
        case 'P' : maxval = ((double)(long)(maxval*10)) /10.0;
                   minval = ((double)(long)(minval*10)) /10.0;
                   break;
        case 'M' : maxval = ((double)(long)(maxval*100)) /100.0;
                   minval = ((double)(long)(minval*100)) /100.0;
                   break;
    }
    cmov2i(100,700);
    charstr("Min Value = ");
    sprintf(tmp,"%12.4f",minval);
    charstr(tmp);
    cmov2i(100,680);
    charstr("Max Value = ");
    sprintf(tmp,"%12.4f",maxval);
    charstr(tmp);
    inc = 10;
    incr = (maxval - minval);
    num = 0;
    for (i=0;i<10;i++)
        z[i] = maxval - incr*log10((double)(i+1));
    while (num > -1)
    {
        color(BLUE);
        rectfi(100,100,1000,680);
        color(YELLOW);
        for (i=0;i<10;i++)
        {
            cmov2i(100,660-20*i);
            charstr("Inc ");
            sprintf(tmp,"%d",i);
            charstr(tmp);
```

```
        sprintf(tmp,"%12.2f",z[i]);
        charstr(tmp);
    }

    cmov2i(100,400);
    charstr("Enter # to change or  '-'  to exit -> _____");
    cmov2i(450,400);
    stat = qread(&val);
    num = val - 48;
    sprintf(tmp,"%d",num);
    tmp[1] = '\0';
    charstr(tmp);

    if (val > 47)
    {
        cmov2i(100,380);
        charstr("Enter VALUE for #");
        charstr(tmp);
        charstr("_____ ");
        cmov2i(280,380);
        i = 0;
        do
        {
            stat = qread(&val);
            if (val != 13)
            {
                tmp[i] = val;
                i++;
                sprintf(st,"%c",val);
                st[1] = '\0';
                charstr(st);
            }
        } while (val != 13);
        tmp[i] = '\0';
        z[num] = atof(tmp);
        if (DEBUG)
        {
            fprintf(prt,"tmp = %s  num = %d  z[num] = %10.2f\n",tmp,num,z[num]);
        }
    }
}

color(BLUE);
clear();

minval = z[9];
maxval = z[0];
fprintf(plt,"%10.2f%10.2f\n",ymin,ymax);
fprintf(plt,"%10.2f%10.2f\n",xmin,xmax);
fprintf(plt,"%10.2f%10.2f%10.2f\n",-1.0,-1.0,-1.0);
for (j=0;j<inc;j++)
{
    lseek(fh,(long)0,0);
    val = FALSE;
```

```
        while ((rbyte = read(fh,(char*)&dat,sizeof(dat))) == sizeof(dat))
        {
            switch(choice)
            {
            case 'T' :  for (i=0;i<9;i++)
                        {
                            if (((z[j] >= dat.tem[i]) &&
                                (z[j] <= dat.tem[i+1])) ||
                                ((z[j] <= dat.tem[i]) &&
                                (z[j] >= dat.tem[i+1])))
                            {
                                r = dat.rad[i] +
                                    ((z[j] - dat.tem[i])/(dat.tem[i+1] -
                                    dat.tem[i]))* (dat.rad[i+1] - dat.rad[i]);
                                fprintf(plt,"%10.2f%10.2f%10.2f\n",r,dat.dis,z[j]);
                                val = TRUE;
                                break;
                            }
                        }
                        break;
            case 'P' :  for (i=0;i<9;i++)
                        {
                            if (((z[j] >= dat.pre[i]) &&
                                (z[j] <= dat.pre[i+1])) ||
                                ((z[j] <= dat.pre[i]) &&
                                (z[j] >= dat.pre[i+1])))
                            {
                                 r = dat.rad[i] +
                                     ((z[j] - dat.pre[i])/(dat.pre[i+1] -
                                     dat.pre[i]))*(dat.rad[i+1] - dat.rad[i]);
                                 fprintf(plt,"%10.2f%10.2f%10.2f\n",
                                                                    r,dat.dis,z[j]);
                                 val = TRUE;
                                 break;
                            }
                        }
                        break;
            case 'M' :  for (i=0;i<9;i++)
                        {
                            if (((z[j] >= dat.mac[i]) &&
                                (z[j] <= dat.mac[i+1])) ||
                                ((z[j] <= dat.mac[i]) &&
                                (z[j] >= dat.mac[i+1])))
                            {
                                r = dat.rad[i] +
                                    ((z[j] - dat.mac[i])/(dat.mac[i+1] -
                                    dat.mac[i]))*(dat.rad[i+1] - dat.rad[i]);
                                fprintf(plt,"%10.2f%10.2f%10.2f\n",
                                                        r,dat.dis,z[j]);
                                val = TRUE;
                                break;
                            }
                        }
                        break;
```

```
            }
        }       /* while */
        if (val) fprintf(plt,"%10.2f%10.2f%10.2f\n",

    } /* for (j       */
    fclose(plt);
    close(fh);
    plplt();
    stat = qread(&val);
    fclose(plt);
    if (DEBUG)
    {
        close(prt);
    }
    color(BLACK);
    clear();
    curson();
    gexit();
    exit(0);
}


void plplt()
{
    int i,j;
    int inc;
    int rbyte;
    int val;
    int itmp;
    double rtmp;
    double incr;
    double x,r,z;
    double asave,bsave,zsave;
    int a,b,c;
    char str[20];
    float loc[3];
    float wid,hgt,rot;
    struct tm *tm;
    long clock;
    time(&clock);
    tm = localtime(&clock);

    if ((plt = fopen("PLUME.PLT","r")) == NULL)
    {
        printf("\n\nCOULD NOT OPEN PRINT FILE\n\n");
        exit(1);
    }
    rbyte = fscanf(plt,"%f%f\n",&minv,&maxv);
    if (DEBUG)
    {
        fprintf(prt,"Rbyte = %d Min V = %10.2f Max V = %10.2f\n",rbyte,minv,maxv);
    }

    rbyte = fscanf(plt,"%f%f\n",&minh,&maxh);
```

```
    if (DEBUG)
    {
        fprintf(prt,"Rbyte = %d Min H = %10.2f Max H = %10.2f\n",rbyte,minh,maxh);
    }

/*    axis      */

    rtmp = maxv - minv;
    z = (double)((int)log10((double)rtmp));
    incv = pow(10.0,z);

    if (DEBUG)
    {
        fprintf(prt,"rtmp %12.4f incv %12.4f\n",rtmp,incv);
    }

    rtmp = maxh - minh;
    z = (double)((int)log10((double)rtmp));
    inch = pow(10.0,z);

    if (DEBUG)
    {
        fprintf(prt,"rtmp %12.4f inch %12.4f\n",rtmp,inch);
    }

    if ((minv - incv) < 0.0) minv = 0.0;
       else minv = (double) ((int)(minv - incv));
    maxv = ((double) ((int)((maxv + incv)/incv))) * incv;

    if ((minh - inch) < 0.0) minh = 0.0;
       else minh = (double) ((int)(minh - inch));
    maxh = ((double) ((int)((maxh + inch)/inch))) * inch;
    if (DEBUG)
    {
        fprintf(prt,"minv %10.2f maxv %10.2f minh %10.2f maxh %10.2f\n",minv,maxv,minh,maxh);
    }
    color(BLACK);
    clear();
    color(WHITE);

/* Draw and label x-axis with grid      */

    move2i(xmin,ymin);
    draw2i(xmax,ymin);
    rtmp = minh;
    while (rtmp <= maxh+inch/2.0)
    {
        a = xmin + (int)(((double)(xmax - xmin)/(maxh - minh)) * (rtmp - minh));
        move2i(a,ymin);
        draw2i(a,ymax);
        if (DEBUG)
        {
            fprintf(prt,"a = %d ymin %d ymax %d  rtmp %10.2f\n",a,ymin,ymax,rtmp);
        }
```

>>>> plumeplt.c 06/24/88 02:05p page: 007

```
        sprintf(str,"%6.2f",rtmp);
        if (DEBUG)
        {
            fprintf(prt,"str %s\n",str);
        }
        wid = strwidth(str);
        hgt = getheight("M");
        loc[0] = (float)(a - wid/2.0);
        loc[1] = (float)(ymin - hgt - 5.0);
        loc[2] = 0.0;
        gennote(wid,hgt,loc,0.0,str);
        rtmp += inch;
    }

/* Draw and label y-axis with grid      */

    move2i(xmin,ymin);
    draw2i(xmin,ymax);
    rtmp = minv;
    while (rtmp <= maxv)
    {
        b = ymin - (int)(((double)(ymin - ymax)/(maxv - minv)) * (rtmp - minv));
        move2i(xmin,b);
        draw2i(xmax,b);
        sprintf(str,"%6.2f",rtmp);
        wid = strwidth(str);
        hgt = getheight("M");
        loc[0] = (float)(xmin - wid-10.0);
        loc[1] = (float)(b);
        loc[2] = 0.0;
        gennote(wid,hgt,loc,0.0,str);
        rtmp += incv;
    }

/* Annotate axis        */

    switch(choice)
    {
        case 'T' :  strcpy(str,"TOTAL TEMPERATURE");
                    break;
        case 'P' :  strcpy(str,"PITOT PRESSURE (PSIG)");
                    break;
        case 'M' :  strcpy(str,"MACH NO.");
                    break;
    }
    wid = strwidth(str);
    hgt = getheight("M");
    loc[0] = (float)((xmax + xmin)/2 - (int)(wid/2.));
    loc[1] = 730.0;
    loc[2] = 0.0;
    gennote(wid,hgt,loc,0.0,str);

/* Plot Title   */
```

```
    strcpy(str,"PLUME PROGRAM");
    wid = strwidth(str);
    hgt = getheight("M");
    loc[0] = (float)(xmax+xmin)/2 - wid/2.0;
    loc[1] = 750;
    loc[2] = 0.0;
    gennote(wid,hgt,loc,0.0,str);
    strcpy(str,"AXIAL DISTANCE (IN.)");
    wid = strwidth(str);
    hgt = getheight("M");
    loc[0] = (float)(xmax+xmin)/2 - wid/2.0;
    loc[1] = 50;
    loc[2] = 0.0;
    gennote(wid,hgt,loc,0.0,str);
    strcpy(str,"RADIAL DISTANCE (IN.)");
    wid = strwidth(str);
    hgt = getheight("M");
    loc[1] = (float)(ymax+ymin)/2 - wid/2.0;
    loc[0] = 20;
    loc[2] = 0.0;
    gennote(wid,hgt,loc,90.0,str);

/* Input Data   */

    hgt = getheight("M")*.8;
    strcpy(str,"ROCKET MOTOR PARAMETERS");
    wid = strwidth(str);
    loc[0] = (float)(xmax+xmin)/2 - wid/2.0;
    loc[1] = 690;
    loc[2] = 0.0;
    gennote(wid,hgt,loc,0.0,str);

    strcpy(str,"T0   Combustion Temp., Deg R :");
    wid = strwidth(str);
    loc[0] = (float)xmin + 50.0;
    loc[1] = 675;
    loc[2] = 0.0;
    gennote(wid,hgt,loc,0.0,str);
    sprintf(str,"%10.2f",in.rad[0]);
    wid = strwidth(str);
    gennote(wid,hgt,loc,0.0,str);

    strcpy(str,"P0   Chamber Pressure, psi   :");
    wid = strwidth(str);
    loc[0] = (float)xmin + 50.0;
    loc[1] = 660;
    loc[2] = 0.0;
    gennote(wid,hgt,loc,0.0,str);
    sprintf(str,"%10.2f",in.rad[1]);
    wid = strwidth(str);
    gennote(wid,hgt,loc,0.0,str);

    strcpy(str,"GAM1 Specific Heat Ratio     :");
    wid = strwidth(str);
```

>>>> plumeplt.c 06/24/88 02:05p page: 009

```
    loc[0] = (float)xmin + 50.0;
    loc[1] = 645;
    loc[2] = 0.0;
    gennote(wid,hgt,loc,0.0,str);
    sprintf(str,"%10.2f",in.rad[2]);
    wid = strwidth(str);
    gennote(wid,hgt,loc,0.0,str);

    strcpy(str,"XMW1 Molecular Weight        :");
    wid   strwidth(str);
    loc[0] = (float)xmin + 50.0;
    loc[1] = 630;
    loc[2] = 0.0;
    gennote(wid,hgt,loc,0.0,str);
    sprintf(str,"%10.2f",in.rad[3]);
    wid = strwidth(str);
    gennote(wid,hgt,loc,0.0,str);

    strcpy(str,"XMA1 Exit Mach Number  :");
    wid = strwidth(str);
    loc[0] = (float)(xmax + xmin)/2.0 + 60.0;
    loc[1] = 675;
    loc[2] = 0.0;
    gennote(wid,hgt,loc,0.0,str);
    sprintf(str,"%8.2f",in.rad[4]);
    wid = strwidth(str);
    gennote(wid,hgt,loc,0.0,str);

    strcpy(str,"R1   Exit Radius, in.  :");
    wid = strwidth(str);
    loc[0] = (float)(xmax + xmin)/2.0 + 60.0;
    loc[1] = 660;
    loc[2] = 0.0;
    gennote(wid,hgt,loc,0.0,str);
    sprintf(str,"%8.2f",in.rad[5]);
    wid = strwidth(str);
    gennote(wid,hgt,loc,0.0,str);

    strcpy(str,"THE1 Nozzle Half Angle :");
    wid = strwidth(str);
    loc[0] = (float)(xmax + xmin)/2.0 + 60.0;
    loc[1] = 645;
    loc[2] = 0.0;
    gennote(wid,hgt,loc,0.0,str);
    sprintf(str,"%8.2f",in.rad[6]);
    wid = strwidth(str);
    gennote(wid,hgt,loc,0.0,str);

    strcpy(str,"Date of Run            :");
    wid = strwidth(str);
    loc[0] = (float)(xmax + xmin)/2.0 + 60.0;
    loc[1] = 630;
    loc[2] = 0.0;
    gennote(wid,hgt,loc,0.0,str);
```

>>>> plumeplt.c 06/24/88 02:05p page: 010

```
    sprintf(str,"  %02d/%02d/%02d",tm->tm_mon+1,tm->tm_mday,tm->tm_year);
    wid = strwidth(str);
    gennote(wid,hgt,loc,0.0,str);

    move2i(xmin+10,710);
    draw2i(xmax-10,710);
    draw2i(xmax-10,620);
    draw2i(xmin+10,620);
    draw2i(xmin+10,710);

    zsave = -9999.99;
    while ((rbyte = fscanf(plt,"%f%f%f\n",&r,&x,&z)) != EOF)
    {
        a = xmin + (int)(((double)(xmax - xmin)/(maxh - minh)) * (x - minh));
        b = ymin - (int)(((double)(ymin - ymax)/(maxv - minv)) * (r - minv));

        if (x < 0.0)
        {
            if (zsave != -9999.99)
            {
                move2i(asave,bsave);
                sprintf(str,"%6.2f",zsave);
                wid = strwidth(str)*.8;
                hgt = getheight("a")*.8;
                loc[1] = bsave;
                loc[0] = asave;
                loc[2] = 0.0;
                gennote(wid,hgt,loc,0.0,str);
            }
            rbyte = fscanf(plt,"%f%f%f\n",&r,&x,&z);
            a = xmin + (int)(((double)(xmax - xmin)/(maxh - minh)) *
                        (x - minh));
            b = ymin - (int)(((double)(ymin - ymax)/(maxv - minv)) *
                        (r - minv));

            move2i(a,b);
        } else
        {
            draw2i(a,b);
            zsave = z;
            asave = a+10;
            bsave = b;
        }
    }       /* while  */
    return;
}
```

```
/* DOC
file:   gennote.c
functions:      gennote(), CharStr(), StrWidth(), Font(), GetHeight()
These functions provide a set of primitive vector text routines which are
used throughout wsmgr in place of different bitmapped fonts.  Vector text
has several advantages:
        1)      It is available (only three bitmapped fonts on the Sil Gr now)
        2)      It can be rotated, translated, etc
        3)      When ported, the text size can be scaled to the new resolution.
The routines can be slow.  In fact a profile of wsmgr shows that gennote
is one of the major hogs in the program.

In practice, you should never have to call gennote.  CharStr() and the other
guys provide a more natural method of handling text.  They were designed to
mirror the bitmapped function calls.  See charstr(), strwidth(), getheight()
as examples.  NOTE that you should carefully check the type and number of
arguments passed to these functions.  These functions do differ from their
bitmap counterparts in that respect.

StrWidth() and GetHeight return an estimate of the the number of pixels
required to print a string in the current font.

gennote() was written by Gregg Geis for other purposes.  The code was ported
to the Silicon Graphics and modified to get around the limitations we found
with bitmapped fonts.
END DOC */

#include <math.h>
#include <gl.h>
int letterwidth=10, letterheight=12, strokewidth=1;

/* Load starting index within vector array for each character. */
short   chstart[128] = { 977,  987,  1015, 1027, 1039, 1047, 1055, 1065,
                         1081, 1097, 1113, 1131, 1137, 1143, 1169, 1183,
                         1199, 1215, 1231, 1247, 1277, 1313, 1327, 1355,
                         1391, 1411, 1435, 1465, 1491, 1519, 1523, 1539,
                         0,    559,  583,  601,  623,  655,  683,  707,
                         719,  731,  743,  759,  769,  781,  785,  795,
                         367,  387,  397,  419,  445,  459,  477,  497,
                         507,  539,  799,  823,  849,  855,  865,  871,
                         901,  1,    11,   35,   53,   67,   81,   93,
                         115,  127,  139,  155,  167,  175,  185,  193,
                         211,  225,  249,  267,  293,  301,  315,  321,
                         331,  341,  351,  943,  951,  955,  963,  969,
                         971,  1561, 1587, 1607, 1625, 1645, 1665, 1683,
                         1709, 1723, 1745, 1769, 1781, 1791, 1815, 1829,
                         1847  1867, 1887, 1899, 1925, 1941, 1955, 1961,
                         1975, 1985, 2005, 0,    0,    0,    0,    0 };

/* Load terminating index for character vectors. */
short   chstop[128] =  { 986,  1014, 1026, 1038, 1046, 1054, 1064, 1080,
                         1096, 1112, 1130, 1136, 1142, 1168, 1182, 1198,
                         1214, 1230, 1246, 1276, 1312, 1326, 1354, 1390,
```

>>>> gennote.c 06/24/88 01:07p page: 001

```
                    1410, 1434, 1464, 1490, 1518, 1522, 1538, 1560,
                    0,    580,  600,  622,  654,  682,  706,  718,
                    730,  742,  758,  768,  780,  784,  794,  798,
                    386,  396,  418,  444,  458,  476,  496,  506,
                    538,  558,  820,  846,  854,  864,  870,  900,
                    942,  10,   34,   52,   66,   80,   92,   114,
                    126,  138,  154,  166,  174,  184,  192,  210,
                    224,  248,  266,  292,  300,  314,  320,  330,
                    340,  350,  366,  950,  954,  962,  968,  972,
                    976,  1586, 1606, 1624, 1644, 1664, 1682, 1708,
                    1722, 1744, 1768, 1780, 1790, 1814, 1828, 1846,
                    1866, 1886, 1898, 1924, 1940, 1954, 1960, 1974,
                    1984, 2004, 2012, 0,    0,    0,    0,    0 };

/* Load character vector data.
   The character set data defines relative vectors within a 14 by 22 grid listed
   as a sequence of (x,y) pairs. The normal base_line is at a y-value of 4.
   Desenders for the lower case alphabet can then extend down to a y-value of 0.
   The characters are printed in fixed width 16 by 22 spaces, so interletter
   spacing does not need to be included in the character vector data.
   An (x,y) with the values 50,50 will force a pen-up move to the succeeding
   location, thereby allowing characters with disjoint components; such as,
   semi-colons, lower case 'i', etc., to be generated. */
short  charset[4000] = {
/* A 1..10 */ 0,3,6,22,12,3,10,10,2,10,
/* B 11..34 */ 0,4,0,22,8,22,12,19,12,14,9,13,0,13,9,13,12,12,12,7,8,4,0,4,
/* C 35..52 */ 12,6,10,4,2,4,0,6,0.20,2,22,10,22,12,20,12,18,
/* D 53..66 */ 0,4,0,22,8,22,12,17,12,9,10,4,0,4,
/* E 67..80 */ 12,4,0,4,0,14,8,14,0,14,0,22,12,22,
/* F 81..92 */ 0,4,0,14,8,14,0,14,0,22,12,22,
/* G 93..114 */ 6,12,12,12,12,6,10,4,2,4,0,6,0,20,2,22,10,22,12,20,12,18,
/* H 115..126 */ 0,4,0,22,0,12,12,12,12,22,12,4,
/* I 127..138 */ 4,4,8,4,6,4,6,22,4,22,8,22,
/* J 139..154 */ 0,8,0,6,2,4,6,4,8,6,8,22,6,22,10,22,
/* K 155..166 */ 0,4,0,22,0,12,10,22,2,14,12,4,
/* L 167..174 */ 0,22,0,4,12,4,12,6,
/* M 175..184 */ 0,3,0,22,6,12,12,22,12,3,
/* N 185..192 */ 0,3,0,22,12,3,12,22,
/* O 193..210 */ 0,7,0,19,3,22,9,22,12,19,12,7,9,4,3,4,0,7,
/* P 211..224 */ 0,3,0,22,10,22,12,20,12,14,10,12,0,12,
/* Q 225..248 */ 0,7,0,19 3,22,9,22,12,19,12,7,9,4,12,1,6,8,9,4,3,4,0,7,
/* R 249..266 */ 0,3,0,22,10,22,12,20,12,14,10,12,0,12,6,12,12,3,
/* S 267..292 */ 0,6,2,4,10,4,12,6,12,10,10,12,4,14,0,16,0,20,2,22,10,22,12,20,12,18,
/* T 293..300 */ 6,3,6,22,0,22,12,22,
/* U 301..314 */ 0,22,0,6,2,4,10,4,12,6,12,4,12,22,
/* V 315..320 */ 0,22,6,4,12,22,
/* W 321..330 */ 0,22,3,4,6,17,9,4,12,22,
/* X 331..340 */ 0,4,12,22,6,13,12,4,0,22,
/* Y 341..350 */ 0,22,6,14,12,22,6,14,6,4,
/* Z 351..366 */ 0,22,12,22,6,13,4,13,8,13,6,13,0,4,12,4,
/* 0 367..386 */ 0,6,0,20,2,22,8,22,10,20,0,6,2,4,8,4,10,6,10,20,
/* 1 387..396 */ 4,4,8,4,6,4,6,22,4,18,
/* 2 397..418 */ 0,18,0,20,2,22,10,22,12,20,12,16,10,14,2,12,0,10,0,4,12,4,
/* 3 419..444 */ 0,20,2,22,10,22,12,20,12,16,8,14,2,14,8,14,12,10,12,6,10,4,2,4,0,6,
```

```
/* 4 445..458 */ 10,22,0,12,0,10,12,10,10,10,10,20,10,4,
/* 5 459..476 */ 0,8,2,4,10,4,12,6,12,12,10,14,0,14,0,22,12,22,
/* 6 477..496 */ 4,22,0,12,0,7,2,4,8,4,10,6,12,8,10,12,5,13,0,11,
/* 7 497..506 */ 0,20,1,22,12,22,3,4,3,4,
/* 8 507..538 */ 2,14,0,16,0,20,2,22,10,22,12,20,12,14,10,12,2,14,0,12,0,6,2,4,10,4,12,6,12,10,10,12,
/* 9 539..558 */ 12,14,10,12,2,12,0,14,0,20,2,22,10,22,12,20,12,13,8,4,
/*   no vector necessary for space character */
/* ! 559..580 */ 5,4,7,4,7,6,5,6,5,4,50,50,6,8,8,20,6,22,4,20,6,8,0,0,
/* " 581..600 */ 2,22,3,20,4,22,2,22,50,50,6,22,7,20,8,22,6,22,
/* # 601..622 */ 0,12,11,12,50,50,1,18,12,18,50,50,2,10,4,20,50,50,8,10,10,20,
/* $ 623..654 */ 0,6,2,4,10,4,12,6,12,10,10,12,2,12,0,14,0,18,2,20,6,20,6,22,6,2,6,20,10,20,12,18,
/* % 655..682 */ 0,22,4,22,4,18,0,18,0,22,50,50,8,4,12,4,12,8,8,8,8,4,50,50,0,4,12,22,
/* & 683..706 */ 12,6,2,18,2,20,4,22,6,22,8,20,0,12,0,6,2,4,8,4,10,6,12,10,
/* ' 707..718 */ 6,18,8,20,8,22,6,22,6,20,8,20,
/* ( 719..730 */ 6,22,4,20,2,16,2,10,4,6,6,4,
/* ) 731..742 */ 6,22,8,20,10,16,10,10,8,6,6,4,
/* * 743..758 */ 0,6,12,18,50,50,0,18,12,6,50,50,6,6,6,18,
/* + 759..768 */ 2,13,10,13,50,50,6,7,6,19,
/* , 769..780 */ 6,2,8,4,8,6,6,6,6,4,8,4,
/* - 781..784 */ 3,13,9,13,
/* . 785..794 */ 8,4,8,6,6,6,6,4,8,4,
/* / 795..798 */ 1,4,11,22,
/* : 799..820 */ 5,6,7,6,7,8,5,8,5,6,50,50,5,16,7,16,7,18,5,18,5,16,0,0,
/*; 821..846 */ 5,2,7,4,7,6,5,6,5,4,7,4,50,50,5,16,7,16,7,18,5,18,5,16,0,0,
/* < 847..854 */ 10,17,0,11,10,5,
/* = 855..864 */ 2,14,10,14,50,50,2,8,10,8,
/* > 865..870 */ 0,17,10,11,0,5,
/* ? 871..900 */ 2,18,2,20,4,22,10,22,12,20,12,16,10,14,6,12,6,8,50,50,5,4,7,4,7,6,5,6,5,4,
/* @ 901..942 */ 9,18,9,12,8,10,4,10,2,12,2,16,4,18,8,18,9,16,9,12,10,10,11,10,12,12,12,18,8,22,4,22,0,18,0,
/* [ 943..950 */ 8,22,4,22,4,4,8,4,
/* \ 951..954 */ 1,22,11,4,
/* ] 955..962 */ 8,22,12,22,12,4,8,4,
/* ^ 963..968 */ 2,16,6,22,10,16,
/* _ 969..972 */ 0,4,10,4,
/* ` 973..976 */ 4,22,8,16,
/* font 1001 symbols: */
/* 000 */ 3,2,15,20,3,20,15,2,3,2,
/* 001 */ 3,11,13,11,50,50,8,7,7,8,8,9,9,8,8,7,50,50,8,13,7,14,8,15,9,14,8,13,
/* 002 */ 3,6,15,6,50,50,15,8,3,13,15,18,
/* 003 */ 3,6,15,6,50,50,3,8,15,13,3,18,
/* 004 */ 4,3,14,3,9,14,4,3,
/* 005 */ 2,14,4,9,7,19,17,19,
/* 006 */ 6,8,12,14,50,50,6,14,12,8,
/* 007 */ 3,17,15,17,50,50,3,11,15,11,50,50,3,5,15,5,
/* 010 */ 2,14,14,14,50,50,2,8,14,8,50,50,4,4,12,19,
/* 011 */ 2,3,3,2,5,2,6,3,10,19,11,20,13,20,14,19,
/* 012 */ 4,5,12,5,14,6,15,8,16,11,15,14,14,16,12,17,4,17,
/* 013 */ 4,17,9,10,14,17,
/* 014 */ 4,10,9,17,14,10,
/* 015 */ 2,14,4,15,6,15,12,13.15,13,16,14,50,50,2,8,4,9,6,9,12,7,14,7,16,8,
/* 016 */ 14,4,14,3,3,3,7,11,3,18,14,18,14,17,
/* 017 */ 8,4,8,13,5,13,7,16,8,20,9,16,11,13,8,13,
/* 020 */ 8,20,8,11,11,11,9,7,8,4,7,7,5,11,8,11,
/* 021 */ 2,11,10,11,10,14,14,12,17,11,14,10,10,8,10,11,
```

>>>> gennote.c 06/24/88 01:07p page: 003

```
/* 022 */ 8,11,8,8,4,10,1,11,4,12,8,14,8,11,16,11,
/* 023 */ 8,2,11,20,50,50,9,17,7,15,6,12,6,9,7,6,10,5,12,6,14,9,14,12,13,15,11,17,9,17,
/* 024 */ 14,12,5,12,4,8,4,6,5,4,7,3,9,3,11,4,12,6,13,8,14,12,14,16,13,18,12,19,10,19,9,18,7,16,5,12,
/* 025 */ 8,3,8,18,50,50,2,17,3,18,13,18,14,17,
/* 026 */ 8,2,11,20,50,50,3,16,4,16,5,15,6,12,7,10,9,9,11,9,12,10,13,12,14,15,14,16,
/* 027 */ 6,15,4,12,3,9,3,6,4,4,6,3,8,3,9,4,10,7,10,10,9,11,8,5,10,3,12,3,15,5,16,9,16,13,15,15,
/* 030 */ 3,19,5,19,6,18,12,4,14,3,15,3,50,50,9,11,5,4,3,3,
/* 031 */ 14,14,13,13,6,13,4,11,3,8,4,5,8,4,11,4,13,7,13,10,12,12,9,13,
/* 032 */ 13,19,12,20,10,19,8,17,8,14,9,11,11,10,12,7,11,4,9,3,6,3,5,5,5,8,7,10,9,11,
/* 033 */ 3,2,5,5,7,15,50,50,5,5,6,4,10,3,12,4,13,6,15,15,50,50,12,4,12,3,
/* 034 */ 4,3,6,12,50,50,11,3,13,12,50,50,4,11,5,12,7,12,9,11,10,11,13,12,15,12,16,13,
/* 035 */ 3,20,16,20,
/* 036 */ 4,8,14,8,50,50,4,15,14,15,50,50,9,10,9,20,
/* 037 */ 9,18,10,16,11,15,13,15,14,16,15,18,14,20,13,21,11,21,10,20,9,18,
/* font 0 lower case alphabet characters */
/* a 1561..1586 */ 3,14,4,14,10,14,11,12,11,5,12,4,11,5,9,4,4,4,2,6,2,10,3,11,11,11,
/* b 1587..1606 */ 2,20,2,4,2,5,3,4,10,4,11,5,11,11,10,12,3,12,2,10,
/* c 1607..1624 */ 11,10,11,12,9,13,3,13,2,10,2,6,4,4,9,4,11,5,
/* d 1625..1644 */ 11,11,9,13,3,13,2,10,2,6,4,4,9,4,11,5,11,4,11,20,
/* e 1645..1664 */ 2,9,11,9,11,11,9,13,3,13,2,11,2,5,4,4,10,4,11,5,
/* f 1665..1682 */ 10,18,10,19,9,20,5,20,4,19,4,4,50,50,2,13,6,13,
/* g 1683..1708 */ 11,11,10,13,3,13,2,11,2,6,3,5,10,5,11,6,11,11,11,0,10,-1,3,-1,2,0,
/* h 1709..1722 */ 2,20,2,4,2,12,3,13,10,13,11,11,11,4,
/* i 1723..1744 */ 6,4,8,4,7,4,7,12,6,12,50,50,6,16,6,17,7,17,7,16,6,16,
/* j 1745..1768 */ 4,1,5,0,6,0,7,1,7,12,6,12,50,50,6,16,6,17,7,17,7,16,6,16,
/* k 1769..1780 */ 2,20,2,4,2,9,7,15,2,9,10,4,
/* l 1781..1790 */ 6,20,7,20,7,4,6,4,8,4,
/* m 1791..1814 */ 1,13,1,4,1,11,2,13,6,13,7,12,7,4,7,12,8,13,10,13,11,12,11,4,
/* n 1815..1828 */ 3,13,3,4,3,12,4,13,10,13,11,12,11,4,
/* o 1829..1846 */ 11,6,11,11,9,13,3,13,2,11,2,6,4,4,10,4,11,6,
/* p 1847..1866 */ 3,-2,3,13,3,11,4,13,10,13,11,11,11,6,10,4,4,4,3,6,
/* q 1867..1886 */ 11,-2,11,13,11,11,10,13,5,13,4,11,4,6,5,4,10,4,11,6,
/* r 1887..1898 */ 5,4,5,14,5,12,7,13,11,14,12,12,
/* s 1899..1924 */ 10,11,9,13,4,13,2,10,2,9,3,8,7,8,10,7,11,6,11,5,10,4,4,4,2,5,
/* t 1925..1940 */ 7,20,7,6,8,4,9,4,10,6,50,50,4,13,10,13,
/* u 1941..1954 */ 3,13,3,6,4,4,10,4,11,6,11,13,11,4,
/* v 1955..1960 */ 3,13,7,4,11,13,
/* w 1961..1974 */ 2,13,5,4,7,13,50,50,6,13,8,4,11,13,
/* x 1975..1984 */ 3,13,10,4,50,50,3,4,10,13,
/* y 1985..2004 */ 3,13,3,7,6,4,10,4,11,6,11,13,11,0,9,-1,6,-1,4,0,
/* z 2005..2112 */ 3,12,10,12,3,4,10,4
};

void gennote (Twidth, Theight, Tlocate, Trotate, Tstring)
float Twidth,  Theight, Tlocate[3], Trotate;
char  *Tstring;
{
  float    charwidth, H_unit, W_unit;
  Coord    x0, x1, y0, y1, xt, yt, *[4];
  register int    i, j;
  int      iskip, iptr, nchars, moveflg;

  nchars = strlen(Tstring);
  charwidth = Twidth / (float)(nchars);
```

>>>> gennote.c 06/24/88 01:07p page: 004

```
 H_unit = Theight / 22.0;
 W_unit = charwidth / 16.0;

 if ( fabs(Trotate) > 0.01 ) {
   /* Text is to be printed at an angle. */
   r[0] = cos(Trotate * M_PI / 180.0);
   r[1] = sin(Trotate * M_PI / 180.0);
   r[2] = Tlocate[0] * ( 1 - r[0] ) + Tlocate[1] * r[1];
   r[3] = Tlocate[1] * ( 1 - r[0] ) - Tlocate[0] * r[1];
 }
 for (i=0; i<nchars; i++) {
   moveflg = 0;
   iskip = 0;
   iptr = *(Tstring++);
   for (j=(chstart[iptr]-1); j<=(chstop[iptr]-3); j += 2) {
     if (iskip == 1) {
       iskip = 0;
       continue;
     }
     if (charset[j+2] > 47) {
       moveflg = 0;
       iskip = 1;
       continue;
     }
     x0 = Tlocate[0] + (float)( charset[j]   ) * W_unit;
     y0 = Tlocate[1] + (float)( charset[j+1] ) * H_unit;
     x1 = Tlocate[0] + (float)( charset[j+2] ) * W_unit;
     y1 = Tlocate[1] + (float)( charset[j+3] ) * H_unit;

     if ( fabs(Trotate) > 0.01 ) {
     /* Apply rotation/translation to text. */
       xt = x0 * r[0] - y0 * r[1] + r[2];
       yt = x0 * r[1] + y0 * r[0] + r[3];
       x0 = xt;
       y0 = yt;
       xt = x1 * r[0] - y1 * r[1] + r[2];
       yt = x1 * r[1] + y1 * r[0] + r[3];
       x1 = xt;
       y1 = yt;
     }
     if (!moveflg) move (x0, y0, 0.0);
     draw (x1, y1, 0.0);
     moveflg = 1;
   }
   Tlocate[0] += charwidth;
 }
}

fortran fcharst(i,TextStr, n, x, y, z, Rotation)
int i;
char *TextStr;
int *n;
float *x, *y, *z, *Rotation;
{
```

```
 extern int letterwidth, letterheight, strokewidth;
 float Twidth, Theight, location[3];
 char s[80];
 memcpy(s,TextStr,*n);
 s[*n] = '\0';
 location[0] = *x;
 location[1] = *y;
 location[2] = *z;
 Twidth = ((float)(letterwidth * strlen(s)));
 Theight = ((float)(letterheight));
 linewidth(strokewidth);
 gennote(Twidth, Theight, location, *Rotation, s);
 linewidth(1);
 return(0);
}

long StrWidth(TextStr)
char *TextStr;
{
 extern int letterwidth; /* letterwidth is set in Font(n) */
 return (long)((strlen(TextStr))*letterwidth);
}

fortran ffont(FontNumber, BoldOpt)
int *FontNumber;
char BoldOpt;
{
 extern int letterwidth, letterheight, strokewidth;
 int i;
 i = *FontNumber;
 strokewidth = 1;
 switch ( i ) {
   case 10: letterwidth = 10;
            letterheight = 13;
            break;
   case 12: letterwidth = 12;
            letterheight = 15;
            if ( BoldOpt == 'b' ) strokewidth = 2;
            break;
   case 14: letterwidth = 15;
            letterheight = 18;
            if ( BoldOpt == 'b' ) strokewidth = 2;
            break;
   case 16: letterwidth = 16;
            letterheight = 20;
            if ( BoldOpt == 'b' ) strokewidth = 2;
            break;
   case 24: letterwidth = 25;
            letterheight = 31;
            if ( BoldOpt == 'b' ) strokewidth = 3;
            break;
   case 32: letterwidth = 30;
            letterheight = 40;
            if ( BoldOpt == 'b' ) strokewidth = 3;
```

>>>> gennote.c 06/24/88 01:07p page: 006

```
            break;
    case 48: letterwidth = 50;
            letterheight = 63;
            if ( BoldOpt == 'b' ) strokewidth = 4;
            break;
    case 8:
    default:
            letterwidth = 7;
            letterheight = 10;
            break;
  }
  return(0);
}

long GetHeight()
{
  return (long)letterheight;
}
```

>>>> gennote.c 06/24/88 01:07p page: 007

DISTRIBUTION

	Copies
U. S. Army Materiel System Analysis Activity ATTN: AMXSY-MP Aberdeen Proving Ground, MD 21005	1
IIT Research Institute ATTN: GACIAC 10 W. 35th Street Chicago, IL 60616	1
AMSMI-RD, Dr. McCorkle	1
Dr. Rhoades	1
Dr. Stephens	1
-RD-ST, Dr. Mixon	1
Mr. Letson	1
-RD-ST-CM, Mr. Howard	1
-RD-ST-GD, Mr. Campbell	1
-RD-ST-SA, Mr. Christensen	1
Mr. Patrick	1
Mr. Reynolds	1
Mr. Schaeffel	1
Mr. Nourse	12
-RD-CS-R	15
-RD-CS-T	1
AMSMI-GC-IP, Mr. Fred Bush	1