General Disclaimer

One or more of the Following Statements may affect this Document

- This document has been reproduced from the best copy furnished by the organizational source. It is being released in the interest of making available as much information as possible.

- This document may contain data, which exceeds the sheet parameters. It was furnished in this condition by the organizational source and is the best copy available.

- This document may contain tone-on-tone or color graphs, charts and/or pictures, which have been reproduced in black and white.

- This document is paginated as submitted by the original source.

- Portions of this document are not fully legible due to the historical nature of some of the material. However, it is the best reproduction available from the original submission.

STIF

NASA CONTRACTOR REPORT 166416

NUMERICAL CALCULATION OF THE PARAMETERS OF THE EFFLUX FROM A HELIUM DEWAR USED FOR COOLING OF HEAT SHEILDS IN A SATELLITE

BY

Keith Brendley and John C. Chato
Department of Mechanical and Industrial Engineering
University of Illinois at Urbana-Champaign

July 1982

CONTRACT NAG2-- 144

NASA

NASA CONTRACTOR REPORT 166416

NUMERICAL CALCULATION OF THE PARAMETERS OF THE EFFLUX FROM A HELIUM DEWAR USED FOR COOLING OF HEAT SHIELDS IN A SATELLITE

BY

Keith Brendley and John C. Chato
Department of Mechanical and Industrial Engineering
University of Illinois at Urbana-Champaign

July 1982

National Aeronautics and
Space Administration

Ames Research Center
Moffett Field, California 94035

TABLE OF CONTENTS

ABSTRACT

The parameters of the efflux from a helium dewar in space were numerically calculated. The flow was modeled as a one-dimensional, compressible, ideal gas with variable properties. The primary boundary conditions are flow with friction and flow with heat transfer and friction. Two PASCAL programs were developed to calculate the efflux parameters: EFFLUXD and EFFLUXM. EFFLUXD calculates the minimum mass flow for the given shield temperatures and shield heat inputs. It then calculates the pipe lengths, diameter, and fluid parameters which satisfy all boundary conditions. Since the diameter returned by EFFLUXD is only rarely of nominal size, EFFLUXM calculates the mass flow and shield heat exchange for given pipe lengths, diameter, and shield temperatures.

NOMENCLATURE

A	area
B	function of M ($= [1 + (\gamma - 1)/2] M^2$)
C_p	specific heat
D	diameter
f	friction factor
h	fluid film constant
k_f	fluid thermal conductivity
L	pipe length
M	Mach number
m	mass flow
Nu	Nusselt number
P	pressure
P_o	stagnation pressure
Pr	Prandtl number
Q_w	shield heat exchange
q	heat flux
Re_D	Reynolds number
St	Stanton number
T	temperature
T_o	stagnation temperature
T_w	wall (shield) temperature
T_{aw}	adiabatic wall temperature
V	velocity
x	distance

Greek Letters

γ	specific heat ratio
ε	shield effectiveness
ρ	density
ρ_o	stagnation density
μ	viscosity

Subscripts

1	entrance
2	exit

1. INTRODUCTION

One of the primary objectives in the design of a long lifetime helium dewar is to minimize the parasitic heat load. This is accomplished to a large extent through low-conduction supports and vapor-cooled shields imbedded in multi-layered insulation (MLI). The vapor-cooled shields can be modeled as a series of heat exchangers with constant temperature walls and constant heat input to the helium efflux. With the ambient pressure of space being approximately zero, the fluid flow is compressible with a Mach number of unity at the exit throat. The solution of this model basically consists of determining (1) the length of pipe required to provide the necessary heat transfer and (2) the diameter required for the optimum mass flow to reach the Mach number of unity at the throat.

This solution entailed the development of two PASCAL programs: EFFLUXM and EFFLUXD. EFFLUXD is used to optimize mass flow and define the length and diameter of the heat exchanger pipe. Since this diameter is only rarely and by chance of nominal size, EFFLUXM is used to determine the mass flow and heat absorption of a similar heat exchanger but with a nominal diameter close to theoretical diameter determined through EFFLUXD. These programs are explained in APPENDIX A.

Although the original scope of the project was to develop these programs with the number of shields, the temperature of these shields and their position as given, a second-law thermal analysis of the shield system would seem beneficial to the project as a whole. Preliminary calculations indicate that the optimal number of shields is less than the five previously cited by other sources. Information on this subject will be forwarded at a later date.

2. THEORETICAL CONSIDERATIONS

The shield heat exchanger system is a bit unusual in that both shield temperature and heat input are specified at each shield from an overall heat transfer analysis of the entire insulation system developed elsewhere. If a shield heat exchanger system is analyzed with the effectiveness method, it becomes clear that for a given effectiveness the mass flow is a dependent variable. This mass flow may be minimized as an explicit function of shield temperature, heat input, and effectiveness. As an example of the method, the three-shield system shown in Fig. 1 will be analyzed. The initial stagnation temperature, the shield temperatures, the shield heat inputs, and the shield effectiveness (or shield-to-helium exit temperature difference) are all given. From the definition of effectiveness as shown in Eq. (1), the fluid stagnation temperatures at points 1, 2, and 3 may be calculated.

$$\varepsilon = (T_{o\ out} - T_{o\ in})/(T_w - T_{o\ in}). \qquad (1)$$

Note that the stagnation temperature at points 1 and 2 are outlet conditions for one shield and inlet conditions for the next. A single mass flow, however, will generally create different effectiveness in each shield. Consequently, the optimal mass flow must be found which will provide the maximum effectiveness in one shield. To accomplish this objective, each jth mass flow is calculated as a function of the initial stagnation temperature, the jth wall temperature, and the summation of heat inputs to the jth wall.

$$\dot{m}_j = \left[\sum_{i=1}^{j} Q_{wi}\right]/[C_p(T_{oj} - T_{o\ in})] \qquad (2)$$

The maximum resulting mass flow is the optimal mass flow for the given conditions. With this mass flow, the actual shield effectivenesses can be calculated. For constant fluid properties, these effectivenesses would

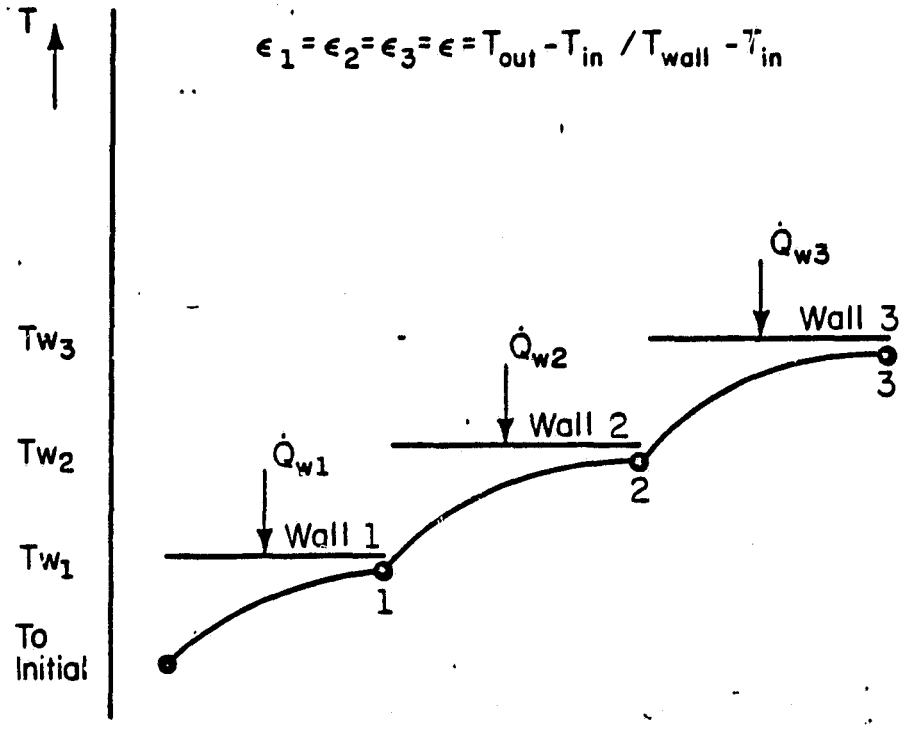


Figure 1 Temperature profile of helium pipe flow over three radiation shields.

directly define the heat exchanger lengths. The widely varying properties of the helium flow, however, force the sectional solution used in the EFFLUXD program.

In addition to the problem of variable fluid properties, the flow is compressible with a Mach number of 1 at the exit throat. Since the mass flow and length have already been set, the condition of M = 1 at the exit must be satisfied by determining the proper diameter. This is done implicitly in EFFLUXD as shown in APPENDIX A. If a nominal rather than calculated diameter is used, the mass flow is no longer optimized and must be varied until the exit condition of M = 1 is reached. This is done implicitly in EFFLUXM as shown in APPENDIX A.

Both of these programs use a number of functions and procedures which solve for the exit Mach number of a short section of pipe, given the inlet Mach number and the boundary conditions. The first of these procedures calculates the pipe entrance Mach number as a function of mass flow and diameter as explained in APPENDIX B.

For sections with friction only (connecting pipes between shields or valves), the flow is assumed laminar below a Reynolds number of 2000 and turbulent above that. This transition changes the frictional and thermal behavior of the fluid. In laminar flow, the friction factor is a function of Reynolds number only and the Nusselt number is constant; in turbulent flow, both the friction factor and Nusselt number are functions of Reynolds and Prandtl numbers as illustrated more clearly in APPENDIX C.

Frictional effects are also important in a heat transfer section of sufficient length. Before the exit Mach number of a short section may be calculated, the exit stagnation temperature must be known. This can be calculated with the simple energy balance in APPENDIX D. One interesting

aspect of this balance is that it is a function of the adiabatic wall temperature. For laminar flow, this is equivalent to the stagnation temperature; for turbulent flow, the adiabatic wall temperature lies between the fluid stagnation temperature and the actual fluid temperature. However, for subsonic pipe flow, the adiabatic wall temperature can be assumed to equal the stagnation temperature with no appreciable error.

After the exit stagnation temperature is calculated, the governing differential equation for compressible flow with heat transfer and friction may be simplified using Reynolds analogy and the previous energy balance as described in APPENDIX E. The resulting equation is solved as shown in APPENDIX F with a fourth-order Runge-Kutta method.

Within each small section, the fluid properties may be assumed constant, but overall thermal conductivity, viscosity, and Prandtl number vary as a function of temperature. The Prandtl number, as a function of absolute temperature, was modeled with a fifth-order least squares fit as shown in Fig. 2, while the thermal conductivity and viscosity are modeled with third-order fits as shown in Figs. 3 and 4. The temperatures used are the section inlet fluid temperatures.

The diameter, once determined, remains constant. Any constriction of the diameter in beginning sections increases the diameter in later sections to maintain the optimal mass flow. Furthermore, diameter increases would either require the design and manufacture of smooth venturi sections or would result in pressure loss across the area change. This added pressure loss would again require increasing the diameter even more or require increasing the mass flow. Earlier suggestions of diameter increases seemed to arise from the problem of extremely long pipes; heat transfer sections about two orders of magnitude longer than needed were proposed since, at the time, the heat

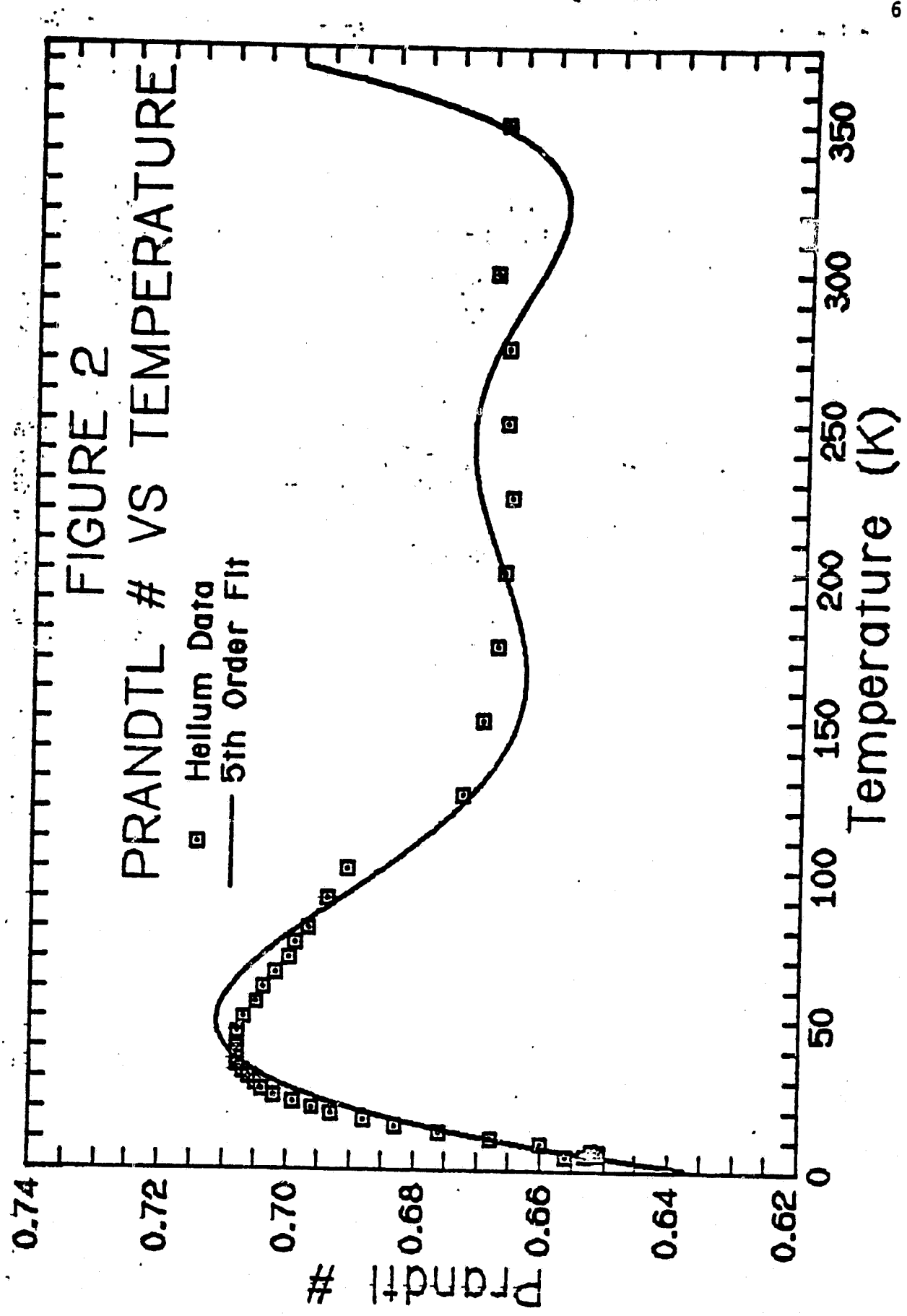


Figure 2 Prandtl Number versus Temperature

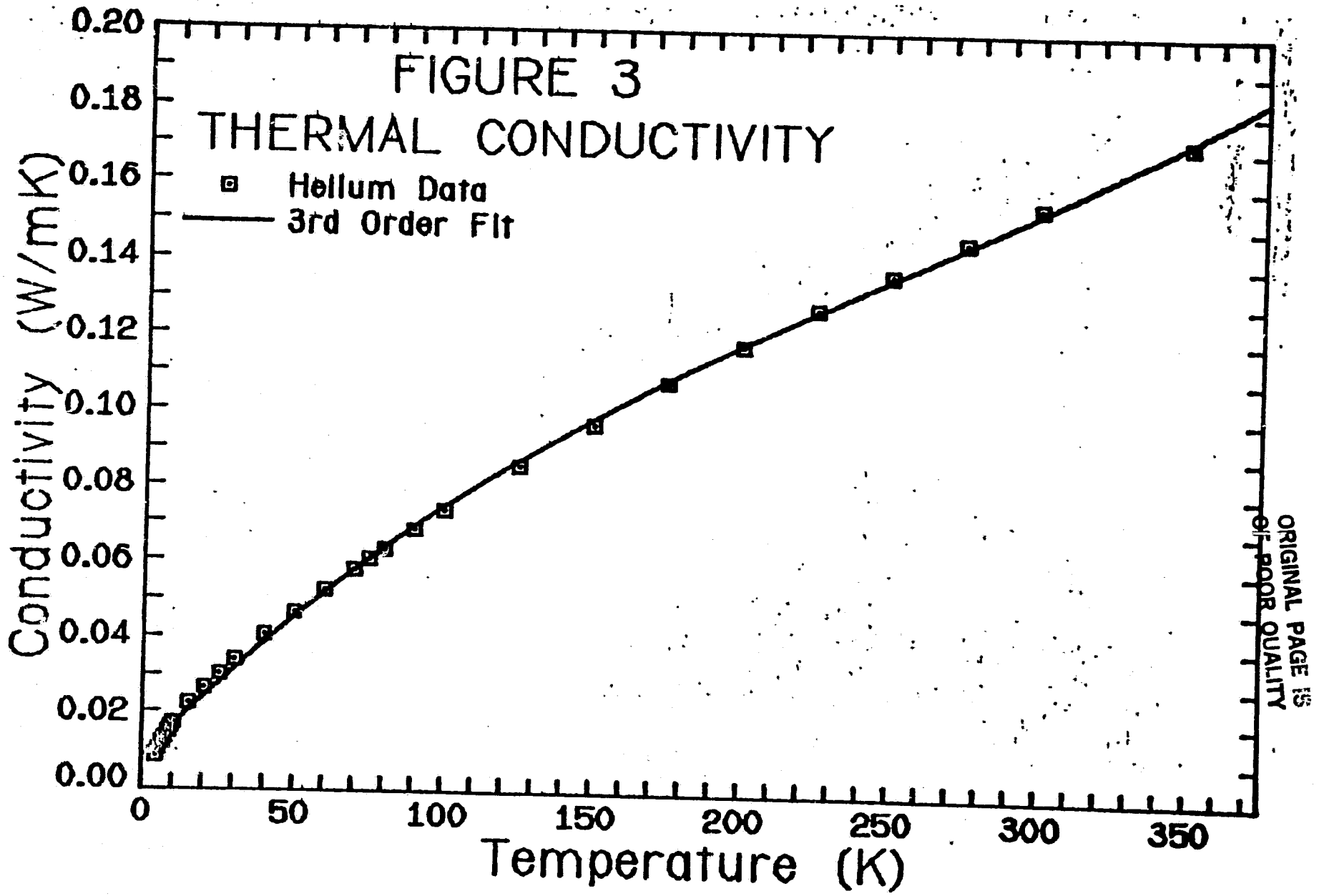


Figure 3 Thermal Conductivity

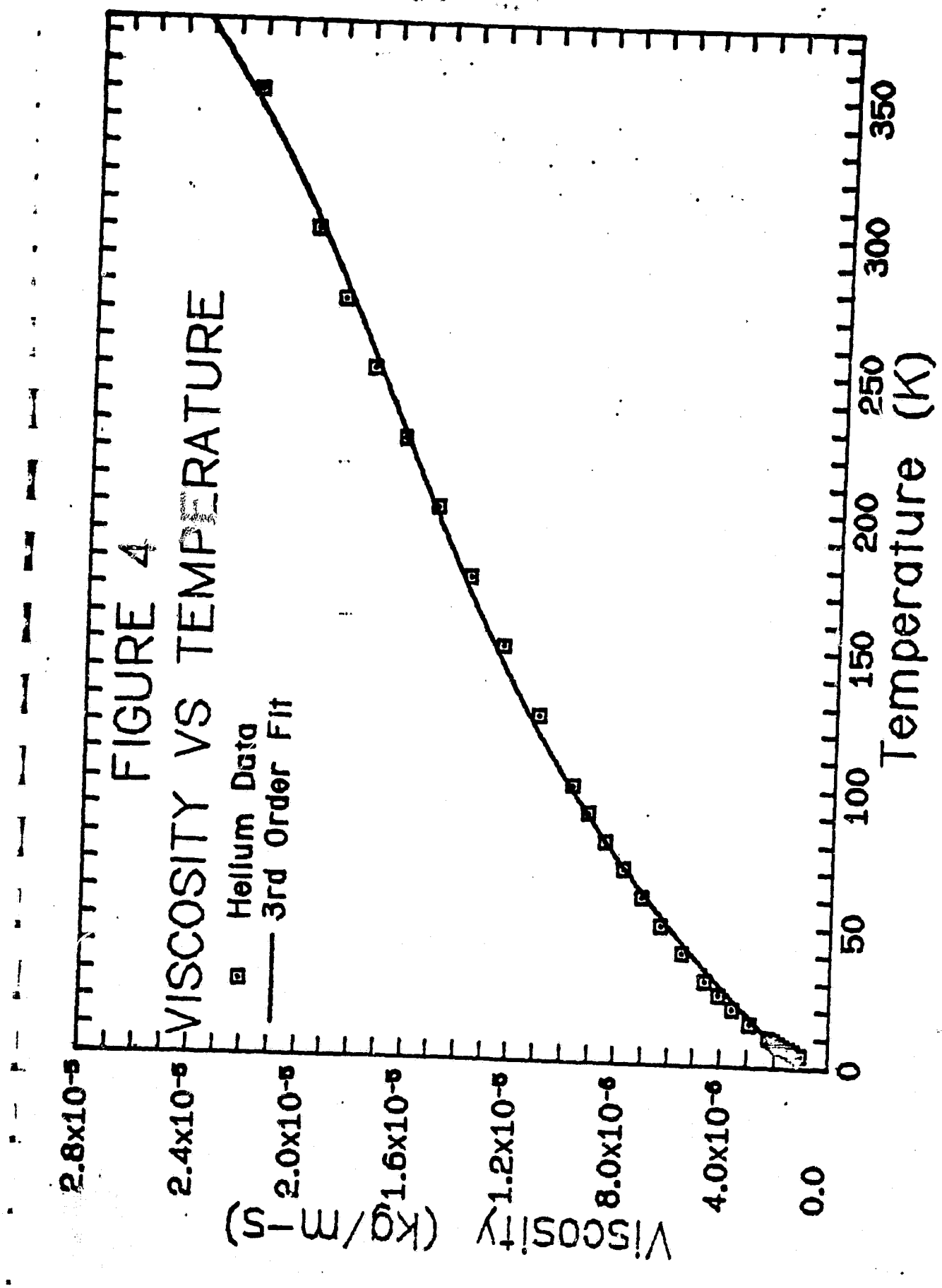


Figure 4 Viscosity versus Temperature

transfer mechanism was not fully understood. Even though it is difficult to see the advantage of area changes, a procedure to deal with them is in EFFLUXD as shown in APPENDIX H. This approach, however, was never utilized.

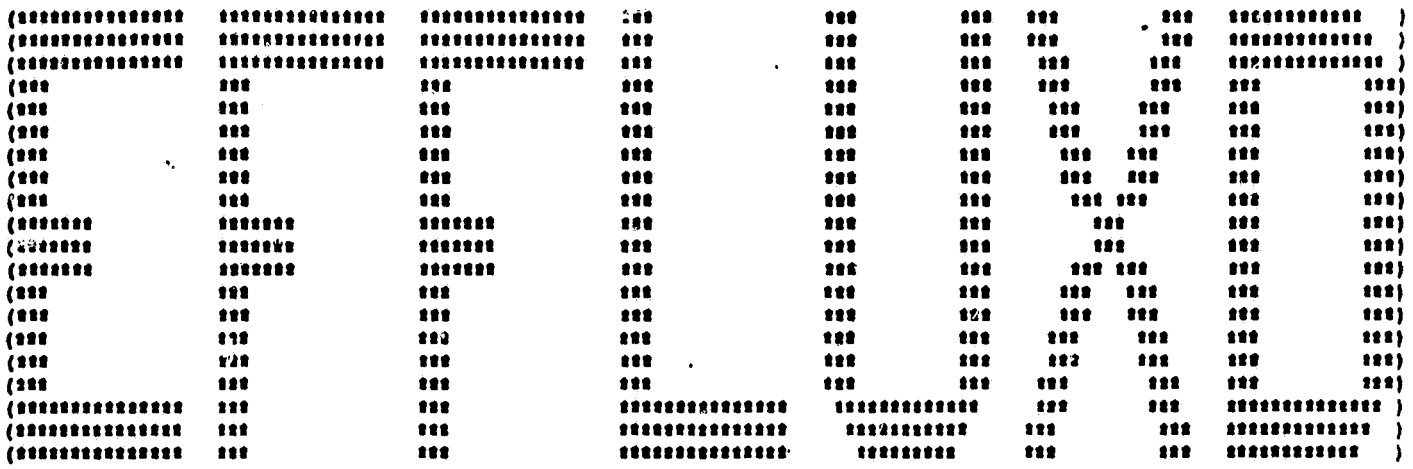
To summarize the design procedures, the following components can be identified:

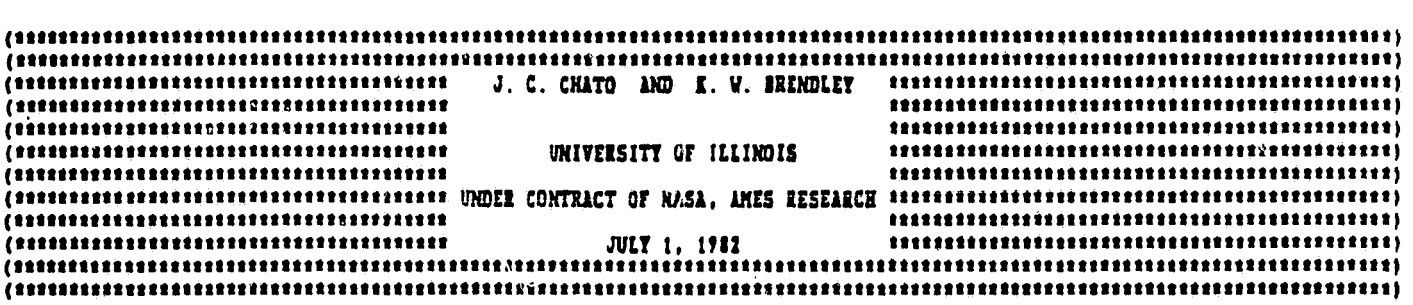
1. The sections in contact with heat shields are treated as heat exchangers with both heat transfer and friction influencing the flow of gas.
2. The connecting sections between heat shields are considered as adiabatic lengths with friction only.
3. Valves in the system can be treated as adiabatic sections as in point 2, with equivalent lengths adjusted to yield the proper pressure drop for a given flow rate and entrance conditions. Depending on the data available for the valve, this calculation will require iterations by the user of the program.

APPENDIX A

The inputs and outputs of the PASCAL programs EFFLUXD and EFFLUXM should be self-explanatory. The units are consistently SI. Example runs of both programs with a five-shield system follows. The input data used were taken from "Feasibility Study for Long Liftime Helium Dewar," Parmley, R. T., Lockheed Missiles and Space Company, Inc., Contract No. NAS2-10848, NASA, 1981, pp. 3-11.

```
 TYPE,EFFLUID
PROGRAM EFFLUID(INPUT/,OUTPUT,EFFOUTD,EFFIND1,EFFIND2,POUTD);
```






```
(*     THIS PROGRAM DETERMINES CERTAIN PARAMETERS OF THE EFFLUX RESULTING*)
(* FROM THE PARASITIC HEAT LOAD ON A HELIUM DEWAR IN SPACE.  THE SHIELD  *)
(* SYSTEM CONSISTS OF CONSTANT TEMPERATURE SHIELDS WITH CONSTANT HEAT    *)
(* INPUTS AS PREDETERMINED BY THE THERMAL CHARACTERISTICS OF THE         *)
(* SATELLITE.  THE MINIMUM MASS FLOW WHICH CAN SATISFY THESE INPUTS IS   *)
(* FIRST CALCULATED.  THIS MASS FLOW IMPLIES THE LENGTH OF THE SHIELD    *)
(* HEAT EXCHANGERS.  THE MASS FLOW, PIPELENTHS, AND THE NECESSARY        *)
(* CONDITION OF CHOKED FLOW AT THE EXIT ARE THEN USED TO CALCULATE THE   *)
(* PIPE DIAMETER.  THIS PROGRAM MAY BE USED IN CONJUNCTION WITH EFFLUIM  *)
(* WHICH SOLVES FOR MASS FLOW GIVEN PIPE LENGTHS AND A NOMINAL DIAMETER  *)


CONST
    PI=3.141592653;
    R=2077;(*J/KG-K*)
    K=1.67;
    CP=5208;

TYPE
    DIVISIONS=ARRAY[0..20000] OF REAL;
    COUNT=ARRAY[0..10] OF REAL;
    ICOUNT=ARRAY[0..20] OF INTEGER;
    REALOUT=FILE OF REAL;
    INTOUT=FILE OF INTEGER;

VAR
    NUMWALLS,NUMDIV,SPACECNT,N,NTEST:INTEGER;
    I,LOOPCOUNT,PKEY,TAB,PIPECNT,WALLCNTINTEGER;
    EFFOUTD,POUTD:TEXT;
    EFFIND1:REALOUT;
    EFFIND2:INTOUT;
    P,T,RHO,DIA:DIVISIONS;
    X,QX,M,P0,T0,Q,RE,F,NU:DIVISIONS;
    QW,TW,LTEMP,LSPACE,EFF,QWALL:COUNT;
```

```
      L,LTOTAL,LOWBOUND,HIBOUND,TOSAVE:REAL;
      MDOT,MAXEFF,DIAI,DIVLENGTH:REAL;
      MI,T01,P01,TI,PI,RHOI,FI,RE1,QI,QXI,MUI:REAL;


PROCEDURE INDATA(INDEX:INTEGER);
    BEGIN (* INTERACTIVE INPUT *)
    IF INDEX ) 0 THEN
        BEGIN
        (* READ IN DATA *)
        WRITELN('INPUT DATA AS  PROMPTED BY TERMINAL');
        WRITELN(' ');
        WRITELN('WHEN ASKED FOR A SERIES OF DATA SUCH AS THE SHIELD');
        WRITELN('TEMPERATURES, INPUT THE DATA IN THE ORDER FROM DEWAR');
        WRITELN('TO THE OUTER BOUNDARY.');
        WRITELN(' ');
        WRITELN('ENTER THE NUMBER OF RADIATION SHIELDS');
        READLN;
        READ(NUMWALLS);
        WRITE(EFFIND2,NUMWALLS);
        WRITELN('NUMBER OF SHIELDS=',NUMWALLS:5);
        WRITELN(' ');
        WRITELN('ENTER SINGLE DIVISION LENGTH (M)');
        READLN;
        READ(DIVLENGTH);
        WRITE(EFFIND1,DIVLENGTH);
        WRITELN('DIVISION LENGTH (M)=',DIVLENGTH:15:5);
        WRITELN(' ');
        WRITELN('ENTER PIPE LENGTHS NOT ON SHIELDS (M)');
        FOR I:=1 TO NUMWALLS+1 DO
            BEGIN
            READLN;
            READ(LSPACE[I]);
            WRITE(EFFIND1,LSPACE[I]);
            END;(*FOR*)
        WRITELN('ENTER TEMPERATURES OF RADIATION SHIELDS (K)');
        FOR I:=1 TO NUMWALLS DO
            BEGIN
            READLN;
            READ(TW[I]);
            WRITE(EFFIND1,TW[I]);
            END;(* FOR *)
        WRITELN('ENTER HEAT INPUTS OF RADIATION SHIELDS (W)');
        FOR I:=1 TO NUMWALLS DO
            BEGIN
            READLN;
            READ(QW[I]);
            WRITE(EFFIND1,QW[I]);
            END;(*FOR*)
        WRITELN('ENTER MAXIMUM HEAT EXCHANGER EFFECTIVENESS AS A FRACTION');
        READLN;
        READ(MAXEFF);
        WRITE(EFFIND1,MAXEFF);
        WRITELN('SHIELD EFFECTIVENESS=',MAXEFF:15:7);
        WRITELN(' ');
        WRITELN('ENTER INITIAL STAGNATION T (K)');
        READLN;
        READ(T0[1]);
        WRITE(EFFIND1,T0[1]);
        TW[0]:=T0[1];
        WRITELN('INITIAL STAGNATION TEMPERATURE (K)=',T0[1]:15:5);
        WRITELN(' ');
        WRITELN('ENTER INITIAL STAGNATION P (PA)');
        READLN;
        READ(P0[1]);
        WRITE(EFFIND1,P0[1]);
        WRITELN('INITIAL STAGNATION PRESSURE (PA)=',P0[1]:15:5);
        WRITELN(' ');
        WRITELN('ENTER NUMBER OF STEPS IN RUNGE-KUTTA METHOD');
        READLN;
        READ(N);
        WRITE(EFFIND2,N);
        WRITELN(' ');
        WRITELN('STEPS IN RUNGE-KUTTA=',N:5);
        END(*IF*)
    ELSE
        (* READ FILE *)
        BEGIN
        RESET(EFFIND1);
```

```
        RESET(EFFIND2);
        READ(EFFIND2,NUMWALLS);
        READ(EFFIND1,DIVLENGTH);
        FOR I:=1 TO NUMWALLS+1 DO READ(EFFIND1,LSPACE[I]);
        FOR I:=1 TO NUMWALLS DO READ(EFFIND1,TW[I]);
        FOR I:=1 TO NUMWALLS DO READ(EFFIND1,QW[I]);
        READ(EFFIND1,MAXEFF);
        READ(EFFIND1,T0[1]);
        TW[0]:=T0[1];
        READ(EFFIND1,P0[1]);
        READ(EFFIND2,N);
        END(*ELSE*)
    END;(* INDATA *)


PROCEDURE OUTDATA;
    BEGIN
    WRITELN(EFFOUTD,'LAMFLOW OUTPUT':60);
    WRITELN(EFFOUTD,' ');
    WRITELN(EFFOUTD,' ');
    WRITELN(EFFOUTD,' ');
    WRITELN(EFFOUTD,'VALUES OF INPUT':60);
    WRITELN(EFFOUTD,' ');
    WRITELN(EFFOUTD,' ');
    WRITELN(EFFOUTD,'INLET STAGNATION TEMPERATURE (K)=':50,T0[1]:15:10);
    WRITELN(EFFOUTD,'INLET STAGNATION PRESSURE (PA)=':50,P0[1]:15:10);
    WRITELN(EFFOUTD,'NUMBER OF RADIATION SHIELDS=':50,NUMWALLS:15);
    WRITELN(EFFOUTD,' ');
    WRITELN(EFFOUTD,'TEMPERATURES OF RADIATIATION SHIELDS':50);
    FOR I:=1 TO NUMWALLS DO WRITELN(EFFOUTD,'SHIELD TEMPERATURE (K)':30,I:2,'=':1,TW[I]:15:10);
    WRITELN(EFFOUTD,' ');
    WRITELN(EFFOUTD,'HEAT INPUTS TO RADIATION SHIELDS':50);
    FOR I:=1 TO NUMWALLS DO WRITELN(EFFOUTD,'SHIELD HEAT (W)':30,I:2,'=':1,QW[I]:15:10);
    WRITELN(EFFOUTD,' ');
    WRITELN(EFFOUTD,'PIPE LENGTHS BETWEEN SHIELDS':50);
    FOR I:=1 TO NUMWALLS+1 DO WRITELN(EFFOUTD,'SPACE LENGTH (M)':30,I:2,'=':1,LSPACE[I]:15:10);
    WRITELN(EFFOUTD,' ');
    WRITELN(EFFOUTD,'SOLUTION SECTION SIZE (M)=':50,DIVLENGTH:15:10);
    WRITELN(EFFOUTD,'MAXIMUM SHIELD EXCHANGER EFFECTIVENESS=':50,MAXEFF:15:10);
    WRITELN(EFFOUTD,' ');
    WRITELN(EFFOUTD,' ');
    WRITELN(EFFOUTD,' ');
    WRITELN(EFFOUTD,' ');
    WRITELN(EFFOUTD,' ');
    WRITELN(EFFOUTD,'VALUES OF OUTPUT':60);
    WRITELN(EFFOUTD,' ');
    WRITELN(EFFOUTD,' ');
    WRITELN(EFFOUTD,'MASS FLOW (KG/S)=':50,MDOT:20:10);
    WRITELN(EFFOUTD,'PIPE DIAMETER (M)=':50,DIA[LOOPCOUNT]:20:10);
    WRITELN(EFFOUTD,' ');
    WRITELN(EFFOUTD,'TOTAL PIPE LENGTH (M)=':50,LTOTAL:20:10);
    WRITELN(EFFOUTD,'SHIELD LENGTHS':40);
    FOR I:=1 TO NUMWALLS DO WRITELN(EFFOUTD,'SHIELD LENGTH (M)':30,I:2,'=':1,LTEMP[I]:15:10);
    WRITELN(EFFOUTD,' ');
    WRITELN(EFFOUTD,'TOTAL HEAT TRANSFER (W)=':50,QWALL[NUMWALLS+1]:20:10);
    WRITELN(EFFOUTD,'SHIELD HEAT TRANSFER':40);
    FOR I:=1 TO NUMWALLS DO WRITELN(EFFOUTD,'SHIELD HEAT (W)':30,I:2,'=':1,QWALL[I]:15:10);
    WRITELN(EFFOUTD,' ');
    WRITELN(EFFOUTD,'OVERALL EXCHANGER EFFECTIVENESS=':50,EFF[NUMWALLS+1]:20:10);
    WRITELN(EFFOUTD,'SHIELD EFFECTIVENESS':40);
    FOR I:=1 TO NUMWALLS DO WRITELN(EFFOUTD,'SHIELD EFF':30,I:2,'=':1,EFF[I]:15:10);
    WRITELN(EFFOUTD,' ');
    WRITELN(EFFOUTD,'EXIT STAGNATION TEMERATURE (K)=':50,T0[NUMDIV]:20:10);
    WRITELN(EFFOUTD,'EXIT STAGNATION PRESSURE (PA)=':50,P0[NUMDIV]:20:10);
    WRITELN(EFFOUTD,' ');
    WRITELN(EFFOUTD,' ');
    WRITELN('FOR AN OUTPUT OF SECTION FLOW VARIABLES, INPUT 1. OTHERWISE INPUT 0.');
    READLN;
    READ(PKEY);
    IF PKEY = 1 THEN
        BEGIN
        WRITE(EFFOUTD,'LENGTH':7,'MACH NUMBER':14,'STAG TEMP':14,'STAG PRES':14);
        WRITE(EFFOUTD,'TEMP':10,'PRESSURE':10,'NUSSELT':10,'REYNOLD NUM':15);
        WRITELN(EFFOUTD,'SECTION HEAT':15,'TOTAL HEAT':15);
        WRITELN(EFFOUTD,' ');
        FOR I:=1 TO NUMDIV DO
            BEGIN
            WRITE(EFFOUTD,X[I]:8:5,' ':4,M[I]:10:6,' ':4,T0[I]:10:6);
            WRITE(EFFOUTD,' ':3,P0[I]:10:5,' ':2,T[I]:8:4,' ':2,P[I]:8:3);
            WRITE(EFFOUTD,' ':2,NU[I]:8:3,' ':5,RE[I]:10:5,' ':5,Q[I]:10:7);
```

```
                WRITELN(EFFOUTD,' ':5,QX[I]:10:7);
                END(*FOR*)
            END;(*IF*)
        WRITELN('FOR A PLOT FILE ENTER 1, OTHERWISE ENTER 0');
        READLN;READ(TAB);
        IF TAB ) 0 THEN
            BEGIN
            WRITELN('FOR PLOT OF STAGNATION TEMPERATURE ENTER 1.');
            WRITELN('FOR PLOT OF STAGNATION PRESSURE ENTER 2.');
            WRITELN('FOR PLOT OF HEAT TRANSFER ENTER 3.');
            READLN;READ(TAB);
            WRITELN(FOUTD,NUMDIV);
            FOR I:=1 TO NUMDIV DO
                BEGIN
                CASE TAB OF
                    1:   WRITELN(FOUTD,T0[I]);
                    2:   WRITELN(FOUTD,P0[I]);
                    3:   WRITELN(FOUTD,QX[I]);
                    END(*CASE*)
                END;(*FOR*)
            FOR I:=1 TO NUMDIV DO WRITELN(FOUTD,X[I]);
            END;(*IF*)
        WRITELN('NUMBER OF INTERATIONS=':50,LOOPCOUNT:5);
        WRITELN('FOR INTERATION INFORMATION, INPUT1. OTHERWISE INPUT 0');
        READLN;
        READ(PKEY);
        IF PKEY = 1 THEN
            FOR I:=1 TO LOOPCOUNT DO
                WRITELN(EFFOUTD,'INTERATION=':20,I:3,'DIAMETER=':20,DIA[I]:15:10);
        END;(* OUTDATA *)

    FUNCTION PWR(N,P:REAL):REAL;
        (* SOLVES FOR N**P *)
        VAR A:REAL;
        BEGIN
        A:=P*LN(N);
        PWR:=EXP(A);
        END;(* PWR *)


    FUNCTION B(M:REAL):REAL;
        (* SOLVES OFTEN OCCURRING FUNCTION *)
        BEGIN
        B:=1.0+(K-1.0)*SQR(M)/2.0;
        END;(* B *)


    FUNCTION TEMP(T0,M:REAL):REAL;
        (* CALCULATES LOCAL TEMPERATURE *)
        BEGIN
        TEMP:=T0/B(M);
        END;(* TEMP *)


    FUNCTION PRES(P0,M:REAL):REAL;
        (* CALCULATES LOCAL PRESSURE *)
        VAR A,C:REAL;
        BEGIN
            C:=B(M);
            A:=K/(1-K);
            PRES:=P0*PWR(C,A);
        END;(* PRES *)



    FUNCTION DENSE(P,T:REAL):REAL;
        (* CALCULATES LOCAL DENSITY *)
        BEGIN
        DENSE:=P/(R*T);
        END;(* DENSE *)



    FUNCTION MU(T:REAL):REAL;
        VAR A,B,C,D:REAL;
        BEGIN(*MU*)
        A:=1.10419265531E-06;
        B:=1.16e-7754644E-07*T;
        C:=-3.22253279240A2E-10*SQR(T);
```

```
        D:=4.72570629%922E-13*T*SQR(T);
        MU:=A+B+C+D;
        END;(*MU*)


    FUNCTION COND(T:REAL):REAL;
        VAR A,B,C,D:REAL;
        BEGIN(*CONDUCTIVITY CALCULATION*)
        A:=0.0157218071;
        B:=4.3416525E-04*T;
        C:=-6.99407521E-07*SQR(T);
        D:=4.52803894E-10*T*SQR(T);
        COND:=A+B+C+D;
        END;(*COND*)


    FUNCTION PRANDTL(T:REAL):REAL;
        VAR A,B,C,D,E,F:REAL;
        BEGIN(*PRANDTL NUMBER CALCULATION*)
        A:=0.570232994;
        B:=0.805587229%*T;
        C:=-7.58324532E-05*SQR(T);
        D:=4.32720501E-07*T*SQR(T);
        E:=-1.11661445E-09*SQR(T)*SQR(T);
        F:=1.87403842E-12*T*SQR(T)*SQR(T);
        PRANDTL:=A+B+C+D+E+F;
        END;(*PRANDTL*)


    FUNCTION MOUT(M1,M2,D1,D2:REAL):REAL;
        (* USED IN CASE 1 OF NEWTON *)
        VAR A,C:REAL;
        BEGIN
        A:=(K+1.0)/(2.0*(K-1.0));
        C:=B(M2)/B(M1);
        MOUT:=(M1/M2)*PWR(C,A)-SQR(D2/D1);
        END;(* MOUT *)


    FUNCTION DMOUT(M1,M2:REAL):REAL;
        (* USED IN CASE 1 OF NEWTON *)
        VAR A,C:REAL;
        BEGIN
        A:=B(M2)/B(M1);
        C:=(K+1.0)/(2.0*(K-1.0));
        DMOUT:=(((K+1.0)/2.0)*(B(M1)/B(M2))-1.0/SQR(M2))*M1*PWR(A,C);
        END;(* DMOUT *)


    FUNCTION MFRIC(M1,M2,D,F,L:REAL):REAL;
        VAR MFR:REAL;
        (* USED IN CASE 2 OF NEWTON *)
        BEGIN
        MFR:=(SQR(M2)-SQR(M1))/(K*SQR(M1*M2))+(K+1.0)/(2.0*K)*LN(B(M2)*SQR(M1)/(B(M1)*SQR(M2)))-4.0*F*L/D;
        MFRIC:=MFR;
        END;(* MFRIC *)


    FUNCTION DMFRIC(M1,M2:REAL):REAL;
        (* USED IN CASE 2 OF NEWTON *)
        VAR DMF:REAL;
        BEGIN
        DMF:=2.0/(K*M2*SQR(M2))+(K+1)/K*M2*SQR(B(M1))/(PWR(M1,4)*PWR(B(M2),3));
        DMFRIC:=DMF;
        IF DMF < 1.274 THEN (*FUNCTION UNSTABLE, MACH=1 APPROACHED*)
            DMFRIC:=1.0;
        END;(* DMFRIC *)


    FUNCTION MFRST(M,D,MDOT,P0,T0:REAL):REAL;
        (* USED IN CASE 3 OF NEWTON *)
        VAR A,C:REAL;
        BEGIN
        A:=B(M);
        C:=(K+1.0)/(2.0*(1.0-K));
        MFRST:=(M*P0*PI*SQR(D)/4.0)*SQRT(K/(R*T0))*PWR(A,C)-MDOT;
```

```
    END;(* MFRST *)


FUNCTION DMFRST(M,D,P0,T0:REAL):REAL;
    (* USED IN CASE 3 OF NEWTON *)
    VAR A,C,E:REAL;
    BEGIN
    A:=R(M);
    C:=(K+1.0)/(2.0*(1.0-K));
    E:=(3.0*K-1.0)/(2.0*(1.0-K));
    DMFRST:=(PI*SQR(D)/4.0)*P0*SQRT(K/(R*T0))*(PWR(A,C)-SQR(M)*(K+1.0)/2.0*PWR(A,E));
    END(* DMFRST *);


FUNCTION NEWTON(KEY:INTEGER; M1,D,F,L,LTOT,P0,T0:REAL):REAL;
    VAR MNEW,MOLD,DIFF,D1,D2,A,MDOT,DMF,MF:REAL;
        COUNT:INTEGER;
    BEGIN(* NEWTON *)
    DIFF:=1.0;
    CASE KEY OF
        1: BEGIN
        (* CALCULATES EXIT MACH NUMBER FOR SECTION WITH AREA CHANGE *)
            D1:=D;
            D2:=F;
            MOLD:=SQR(D1/D2)*M1;
            WHILE (DIFF >= 0.001) DO
                BEGIN
                MNEW:=MOLD-MOUT(M1,MOLD,D1,D2)/(DMOUT(M1,MOLD));
                DIFF:=ABS(MOLD-MNEW);
                MOLD:=MNEW;
                END(* WHILE *)
            END;(* 1 *)
        2: BEGIN
        (* CALCULATES EXIT MACH FOR SECTION WITH FRICTION ONLY *)
            MOLD:=1.01*M1;
            COUNT:=1;
            WHILE (DIFF >= 0.00005) AND (COUNT < 10) DO
                BEGIN
                DMF:=DMFRIC(M1,MOLD);
                MF:=MFRIC(M1,MOLD,D,F,L);
                MNEW:=MOLD-MFRIC(M1,MOLD,D,F,L)/DMF;
                DIFF:=ABS(MOLD-MNEW);
                IF DMF = 1.0 THEN (*FUNCTION UNSTABLE, EXIT LOOP*)
                    BEGIN
                    DIFF:=0.00001;
                    MNEW:=1.0;
                    END;(*IF*)
                COUNT:=COUNT+1;
                MOLD:=MNEW;
                END(* WHILE *)
            END;(*CASE 2*)
        3: BEGIN
        (* CALCULATES INLET MACH NUMBER *)
            MDOT:=L;
            MOLD:=M1;
            WHILE (DIFF >= 0.00001) DO
                BEGIN
                MNEW:=MOLD-MFRST(MOLD,D,MDOT,P0,T0)/DMFRST(MOLD,D,P0,T0);
                DIFF:=ABS(MOLD-MNEW);
                MOLD:=MNEW;
                END(* WHILE *)
            END(* 3 *)
        END;(* CASE *)
    NEWTON:=MOLD;
    END;(* NEWTON *)



PROCEDURE INLETMACH(D,MDOT,P0,T0:REAL; VAR M,T,P,RHO:REAL);
    (* CALCULATES INLET CONDITIONS *)
    VAR A:REAL;
    BEGIN(* INLETMACH *)
        M:=NEWTON(3,M,D,0,MDOT,0.0,P0,T0);
        T:=TEMP(T0,M);
        P:=PRES(P0,M);
        RHO:=DENSE(P,T);
    END;(* INLETMACH *)
```

```
PROCEDURE AREACHANGE(D1,D2,T0,P0:REAL;VAR M,T,P,RHO,RE,F,NU,PR:REAL);
    (* FOR DIAMETER CHANGE *)
    VAR A:REAL;
    BEGIN
    M:=NEWTON(1,M,D1,D2,0,0,0,0);
    T:=TEMP(T0,M);
    P:=PRES(P0,M);
    RHO:=DENSE(P,T);
    RE:=RE*D1/D2;
    IF RE (= 2000 THEN
        BEGIN (* LAMINAR FLOW *)
        F:=16.0/RE;
        NU:=3.66;
        END(*IF*)
    ELSE
        BEGIN (* TURBULENT FLOW *)
        F:=0.046/PWR(RE,0.2);
        NU:=0.023*PWR(RE,0.8)*PWR(PR,0.4);
        END;(*ELSE*)
    END;(* AREACHANGE *)


PROCEDURE FRICTION(T0,DIA,F,L:REAL;VAR M,T,P,RHO,P0,Q:REAL);
    (* FOR FLOW WITH FRICTION ONLY *)
    VAR A,C,NEWM:REAL;
    BEGIN
    NEWM:=NEWTON(2,M,DIA,F,L,LTOTAL,0,0);
    IF NEWM ) 1.0 THEN NEWM:=1.0;
    A:=B(NEWM)/B(M);
    C:=(K+1.0)/(2.0*(K-1.0));
    P0:=P0*M/NEWM*PWR(A,C);
    M:=NEWM;
    T:=TEMP(T0,M);
    P:=PRES(P0,M);
    RHO:=DENSE(P,T);
    Q:=0;
END;(*FRICTION*)


FUNCTION DIAMETER(MDOT,T0,P0,M:REAL):REAL;
    (* CALCULATES DIAMETER GIVEN MDOT AND MACH *)
    VAR A,C:REAL;
    BEGIN(*DIAMETER*)
    A:=4.0*MDOT*SQRT(R*T0/K)/(PI*M*P0);
    C:=(K+1.0)/(2.0*(K-1.0));
    DIAMETER:=SQRT(A*PWR(B(M) C));
    END;(*DIAMETER*)


PROCEDURE SECTION(NUMTOT:INTEGER; LENGTH,DIVL:REAL; VAR X:DIVISIONS; VAR NUMDIV:INTEGER);
    (* DIVIDES PIPE LENGTHS INTO SMALLER SECTIONS *)
    VAR
        RESL,REALDIV:REAL;
        I,J:INTEGER;

    BEGIN(*SECTION*)
    IF NUMTOT = 1 THEN
        BEGIN
        NUMTOT:=0;
        X[0]:=0;
        END;(*IF*)
    (* CALCULATE RESIDUAL SOLUTION SECTION LENGTH *)
    REALDIV:=LENGTH/DIVL;
    NUMDIV:=TRUNC(REALDIV);
    RESL:=(REALDIV-NUMDIV)*DIVL;
    IF RESL ) 0 THEN
        BEGIN
        NUMDIV:=NUMDIV+1;
        END(*IF*)
    ELSE
        BEGIN
        RESL:=DIVL;
        END;(*ELSE*)
    (* DEFINE LENGTH SECTION ARRAY *)
    NUMDIV:=NUMTOT+NUMDIV;
    FOR I:=NUMTOT+1 TO NUMDIV DO X[I]:=X[I-1]+DIVL;
    X[NUMDIV]:=X[NUMDIV-1]+RESL;
    X[NUMDIV+1]:=X[NUMDIV]+DIVL;
    END;(*SECTION*)
```

```
FUNCTION STAGTEMP(T0,TW,NU,KF,MDOT,L:REAL):REAL;
    (* CALCULATES NEW STAGNATION TEMPERATURE FOR FLOW WITH HEAT TRANSFER *)
    BEGIN
    STAGTEMP:=TW+((T0-TW)/EXP((NU*PI*KF*L)/(MDOT*CP)));
    END;(*STAGTEMP*)

FUNCTION RFUNC(M,T0,TW,PR,NU:REAL; KEY:INTEGER):REAL;
    (* SOLVES AN OFTEN USED FUNCTION IN PROCEDURE HEATTRANS *)
    VAR A,C,D,E:REAL;
    BEGIN
    A:=M*M/(1.0*(1.0-SQR(M)));
    C:=(1.0+K*SQR(M))/T0;
    D:=K*SQR(M);
    IF KEY = 1 THEN
        BEGIN
        E:=2.0/(TW-T0);
        END(*IF*)
    ELSE
        BEGIN
        E:=16.0*PR/(NU*(TW-T0));
        END;(*ELSE*)
    RFUNC:=A*(D*E+C);
    END;(* RFUNC *)


PROCEDURE HEATTRANS(TW,L,D,MDOT:REAL; N:INTEGER; VAR M,T0,P0,T,P,RHO,RE,F,Q,QX,NU:REAL);
    (* CALCULATES FLOW VARIABLES FOR FLOW WITH FRICTION AND HEAT TRANSFER *)
    VAR
        C1,C2,C3,C4,STEP,A,C:REAL;
        T01,T02,M1,M2,PR,KF,DELM,X:REAL;
        KEY,I:INTEGER;
    BEGIN
    KEY:=0;
    (* FLAG KEY IF FLOW TURBULENT *)
    IF NU > 3.66 THEN KEY:=1;
    (* CALCULATE NEW STAGNATION TEMPERATURE *)
    PR:=PRANDTL(T);
    KF:=COND(T);
    T01:=T0;
    T02:=STAGTEMP(T01,TW,NU,KF,MDOT,L);
    (* STEPWISE SOLUTION USING RUNGE-KUTTE METHOD *)
    (* CALCULATE STEP SIZE *)
    STEP:=ABS((T02-T01)/N);
    M1:=M;
    I:=1;
    WHILE I <= N DO
        BEGIN
        C1:=RFUNC(M,T0,TW,PR,NU,KEY);
        IF C1 < 0 THEN C1:=0;(*FUNCTION UNSTABLE*)
        C2:=RFUNC((M+C1*STEP/2.0),(T0+STEP/2.0),TW,PR,NU,KEY);
        IF C2 < 0 THEN C2:=0;(*FUNCTION UNSTABLE*)
        C3:=RFUNC((M+C2*STEP/2.0),(T0+STEP/2.0),TW,PR,NU,KEY);
        IF C3 < 0 THEN C3:=0;(*FUNCTION UNSTABLE*)
        C4:=RFUNC(M+C3*STEP,T0+STEP,TW,PR,NU,KEY);
        IF C4 < 0 THEN C4:=0;(*FUNCTION UNSTABLE*)
        DELM:=STEP/6.0*(C1+2.0*C2+2.0*C3+C4);
        M:=M+DELM;
        T0:=T0+STEP;
        I:=I+1;
        IF M > 1 THEN (*FUNCTION UNSTABLE, EXIT LOOP*)
            BEGIN
            I:=N+1;
            M:=1.0;
            END(*IF*)
        END;(* WHILE *)
    IF (C1 = 0) OR (C2 = 0) OR (C3 = 0) OR (C4 = 0) THEN
        (*FUNCTION UNSTABLE, EXIT LOOP*)
        BEGIN
        I:=N+1;
        M:=1;
        END;(*IF*)
    M2:=M;
    A:=M(M2)/M(M1);
    C:=(K+1.0)/(2.0*(K-1.0));
    P0:=P0*(M1/M2)*SQRT(T02/T01)*PWR(A,C);
```

```
        P:=PRES(P0,M1);
        T:=TEMP(T01,M1);
        RHO:=DENSE(P,T);
        RE:=4.0*MDOT/(PI*D*MU(T));
        IF RE (= 2000 THEN
             BEGIN (* LAMINAL FLOW *)
             F:=16.0/RE;
             NU:=3.66;
             END(*IF*)
        ELSE
             BEGIN (* TURBULENT FLOW *)
             F:=0.046/PWR(RE,0.2);
             NU:=0.023*PWR(RE,0.8)*PWR(PR,0.4);
             END;(*ELSE*)
        Q:=CP*MDOT*(T02-T01);
        QX:=QX+Q;
        END;(* HEATTRANS *)




PROCEDURE SETBOUND(KEY:INTEGER; VAR HIB,LOWB,DIA:REAL);
    (* ALTERS DIAMETER SO THAT MACH MAY BE INTERATED IN THE MAIN PROGRAM TO REACH ONE *)
    BEGIN(*SETBOUND*)
    IF KEY = 1 THEN
         BEGIN
         HIB:=DIA;
         END(*IF*)
    ELSE
         BEGIN
         LOWB:=DIA;
         END;(*ELSE*)
    DIA:=(LOWB+HIB)/2.0;
    END;(*SETBOUND*)

PROCEDURE SWAP(VAR A,B:REAL);
    (* USED BY PROCEDURE BSORT *)
    VAR T:REAL;
    BEGIN(*SWAP*)
    T:=A;
    A:=B;
    B:=T;
    END;(*SWAP*)


PROCEDURE BSORT(START,TOP:INTEGER; VAR ARRY:COUNT; VAR CNT:ICOUNT);
    (* USED IN FUNCTION MDOTMIN TO FIND OPTIMUM MASS FLOW *)

    VAR
         RCNT:COUNT;
         INTEGER;
         SWITCH:BOOLEAN;
    BEGIN(*BSORT*)
    FOR I:=START TO TOP DO RCNT[I]:=CNT[I];
    REPEAT(* UNTIL SORTED WITH SMALLEST AT TOP *)
         SWITCH:=FALSE;
         FOR I:=START TO TOP-1 DO
              BEGIN
              IF ARRY[I] ) ARRY[I+1] THEN
                   BEGIN
                   SWAP(ARRY[I],ARRY[I+1]);
                   SWAP(RCNT[I],RCNT[I+1]);
                   SWITCH:=TRUE;
                   END(*IF*)
              END;(*FOR*)
    UNTIL SWITCH = FALSE;
    FOR I:=START TO TOP DO CNT[I]:=ROUND(RCNT[I]);
    END;(*BSORT*)


FUNCTION MDOTMIN(Q,TW:COUNT; T01,EFF:REAL; N:INTEGER):REAL;
    (* DETERMINES OPTIMUM MASS FLOW *)

    VAR
         QTOT,TOMAX,MDOT:COUNT;
         INDEX:ICOUNT;
         I,NCONT:INTEGER;
```

```
      BEGIN(*MDOTMIN*)
      (* INITIALIZE INDEX ARRAY *)
      FOR I:=0 TO N DO INDEX[I]:=I;
      (* CALCULATE SHIELD EXIT STAGNATION TEMPERATURES *)
      TOMAX[0]:=T01;
      FOR I:=1 TO N DO TOMAX[I]:=EFF*(TW[I]-TOMAX[I-1])+TOMAX[I-1];
      (* CALCULATE MASS FLOWS AS DETERMINED BY SECTIONS *)
      QTOT[0]:=0;
      FOR I:=1 TO N DO
           BEGIN
           QTOT[I]:=QTOT[I-1]+Q[I];
           MDOT[I]:=QTOT[I]/(CP*(TOMAX[I]-T01));
           END;(*FOR*)
      (* SEARCH FOR LARGEST MDOT *)
      BSORT(1,N,MDOT,INDEX);
      MDOTMIN:=MDOT[N];
      END;(*MDOTMIN*)


  PROCEDURE INREASSIGN;
      (*DEFINES VALUES FOR USE IN PARAMETER LIST *)
      BEGIN(*INREASSIGN*)
      M1:=M[I];
      T01:=T0[I];
      P01:=P0[I];
      T1:=T[I];
      P1:=P[I];
      RHO1:=RHO[I];
      F1:=F[I];
      RE1:=RE[I];
      Q1:=Q[I];
      QX1:=QX[I];
      MU1:=MU[I];
      END;(*INREASSIGN*)


  PROCEDURE OUTREASSIGN;
      (* DEFINES NEW VALUES OBTAINED FROM PARAMETER LIST *)
      BEGIN(*OUTREASSIGN*)
      I:=I+1;
      M[I]:=M1;
      T0[I]:=T01;
      P0[I]:=P01;
      T[I]:=T1;
      P[I]:=P1;
      RHO[I]:=RHO1;
      F[I]:=F1;
      RE[I]:=RE1;
      Q[I]:=Q1;
      QX[I]:=QX1;
      MU[I]:=MU1;
      END;(*OUTREASSIGN*)


  PROCEDURE INITIALIZE;
      (* PREPARES VARIABLES FOR USE IN MAIN PROGRAM *)
      BEGIN(*INITIALIZE*)
      (*DETERMINE MINIMUM MASS FLOW*)
      MDOT:=MDOTMIN(QW,TW,T0[1],MAXEFF,NUMWALLS);
      (* DEFINE FIRST SECTION SIZE *)
      X[1]:=DIVLENGTH;
      (* DIFINE IMAGINARY LAST PIPE LENGTH *)
      LSPACE[NUMWALLS+2]:=0;
      (*START LOOP COUNTER*)
      LOOPCOUNT:=0;
      (*INITIAL MACH NUMBER GUESS*)
      M[1]:=0.1;
      (* INITIAL DIAMETER GUESS, ASSUMES INLET MACH OF 0.1 *)
      DIA[1]:=DIAMETER(MDOT,T0[1],P0[1],M[1]);
      (* CALCULATE MINIMUM DIAMETER, MACH = 1 *)
      LOWBOUND:=DIAMETER(MDOT,T0[1],P0[1],1.0);
      (* DEFINE MAXIMUM DIAMETER *)
      HIBOUND:=100*LOWBOUND;
      END;(*INITIALIZE*)


  PROCEDURE REINITIAL;
      (* PREPARES VARIABLES FOR USE IN MAIN PROGRAM *)
      BEGIN(*REINITIAL*)
```

```
        (*INITIALIZE COUNTERS*)
        I:=1;
        PIPECNT:=1;
        WALLCNT:=1;
        SPACECNT:=1;
        (*INCREMENT LOOP COUNTER*)
        LOOPCOUNT:=LOOPCOUNT+1;
        (* DEFINE FIRST PIPE LENGTH *)
        LTOTAL:=LSPACE[1];
        (*CALCULATE ENTRANCE CONDITIONS*)
        QX[1]:=0;Q[1]:=0;
        RE[1]:=4.0*MDOT/(PI*MU(T0[1])*DIA[LOOPCOUNT]);
        IF RE[1] (= 2000 THEN
             BEGIN (* LAMINAR FLOW *)
             F[1]:=16.0/RE[1];
             MU[1]:=3.66;
             END(*IF*)
        ELSE
             BEGIN (* TURBULENT FLOW *)
             F[1]:=0.046/PWR(RE[1],0.2);
             MU[1]:=0.023*PWR(RE[1],0.8)*PWR(PRANDTL(T0[1]),0.4);.
             END;(*ELSE*)
        (*CALCULATE INLET CONDITIONS*)
        INLETMACH(DIA[LOOPCOUNT],MDOT,P0[1],T0[1],M[1],T[1],P[1],RHO[1]);
        (*INITIALIZE CONTROL CARD*)
        MTEST:=1;
        END;(*REINITIAL*)


BEGIN(* MAIN PROGRAM *)
     WRITELN('TO ENTER DATA, ENTER 1.  OTHERWISE,ENTER 0.');
     READLN;
     READ(TAB);
     INDATA(TAB);
     INITIALIZE;

     REPEAT (* UNTIL THE FLOW IS CHOKED *)
     REINITIAL;
     WHILE (SPACECNT (= NUMWALLS+1) AND (MTEST ( 2) DO
          BEGIN
          (* TEST FOR CHANGE IN TEMPERATURE BOUNDARY *)
          IF X[I] ( (LTOTAL-DIVLENGTH/2.0) THEN
               BEGIN(* FRICTION FLOW *)
               (* DIVIDE FRICTION LENGTH INTO SECTIONS *)
               SECTION(I,LSPACE[SPACECNT],DIVLENGTH,X,NUMDIV);
               (* DEFINE END MACH NUMBER FOR TEST AT END *)
               M[NUMDIV]:=0;
               WHILE (I ( NUMDIV) AND (MTEST ( 2) DO
                    BEGIN
                    L:=X[I]-X[I-1];
                    INREASSIGN;
                    FRICTION(T01,DIA[LOOPCOUNT],F1,L,M1,T1,P1,RHO1,P01,Q1);
                    OUTREASSIGN;
                    IF M[I] ) 0.97 THEN MTEST:=2 (*EXIT LOOP*)
                    END;(*WHILE*)
               SPACECNT:=SPACECNT+1;
               PIPECNT:=PIPECNT+1;
               END(* FRICTION FLOW *)
          ELSE
               BEGIN(* TEMPERATURE BOUNDARY CHANGE *)
               QWALL[WALLCNT]:=0;
               LTEMP[WALLCNT]:=0;
               EFF[WALLCNT]:=0;
               TOSAVE:=T0[I];
               WHILE (QW[WALLCNT] ) QWALL[WALLCNT])  AND (EFF[WALLCNT] ( MAXEFF) AND (MTEST ( 2) DO
                    BEGIN
                    INREASSIGN;
                    HEATTRANS(TW[WALLCNT],DIVLENGTH,DIA[LOOPCOUNT],MDOT,N,M1,T01,P01,T1,P1,RHO1,RE1,F1,Q1,QX1,MU1);
                    OUTREASSIGN;
                    QWALL[WALLCNT]:=QWALL[WALLCNT]+Q1;
                    EFF[WALLCNT]:=(T0[I]-TOSAVE)/(TW[WALLCNT]-TOSAVE);
                    LTEMP[WALLCNT]:=LTEMP[WALLCNT]+DIVLENGTH;
                    X[I]:=X[I-1]+DIVLENGTH;
                    IF M[I] ) 0.97 THEN MTEST:=3;(*EXIT LOOPS*)
                    END;(*WHILE*)
               LTOTAL:=LTOTAL+LTEMP[WALLCNT]+LSPACE[SPACECNT];
               PIPECNT:=PIPECNT+1;
               WALLCNT:=WALLCNT+1;
               END;(* TEMPERATURE BOUNDARY CHANGE *)
          END;(*WHILE*);
```

```
DIA1:=DIA[LOOPCOUNT];
SETBOUND(MTEST,HIBOUND,LOWBOUND,DIA1);
DIA[LOOPCOUNT+1]:=DIA1;
UNTIL (M[NUMDIV] > 0.97) AND (PIPECNT >= 1*NUMWALLS+1) OR (LOOPCOUNT > 30);


(*CALCULATE OVERALL EFFECTIVENESS *)
EFF[NUMWALLS+1]:=(T0[NUMDIV]-T0[1])/(TW[NUMWALLS]-T0[1]);
(* CALCULATE TOTAL HEAT TRANSFER *)
QWALL[NUMWALLS+1]:=MDOT*CP*(T0[NUMDIV]-T0[1]);
WRITELN('OUTPUT STORED IN FILE EFFOUTD.');
WRITELN('INPUT STORED IN FILES EFFIND1 AND EFFIND2.');
OUTDATA;
END.(*MAIN PROGRAM*)
```

```
BIND.
TO ENTER DATA, ENTER 1.  OTHERWISE,ENTER 0.
? 1
INPUT DATA AS  PROMPTED BY TERMINAL

WHEN ASKED FOR A SERIES OF DATA SUCH AS THE SHIELD
TEMPERATURES, INPUT THE DATA IN THE ORDER FROM DEWAR
TO THE OUTER BOUNDARY.

ENTER THE NUMBER OF RADIATION SHIELDS
? 5
NUMBER OF SHIELDS=     5

ENTER SINGLE DIVISION LENGTH (M)
? 0.002
DIVISION LENGTH (M)=          0.00200

ENTER PIPE LENGTHS NOT ON SHIELDS (M)
? 0.019
? 0.021
? 0.024
? 0.026
? 0.034
? 0.035
ENTER TEMPERATURES OF RADIATION SHIELDS (K)
? 21.7
? 39.7
? 65.4
? 101.1
? 146.6
ENTER HEAT INPUTS OF RADIATION SHIELDS (W)
? 0.0884
? 0.054
? 0.0916
? 0.1386
? 0.8242
ENTER MAXIMUM HEAT EXCHANGER EFFECTIVENESS AS A FRACTION
? 0.98
SHIELD EFFECTIVENESS=       0.9800000

ENTER INITIAL STAGNATION T (K)
? 4.23
INITIAL STAGNATION TEMPERATURE (K)=          4.23000

ENTER INITIAL STAGNATION P (PA)
? 2100.0
INITIAL STAGNATION PRESSURE (PA)=      2100.00000

ENTER NUMBER OF STEPS IN RUNGE-KUTTA METHOD
? 4

STEPS IN RUNGE-KUTTA=     4
OUTPUT STORED IN FILE EFFOUTD.
INPUT STORED IN FILES EFFIND1 AND EFFIND2.
FOR AN OUTPUT OF SECTION FLOW VARIABLES, INPUT 1. OTHERWISE INPUT 0.
? 1
FOR A PLOT FILE ENTER 1, OTHERWISE ENTER 0
? 0
                              NUMBER OF INTERATIONS=    13
FOR INTERATION INFORMATION, INPUT1. OTHERWISE INPUT 0
? 1
     2.806 CP SECS, 102716B CM USED.
/
```

VALUES OF INPUT

INLET STAGNATION TEMPERATURE (K)= 4.2300000000
INLET STAGNATION PRESSURE (PA)=2100.0000000000
NUMBER OF RADIATION SHIELDS= 5

TEMPERATURES OF RADIATIATION SHIELDS
SHIELD TEMPERATURE (K) 1= 21.7000000000
SHIELD TEMPERATURE (K) 2= 39.7000000000
SHIELD TEMPERATURE (K) 3= 65.4000000000
SHIELD TEMPERATURE (K) 4= 101.1000000000
SHIELD TEMPERATURE (K) 5= 146.6000000000

HEAT INPUTS TO RADIATION SHIELDS
SHIELD HEAT (W) 1= 0.0884000000
SHIELD HEAT (W) 2= 0.0540000000
SHIELD HEAT (W) 3= 0.0916000000
SHIELD HEAT (W) 4= 0.1386000000
SHIELD HEAT (W) 5= 0.8242000000

PIPE LENGTHS BETWEEN SHIELDS
SPACE LENGTH (M) 1= 0.0190000000
SPACE LENGTH (M) 2= 0.0210000000
SPACE LENGTH (M) 3= 0.0240000000
SPACE LENGTH (M) 4= 0.0260000000
SPACE LENGTH (M) 5= 0.0340000000
SPACE LENGTH (M) 6= 0.0350000000

SOLUTION SECTION SIZE (M)= 0.0020000000
MAXIMUM SHIELD EXCHANGER EFFECTIVENESS= 0.9800000000

VALUES OF OUTPUT

MASS FLOW (KG/S)= 0.0000016272
PIPE DIAMETER (M)= 0.0018264434

TOTAL PIPE LENGTH (M)= 0.2530000000
SHIELD LENGTHS
SHIELD LENGTH (M) 1= 0.0320000000
SHIELD LENGTH (M) 2= 0.0100000000
SHIELD LENGTH (M) 3= 0.0080000000
SHIELD LENGTH (M) 4= 0.0060000000
SHIELD LENGTH (M) 5= 0.0380000000

TOTAL HEAT TRANSFER (W)= 1.1923520433
SHIELD HEAT TRANSFER
SHIELD HEAT (W) 1= 0.0906996368
SHIELD HEAT (W) 2= 0.0642488155
SHIELD HEAT (W) 3= 0.1081726368
SHIELD HEAT (W) 4= 0.1577594101
SHIELD HEAT (W) 5= 0.7714723449

OVERALL EXCHANGER EFFECTIVENESS= 0.7898146448
SHIELD EFFECTIVENESS
SHIELD EFF 1= 0.6135927631
SHIELD EFF 2= 0.3067910789
SHIELD EFF 3= 0.2983648314
SHIELD EFF 4= 0.2834763520
SHIELD EFF 5= 0.9843446598

EXIT STAGNATION TEMERATURE (K)= 145.1498257355
EXIT STAGNATION PRESSURE (PA)= 469.2579023906

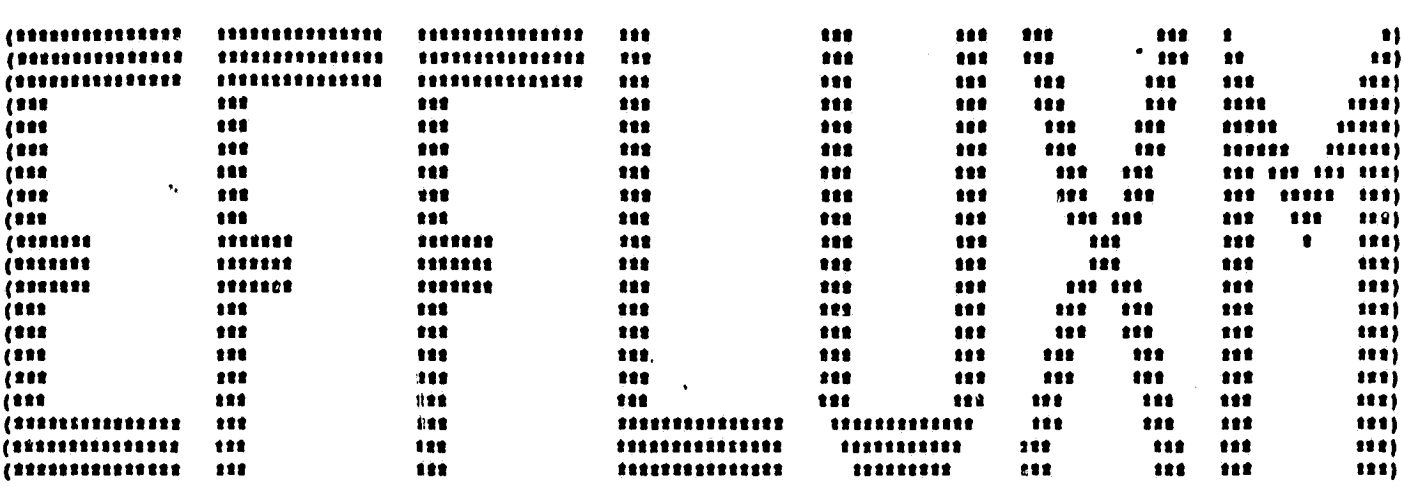
LENGTH	MACH NUMBER	STAG TEMP	STAG PRES	TEMP	PRESSURE	NUSSELT	REYNOLD NUM	SECTION HEAT	TOTAL HEAT
0.00200	0.02145643	4.230000	2100.00000	4.2293	2099.193	3.660	713.65327	0.0000000	0.0000000
0.00400	0.02149790	4.230000	2095.97130	4.2293	2095.163	3.660	713.65327	0.0000000	0.0000000
0.00600	0.02153925	4.230000	2091.95006	4.2293	2091.140	3.660	713.65327	0.0000000	0.0000000

0.00800	0.02158068	4.230000	2087.93629	4.2293	2087.125	3.660	713.65327	0.0000000	0.0000000
0.01000	0.02162220	4.230000	2083.92994	4.2293	2083.117	3.660	713.65327	0.0000000	0.0000000
0.01200	0.02166379	4.230000	2079.93106	4.2293	2079.116	3.660	713.65327	0.0000000	0.0000000
0.01400	0.02170547	4.230000	2075.93957	4.2293	2075.123	3.660	713.65327	0.0000000	0.0000000
0.01600	0.02174724	4.230000	2071.95549	4.2293	2071.137	3.660	713.65327	0.0000000	0.0000000
0.01800	0.02178908	4.230000	2067.97879	4.2293	2067.159	3.660	713.65327	0.0000000	0.0000000
0.01900	0.02183101	4.230000	2064.00946	4.2293	2063.188	3.660	713.65327	0.0000000	0.0000000
0.02100	0.02393310	5.001781	2063.72687	5.0008	2062.740	3.660	672.91490	0.0072071	0.0072071
0.02300	0.02582612	5.914977	2063.42159	5.9137	2062.273	3.660	637.46097	0.0070498	0.0142569
0.02500	0.02754926	6.727647	2063.09309	6.7259	2061.786	3.660	606.44139	0.0068762	0.0211331
0.02700	0.02912941	7.518072	2062.74092	7.5159	2061.280	3.660	579.15077	0.0066879	0.0278211
0.02900	0.03058602	8.284762	2062.36478	8.2822	2060.755	3.660	555.03770	0.0064871	0.0343082
0.03100	0.03193379	9.026443	2061.96446	9.0234	2060.210	3.660	533.63239	0.0062757	0.0405839
0.03300	0.03318426	9.742153	2061.53991	9.7386	2059.646	3.660	514.56304	0.0060556	0.0466395
0.03500	0.03434676	10.431042	2061.09116	10.4269	2059.062	3.660	497.51821	0.0058288	0.0524683
0.03700	0.03542904	11.092568	2060.61836	11.0879	2058.460	3.660	482.23767	0.0055972	0.0580655
0.03900	0.03643746	11.726346	2060.12177	11.7211	2057.840	3.660	468.50226	0.0053626	0.0634281
0.04100	0.03737833	12.332237	2059.60174	12.3265	2057.201	3.660	456.12613	0.0051264	0.0685547
0.04300	0.03825605	12.910260	2059.05871	12.9039	2056.545	3.660	444.95061	0.0048908	0.0734455
0.04500	0.03907531	13.460593	2058.49318	13.4537	2055.871	3.660	434.83956	0.0046565	0.0781020
0.04700	0.03984014	13.983576	2057.90573	13.9761	2055.181	3.660	425.67537	0.0044251	0.0825270
0.04900	0.04055424	14.479673	2057.29700	14.4717	2054.474	3.660	417.35605	0.0041976	0.0867246
0.05100	0.04122099	14.949466	2056.66767	14.9410	2053.753	3.660	409.77270	0.0039750	0.0906996
0.05300	0.04127484	14.949466	2053.99041	14.9409	2051.071	3.660	409.77270	0.0000000	0.0906996
0.05500	0.04132878	14.949466	2051.31545	14.9409	2048.393	3.660	409.77270	0.0000000	0.0906996
0.05700	0.04138283	14.949466	2048.64279	14.9409	2045.716	3.660	409.77270	0.0000000	0.0906996
0.05900	0.04143696	14.949466	2045.97241	14.9407	2043.042	3.660	409.77270	0.0000000	0.0906996
0.06100	0.04149119	14.949466	2043.30433	14.9408	2040.370	3.660	409.77270	0.0000000	0.0906996
0.06300	0.04154552	14.949466	2040.63853	14.9408	2037.700	3.660	409.77270	0.0000000	0.0906996
0.06500	0.04159994	14.949466	2037.97501	14.9408	2035.033	3.660	409.77270	0.0000000	0.0906996
0.06700	0.04165446	14.949466	2035.31377	14.9408	2032.368	3.660	409.77270	0.0000000	0.0906996
0.06900	0.04170908	14.949466	2032.65480	14.9408	2029.705	3.660	409.77270	0.0000000	0.0906996
0.07100	0.04176379	14.949466	2029.99810	14.9407	2027.045	3.660	409.77270	0.0000000	0.0906996
0.07300	0.04181860	14.949466	2027.34367	14.9407	2024.386	3.660	409.77270	0.0000000	0.0906996
0.07400	0.04406303	16.577966	2026.42778	16.5672	2023.146	3.660	385.71106	0.0137791	0.1044787
0.07600	0.04614323	18.157728	2025.42840	18.1448	2021.832	3.660	365.08728	0.0133666	0.1178453
0.07800	0.04807253	19.681992	2024.34396	19.6668	2020.443	3.660	347.32619	0.0128972	0.1307425
0.08000	0.04986203	21.145152	2023.17362	21.1276	2018.980	3.660	331.95665	0.0123801	0.1431226
0.08200	0.05152121	22.542789	2021.91730	22.5227	2017.443	3.660	318.68058	0.0118250	0.1549477
0.08400	0.05160007	22.542789	2018.83798	22.5226	2014.357	3.660	318.68058	0.0000000	0.1549477
0.08600	0.05167913	22.542789	2015.76063	22.5226	2011.272	3.660	318.68058	0.0000000	0.1549477
0.08800	0.05175838	22.542789	2012.68523	22.5225	2008.190	3.660	318.68058	0.0000000	0.1549477
0.09000	0.05183782	22.542789	2009.61177	22.5224	2005.110	3.660	318.68058	0.0000000	0.1549477
0.09200	0.05191746	22.542789	2006.54025	22.5224	2002.031	3.660	318.68058	0.0000000	0.1549477
0.09400	0.05199729	22.542789	2003.47067	22.5223	1998.955	3.660	318.68058	0.0000000	0.1549477
0.09600	0.05207732	22.542789	2000.40302	22.5222	1995.880	3.660	318.68058	0.0000000	0.1549477
0.09800	0.05215754	22.542789	1997.33730	22.5222	1992.807	3.660	318.68058	0.0000000	0.1549477
0.10000	0.05223797	22.542789	1994.27349	22.5221	1989.737	3.660	318.68058	0.0000000	0.1549477
0.10200	0.05231859	22.542789	1991.21160	22.5221	1986.668	3.660	318.68058	0.0000000	0.1549477
0.10400	0.05239941	22.542789	1988.15161	22.5220	1983.601	3.660	318.68058	0.0000000	0.1549477
0.10600	0.05248043	22.542789	1985.09353	22.5219	1980.536	3.660	318.68058	0.0000000	0.1549477
0.10800	0.05627788	25.861383	1983.27537	25.8340	1978.040	3.660	291.16467	0.0280806	0.1830276
0.11000	0.05979693	29.119928	1981.20513	29.0851	1975.302	3.660	268.85406	0.0275712	0.2105989
0.11200	0.06305392	32.285005	1978.87034	32.2421	1972.316	3.660	250.54924	0.0267804	0.2373793
0.11400	0.06606199	35.327246	1976.26272	35.2757	1969.079	3.660	235.41742	0.0257410	0.2631203
0.11600	0.06619920	35.327246	1972.19049	35.2755	1964.992	3.660	235.41742	0.0000000	0.2631203
0.11800	0.06633697	35.327246	1968.11862	35.2752	1960.905	3.660	235.41742	0.0000000	0.2631203
0.12000	0.06647530	35.327246	1964.04710	35.2750	1956.819	3.660	235.41742	0.0000000	0.2631203
0.12200	0.06661420	35.327246	1959.97592	35.2748	1952.732	3.660	235.41742	0.0000000	0.2631203
0.12400	0.06675367	35.327246	1955.90507	35.2746	1948.646	3.660	235.41742	0.0000000	0.2631203
0.12600	0.06689372	35.327246	1951.83451	35.2744	1944.561	3.660	235.41742	0.0000000	0.2631203
0.12800	0.06703435	35.327246	1947.76426	35.2741	1940.475	3.660	235.41742	0.0000000	0.2631203
0.13000	0.06717557	35.327246	1943.69423	35.2739	1936.390	3.660	235.41742	0.0000000	0.2631203
0.13200	0.06731737	35.327246	1939.62448	35.2737	1932.305	3.660	235.41742	0.0000000	0.2631203
0.13400	0.06745977	35.327246	1935.55496	35.2735	1928.220	3.660	235.41742	0.0000000	0.2631203
0.13600	0.06760277	35.327246	1931.48565	35.2732	1924.135	3.660	235.41742	0.0000000	0.2631203
0.13800	0.06774636	35.327246	1927.41654	35.2730	1920.050	3.660	235.41742	0.0000000	0.2631203
0.14000	0.06789057	35.327246	1923.34761	35.2728	1915.965	3.660	235.41742	0.0000000	0.2631203
0.14200	0.07391159	41.658562	1919.54817	41.5825	1910.828	3.660	209.77276	0.0535706	0.3166909
0.14400	0.07949989	47.914294	1915.02088	47.8131	1904.952	3.660	190.04427	0.0529311	0.3696219
0.14600	0.08465805	53.972266	1909.71888	53.8430	1898.338	3.660	174.69468	0.0512578	0.4208797
0.14800	0.08490560	53.972266	1904.20428	53.8422	1892.798	3.660	174.69468	0.0000000	0.4208797
0.15000	0.08515503	53.972266	1898.68018	53.8415	1887.232	3.660	174.69468	0.0000000	0.4208797
0.15200	0.08540637	53.972266	1893.14674	53.8407	1881.665	3.660	174.69468	0.0000000	0.4208797
0.15400	0.08565963	53.972266	1887.60377	53.8399	1876.088	3.660	174.69468	0.0000000	0.4208797
0.15600	0.08591486	53.972266	1882.05119	53.8391	1870.501	3.660	174.69468	0.0000000	0.4208797
0.15800	0.08617207	53.972266	1876.48898	53.8383	1864.904	3.660	174.69468	0.0000000	0.4208797
0.16000	0.08643129	53.972266	1870.91681	53.8375	1859.297	3.660	174.69468	0.0000000	0.4208797
0.16200	0.08669254	53.972266	1865.33480	53.8367	1853.680	3.660	174.69468	0.0000000	0.4208797
0.16400	0.08695586	53.972266	1859.74279	53.8359	1848.053	3.660	174.69468	0.0000000	0.4208797

0.16600	0.08722128	53.972266	1854.14068	53.8351	1842.415	3.660	174.69468	0.0000000	0.4208797
0.16800	0.08746881	53.972266	1848.52835	53.8342	1836.764	3.660	174.69468	0.0000000	0.4208797
0.17000	0.08775850	53.972266	1842.90572	53.8334	1831.108	3.660	174.69468	0.0000000	0.4208797
0.17200	0.08803037	53.972266	1837.27266	53.8325	1825.438	3.660	174.69468	0.0000000	0.4208797
0.17400	0.08830444	53.972266	1831.62908	53.8316	1819.757	3.660	174.69468	0.0000000	0.4208797
0.17600	0.08858077	53.972266	1825.97487	53.8308	1814.066	3.660	174.69468	0.0000000	0.4208797
0.17800	0.08885936	53.972266	1820.30992	53.8299	1808.364	3.660	174.69468	0.0000000	0.4208797
0.18000	0.08914026	53.972266	1814.63411	53.8290	1802.650	3.660	174.69468	0.0000000	0.4208797
0.18200	0.09858068	65.379618	1806.69984	65.0678	1792.122	3.660	152.82677	0.0956739	0.5165536
0.18400	0.10736737	76.408518	1796.84721	76.1146	1779.667	3.660	136.99533	0.0941639	0.6107174
0.18600	0.11546331	86.992652	1784.96896	86.6059	1765.253	3.660	125.39333	0.0895546	0.7002721
0.18800	0.12286091	96.739112	1771.04415	96.2524	1748.917	3.660	116.77958	0.0824468	0.7827189
0.19000	0.12958520	105.454112	1755.12443	104.8642	1730.755	3.660	110.37922	0.0737394	0.8564723
0.19200	0.13569032	113.846569	1737.31321	112.3536	1710.889	3.660	105.54919	0.0642414	0.9207197
0.19400	0.14125145	119.514122	1717.74464	118.7206	1689.458	3.660	101.89539	0.0547233	0.9754430
0.19600	0.14635571	124.919892	1696.56558	124.0299	1666.598	3.660	99.11946	0.0457394	1.0211824
0.19800	0.15109428	129.367649	1673.92187	128.3858	1642.434	3.660	97.00377	0.0376334	1.0588158
0.20000	0.15553611	132.980421	1649.94869	131.9111	1617.078	3.660	95.38783	0.0305684	1.0893842
0.20200	0.15982530	135.894660	1624.76454	134.7317	1590.621	3.660	94.15215	0.0245734	1.1139576
0.20400	0.16397866	138.200034	1598.46807	136.9663	1563.136	3.660	93.20710	0.0195909	1.1335484
0.20600	0.16800595	140.033837	1571.13709	138.7209	1534.676	3.660	92.48498	0.0155162	1.1490646
0.20800	0.17221054	141.478722	1542.82886	140.0870	1505.277	3.660	91.93447	0.0122255	1.1612901
0.21000	0.17641887	142.612558	1513.58117	141.1411	1474.954	3.660	91.51646	0.0095936	1.1708837
0.21200	0.18074186	143.499492	1483.41368	141.9461	1443.711	3.660	91.20114	0.0075045	1.1783882
0.21400	0.18525494	144.191587	1452.32920	142.5526	1411.531	3.660	90.96574	0.0058568	1.1842442
0.21600	0.19000991	144.730619	1420.31471	143.0011	1378.385	3.660	90.79290	0.0045609	1.1888051
0.21800	0.19505714	145.149826	1387.34260	143.3231	1344.229	3.660	90.66941	0.0035478	1.1923521
0.22000	0.20024529	145.149826	1353.22473	143.1259	1308.958	3.660	90.66941	0.0000000	1.1923521
0.22200	0.20588378	145.149826	1318.15016	143.1176	1272.628	3.660	90.66941	0.0000000	1.1923521
0.22400	0.21207726	145.149826	1281.83618	142.9953	1234.935	3.660	90.66941	0.0000000	1.1923521
0.22600	0.21888823	145.149826	1244.34869	142.6569	1195.929	3.660	90.66941	0.0000000	1.1923521
0.22800	0.22642935	145.149826	1205.56157	142.0989	1155.460	3.660	90.66941	0.0000000	1.1923521
0.23000	0.23484484	145.149826	1165.32401	142.5167	1113.343	3.660	90.66941	0.0000000	1.1923521
0.23200	0.24432308	145.149826	1123.45314	142.3041	1069.354	3.660	90.66941	0.0000000	1.1923521
0.23400	0.25511598	145.149826	1079.72353	142.0526	1023.209	3.660	90.66941	0.0000000	1.1923521
0.23600	0.26756947	145.149826	1033.85181	141.7581	974.546	3.660	90.66941	0.0000000	1.1923521
0.23800	0.28217612	145.149826	985.47338	141.3787	922.888	3.660	90.66941	0.0000000	1.1923521
0.24000	0.29966631	145.149826	934.10549	140.9108	867.584	3.660	90.66941	0.0000000	1.1923521
0.24200	0.32118403	145.149826	879.08606	140.3012	807.707	3.660	90.66941	0.0000000	1.1923521
0.24400	0.34865459	145.149826	819.46558	139.4782	741.861	3.660	90.66941	0.0000000	1.1923521
0.24600	0.38565968	145.149826	753.80333	138.2689	667.763	3.660	90.66941	0.0000000	1.1923521
0.24800	0.43993993	145.149826	679.76833	136.3116	581.233	3.660	90.66941	0.0000000	1.1923521
0.25000	0.53298068	145.149826	593.38541	132.5372	473.880	3.660	90.66941	0.0000000	1.1923521
0.25200	0.74448831	145.149826	492.88279	121.3842	315.637	3.660	90.66941	0.0000000	1.1923521
0.25300	1.00000000	145.149826	469.25790	108.7265	228.373	3.660	90.66941	0.0000000	1.1923521

INTERATION= 1 DIAMETER= 0.0008487349
INTERATION= 2 DIAMETER= 0.0182608843
INTERATION= 3 DIAMETER= 0.0095547696
INTERATION= 4 DIAMETER= 0.0052017523
INTERATION= 5 DIAMETER= 0.0030252436
INTERATION= 6 DIAMETER= 0.0019369893
INTERATION= 7 DIAMETER= 0.0013928621
INTERATION= 8 DIAMETER= 0.0016649257
INTERATION= 9 DIAMETER= 0.0018009575
INTERATION= 10 DIAMETER= 0.0018689734
INTERATION= 11 DIAMETER= 0.0018349654
INTERATION= 12 DIAMETER= 0.0018179615
INTERATION= 13 DIAMETER= 0.0018264634

```
TYPE,EFFLUXM
PROGRAM EFFLUXM(INPUT/,OUTPUT,EFFOUTM,EFFINM1,EFFINM2,POUT);
```



J. C. CHATO AND K. V. BRENDLEY

UNIVERSITY OF ILLINOIS

UNDER CONTRACT OF NASA, AMES RESEARCH

JULY 1, 1982

```
(*     THIS PROGRAM DETERMINES CERTAIN PARAMETERS OF THE EFFLUX     *)
(*  RESULTING FROM THE PARASITIC HEAT LOAD ON A HELIUM DEWAR IN     *)
(*  SPACE.  THE SHIELD SYSTEM CONSISTS OF CONSTANT TEMPERATURE      *)
(*  SHIELDS WITH A GIVEN DIAMETER AND PIPE LENGTHS.  THE MASS FLOW  *)
(*  AND SHIELD HEAT EXCHANGE ARE THEN CALCULATED ALONG WITH THE FLOW*)
(*  VARIABLES.                                                      *)


LABEL 16;

CONST
    PI=3.141592653;
    R=2077;(*J/KG-K*)
    K=1.67;
    CP=5200;

TYPE
    DIVISIONS=ARRAY[0..20001 OF REAL;
    COUNT=ARRAY[0..20] OF REAL;
    ICOUNT=ARRAY[0..20] OF INTEGER;
    REALOUT=FILE OF REAL;
    INTOUT=FILE OF INTEGER;

VAR
    NUMWALLS,NUMDIV,N,NTEST,PKEY:INTEGER;
    I,LOOPCOUNT,TAB,PIPECNT,WALLCNT,SPACECNT:INTEGER;
    EFFOUTM,POUT:TEXT;
    EFFINM1:REALOUT;
    EFFINM2:INTOUT;
    P,T,RHO:DIVISIONS;
    X,QX,M,P0,T0,Q,RE,F,MU,MDOT:DIVISIONS;
    EFF,TV,LSPACE,LPIPE,LWALL,QSHIELD:COUNT;
    L,LTOTAL,DIA,LOWBOUND,HIBOUND:REAL;
    T0SAVE,QSAVE,MDOT1,MDOTMAX,MAXEFF:REAL;
```

```
    M1,T01,P01,T1,P1,RHO1,F1,RE1,Q1,QX1,MU1:REAL;


PROCEDURE INDATA(INDEX:INTEGER);
    BEGIN
    IF INDEX ) 0 THEN
        BEGIN
        (* READ IN DATA *)
        WRITELN('INPUT DATA AS  PROMPTED BY TERMINAL');
        WRITELN(' ');
        WRITELN('WHEN ASKED FOR A SERIES OF DATA SUCH AS THE');
        WRITELN('SHIELD TEMPERATURES, INPUT THE DATA IN');
        WRITELN('ORDER FROM THE DEWAR TO THE OUTER BOUNDARY');
        WRITELN(' ');
        WRITELN('ENTER THE NUMBER OF RADIATION SHIELDS');
        READLN;
        READ(NUMWALLS);
        WRITE(EFFINM2,NUMWALLS);
        WRITELN('NUMBER OF SHIELDS=',NUMWALLS:5);
        WRITELN(' ');
        WRITELN('ENTER NUMBER OF SECTIONS FOR SOLUTION');
        READLN;
        READ(NUMDIV);
        WRITE(EFFINM2,NUMDIV);
        WRITELN('NUMBER OF SOLUTION SECTIONS=',NUMDIV:5);
        WRITELN(' ');
        WRITELN('ENTER DIAMETER (M)');
        READLN;
        READ(DIA);
        WRITE(EFFINM1,DIA);
        WRITELN('DIAMETER (M)=',DIA:15:7);
        WRITELN(' ');
        WRITELN('INPUT DISTANCES BETWEEN SHIELDS (M)');
        FOR I:=1 TO NUMWALLS+1 DO
            BEGIN
            READLN;
            READ(LSPACE[I]);
            WRITE(EFFINM1,LSPACE[I]);
            END;(*FOR*)
        WRITELN('ENTER PIPE LENGTHS ALONG SHIELDS (M)');
        FOR I:=1 TO NUMWALLS DO
            BEGIN
            READLN;
            READ(LWALL[I]);
            WRITE(EFFINM1,LWALL[I]);
            END;(*FOR*)
        WRITELN('ENTER TEMPERATURES OF RADIATION SHIELDS (K)');
        FOR I:=1 TO NUMWALLS DO
            BEGIN
            READLN;
            READ(TW[I]);
            WRITE(EFFINM1,TW[I]);
            END;(* FOR *)
        WRITELN('ENTER MAXIMUM SHIELD EXCHANGER EFFECTIVENESS');
        READLN;
        READ(MAXEFF);
        WRITE(EFFINM1,MAXEFF);
        WRITELN('MAXIMUM SHIELD EFFECTIVENESS=',MAXEFF:15:9);
        WRITELN(' ');
        WRITELN('ENTER INITIAL STAGNATION T (K)');
        READLN;
        READ(T0[1]);
        WRITE(EFFINM1,T0[1]);
        WRITELN('INITIAL STAGNATION TEMPERATURE (K)=',T0[1]:15:9);
        WRITELN(' ');
        TW[0]:=T0[1];
        WRITELN('ENTER INITIAL STAGNATION P (PA)');
        READLN;
        READ(P0[1]);
        WRITE(EFFINM1,P0[1]);
        WRITELN('INITIAL STAGNATION PRESSURE (PA)=',P0[1]:15:9);
        WRITELN(' ');
        WRITELN('ENTER MASS FLOW APPROXIMATION (KG/S)');
        READLN;
        READ(MDOT[1]);
        WRITE(EFFINM1,MDOT[1]);
        WRITELN('MASS FLOW (KG/S)=',MDOT[1]:15:9);
        WRITELN(' ');
        WRITELN('ENTER NUMBER OF STEPS IN RUNGE-KUTTA METHOD');
```

```
            READLN;
            READ(N);
            WRITE(EFFINM2,N);
            WRITELN('NUMBER OF STEPS IN RUNGE-KUTTA=',N:5);
            END(*IF*)
        ELSE
            (* READ FILE *)
            BEGIN
            RESET(EFFINM1);
            RESET(EFFINM2);
            READ(EFFINM2,NUMWALLS);
            READ(EFFINM2,NUMDIV);
            READ(EFFINM1,DIA);
            FOR I:=1 TO NUMWALLS+1 DO READ(EFFINM1,LSPACE[I]);
            FOR I:=1 TO NUMWALLS DO READ(EFFINM1,LWALL[I]);
            FOR I:=1 TO NUMWALLS DO READ(EFFINM1,TW[I]);
            READ(EFFINM1,MAXEFF);
            READ(EFFINM1,T0[1]);
            TW[0]:=T0[1];
            READ(EFFINM1,P0[1]);
            READ(EFFINM1,MDOT[1]);
            READ(EFFINM2,N);
            END(*ELSE*)
        END;(* INDATA *)


    PROCEDURE OUTDATA;
        BEGIN
        WRITELN(EFFOUTM,'LAMFLOW OUTPUT':60);
        WRITELN(EFFOUTM,' ');
        WRITELN(EFFOUTM,' ');
        WRITELN(EFFOUTM,' ');
        WRITELN(EFFOUTM,'VALUES OF INPUT':60);
        WRITELN(EFFOUTM,' ');
        WRITELN(EFFOUTM,'NUMBER OF SECTIONS =':50,NUMDIV:5);
        WRITELN(EFFOUTM,'DEWAR STAGNATION TEMPERATURE (K)=':50,T0[1]:20);
        WRITELN(EFFOUTM,'DEWAR STAGNATION PRESSURE (PA)=':50,P0[1]:20);
        WRITELN(EFFOUTM,'MASS FLOW APPROXIMATION (KG/S)=':50,MDOT[1]:20);
        WRITELN(EFFOUTM,' ');
        WRITELN(EFFOUTM,' ');
        WRITELN(EFFOUTM,'SHIELD TEMPERATURES (K)':60);
        FOR I:=1 TO NUMWALLS DO WRITELN(EFFOUTM,'SHIELD TEMPERATURE':46,I:2,'=':1,TW[I]:15:9);
        WRITELN(EFFOUTM,' ');
        WRITELN(EFFOUTM,' ');
        WRITELN(EFFOUTM,' ');
        WRITELN(EFFOUTM,'VALUES OF OUTPUT':60);
        WRITELN(EFFOUTM,' ');
        WRITELN(EFFOUTM,'MASS FLOW (KG/S)=':50,MDOT[LOOPCOUNT]:15:9);
        WRITELN(EFFOUTM,'TOTAL HEAT TRANSFER (W)=':50,QX[NUMDIV]:15:9);
        WRITELN(EFFOUTM,' ');
        WRITELN(EFFOUTM,'SHIELD HEAT TRANSFER (W)':60);
        FOR I:=1 TO NUMWALLS DO WRITELN(EFFOUTM,'SHIELD HEAT (W)':48,I:2,'=':1,QSHIELD[I]:15:9);
        WRITELN(EFFOUTM,' ');
        WRITELN(EFFOUTM,'SHIELD HEAT EXCHANGER EFFECTIVENESS':60);
        FOR I:=1 TO NUMWALLS DO WRITELN(EFFOUTM,'SHIELD EFFECTIVENESS':48,I:2,'=':1,EFF[I]:15:9);
        WRITELN(EFFOUTM,' ');
        WRITELN(EFFOUTM,' ');
        WRITELN('FOR AN OUTPUT OF SECTION FLOW VARIABLES, INPUT 1. OTHERWISE INPUT 0.');
        READLN;
        READ(PKEY);
        IF PKEY = 1 THEN
            BEGIN
            WRITE(EFFOUTM,'LENGTH':10,'MACH NUMBER':15,'STAG TEMP':15,'STAG PRES':15);
            WRITE(EFFOUTM,'TEMP':10,'PRESSURE':10,'NUSSELT #':10,'REYNOLD #':12);
            WRITELN(EFFOUTM,'SECTION HEAT':15,'TOTAL HEAT':15);
            WRITELN(EFFOUTM,' ');
            FOR I:=1 TO NUMDIV DO
                BEGIN
                WRITE(EFFOUTM,' ':2,X[I]:8:5,' ':5,M[I]:10:8,' ':5,T0[I]:10:6);
                WRITE(EFFOUTM,' ':5,P0[I]:10:5,' ':2,T[I]:8:4,' ':2,P[I]:8:3);
                WRITE(EFFOUTM,' ':2,NU[I]:8:3,' ':3,RE[I]:10:5,' ':5,Q[I]:10:7);
                WRITELN(EFFOUTM,' ':5,QX[I]:10:7);
                END(*FOR*)
            END;(*IF*)
        WRITELN(' ');
        WRITELN('FOR A PLOT FILE ENTER 1, OTHERWISE ENTER 0');
        READLN;READ(TAB);
        IF TAB > 0 THEN
            BEGIN
            WRITELN('FOR PLOT OF STAGNATION TEMPERATURE ENTER 1.');
```

```
        WRITELN('FOR PLOT OF STAGNATION PRESSURE ENTER 2.');
        WRITELN('FOR PLOT OF HEAT TRANSFER ENTER 3.');
        READLN;READ(TAB);
        FOR I:=1 TO NUMDIV DO
            BEGIN
            CASE TAB OF
                1:  WRITELN(FOUT,T0[I]);
                2:  WRITELN(FOUT,P0[I]);
                3:  WRITELN(FOUT,QX[I]);
                END(*CASE*)
            END(*FOR*)
        END;(*IF*)
    WRITELN('FOR INTERATION INFORMATION, INPUT 1.  OTHERWISE INPUT 0.');
    READLN;
    READ(FKEY);
    IF FKEY = 1 THEN
        FOR I:=1 TO LOOPCOUNT DO
            WRITELN(EFFOUTH,'INTERATION=':20,I:3,'MASS FLOW=':20,MDOT[I]:15);
    WRITELN('OUTPUT STORED IN FILE EFFOUTH');
    END;(* OUTDATA *)

FUNCTION PWR(N,P:REAL):REAL;
    (* SOLVES FOR N**P *)
    VAR A:REAL;
    BEGIN
    A:=P*LN(N);
    PWR:=EXP(A);
    END;(* PWR *)


FUNCTION B(M:REAL):REAL;
    (* SOLVES OFTEN OCCURRING FUNCTION *)
    BEGIN
    B:=1.0+(K-1.0)*SQR(M)/2.0;
    END;(* B *)


FUNCTION TEMP(T0,M:REAL):REAL;
    (* CALCULATES LOCAL TEMPERATURE *)
    BEGIN
    TEMP:=T0/B(M);
    END;(* TEMP *)


FUNCTION PRES(P0,M:REAL):REAL;
    (* CALCULATES LOCAL PRESSURE *)
    VAR A,C:REAL;
    BEGIN
        C:=B(M);
        A:=K/(1-K);
        PRES:=P0*PWR(C,A);
    END;(* PRES *)


FUNCTION DENSE(P,T:REAL):REAL;
    (* CALCULATES LOCAL DENSITY *)
    BEGIN
    DENSE:=P/(R*T);
    END;(* DENSE *)


FUNCTION MU(T:REAL):REAL;
    VAR A,B,C,D:REAL;
    BEGIN(*MU*)
    A:=1.10419265531E-06;
    B:=1.16867759464E-07*T;
    C:=-3.22292270462E-10*SQR(T);
    D:=4.72570029322E-13*T*SQR(T);
    MU:=A+B+C+D;
    END;(*MU*)


FUNCTION COND(T:REAL):REAL;
    VAR A,B,C,D:REAL;
    BEGIN(*CONDUCTIVITY CALCULATION*)
    A:=0.0157218071;
```

```
        B:=6.3016525E-04*T;
        C:=-6.99407521E-07*SQR(T);
        D:=4.52803094E-10*T*SQR(T);
        COND:=A+B+C+D;
        END;(*COND*)


    FUNCTION PRANDTL(T:REAL):REAL;
        VAR A,B,C,D,E,F:REAL;
        BEGIN(*PRANDTL NUMBER CALCULATION*)
        A:=0.570232994;
        B:=0.005587291*T;
        C:=-7.50324532E-05*SQR(T);
        D:=4.32720501E-07*T*SQR(T);
        E:=-1.11661445E-09*SQR(T)*SQR(T);
        F:=1.07403842E-12*T*SQR(T)*SQR(T);
        PRANDTL:=A+B+C+D+E+F;
        END;(*PRANDTL*)


    FUNCTION MFRIC(M1,M2,D,F,L:REAL):REAL;
        VAR MFR:REAL;
        (* USED IN CASE 2 OF NEWTON *)
        BEGIN
        MFR:=(SQR(M2)-SQR(M1))/(K*SQR(M1*M2))+(K+1.0)/(2.0*K)*LN(B(M2)*SQR(M1)/(B(M1)*SQR(M2)))-4.0*F*L/D;
        MFRIC:=MFR;
        END;(* MFRIC *)


    FUNCTION DMFRIC(M1,M2:REAL):REAL;
        (* USED IN CASE 2 OF NEWTON *)
        VAR DMF:REAL;
        BEGIN
        DMF:=2.0/(K*M2*SQR(M2))+(K+1)/K*M2*SQR(B(M1))/(PWR(M1,4)*PWR(B(M2),3));
        DMFRIC:=DMF;
        IF DMF < 1.274 THEN (*FUNCTION UNSTABLE, MACH=1 APPROACHED*)
            DMFRIC:=1.0;
        END;(* DMFRIC *)


    FUNCTION MFRST(M,D,MDOT,P0,T0:REAL):REAL;
        (* USED IN CASE 3 OF NEWTON *)
        VAR A,C:REAL;
        BEGIN
        A:=B(M);
        C:=(K+1.0)/(2.0*(1.0-K));
        MFRST:=(M*P0*PI*SQR(D)/4.0)*SQRT(K/(R*T0))*PWR(A,C)-MDOT;
        END;(* MFRST *)


    FUNCTION DMFRST(M,D,P0,T0:REAL):REAL;
        (* USED IN CASE 3 OF NEWTON *)
        VAR A,C,E:REAL;
        BEGIN
        A:=B(M);
        C:=(K+1.0)/(2.0*(1.0-K));
        E:=(3.0*K-1.0)/(2.0*(1.0-K));
        DMFRST:=(PI*SQR(D)/4.0)*P0*SQRT(K/(R*T0))*(PWR(A,C)-SQR(M)*(K+1.0)/2.0*PWR(A,E));
        END(* DMFRST *);


    FUNCTION NEWTON(KEY:INTEGER; M1,D,F,L,LTOT,P0,T0:REAL):REAL;
        VAR MNEW,MOLD,DIFF,D1,D2,A,MDOT,DMF:REAL;
            COUNT:INTEGER;
        BEGIN(* NEWTON *)
        DIFF:=1.0;
        CASE KEY OF
            1: BEGIN
            (* CALCULATES EXIT MACH FOR SECTION WITH FRICTION ONLY *)
                MOLD:=1.01*M1;
                COUNT:=1;
                WHILE (DIFF >= 0.00005) AND (COUNT < 15) DO
                    BEGIN
                    DMF:=DMFRIC(M1,MOLD);
                    MNEW:=MOLD-MFRIC(M1,MOLD,D,F,L)/DMF;
                    DIFF:=ABS(MOLD-MNEW);
                    IF DMF = 1.0 THEN (*FUNCTION UNSTABLE, EXIT LOOP*)
```

```
                        BEGIN
                        DIFF:=0.00001;
                        MNEW:=1.0;
                        END;(*IF*)
                    COUNT:=COUNT+1;
                    MOLD:=MNEW;
                    END(* WHILE *)
                END;(*CASE 1*)
            2: BEGIN
            (* CALCULATES INLET MACH NUMBER *)
                MDOT:=L;
                MOLD:=M1;
                WHILE (DIFF )= 0.00001) DO
                    BEGIN
                    MNEW:=MOLD-MFRST(MOLD,D,MDOT,P0,T0)/DMFRST(MOLD,D,P0,T0);
                    DIFF:=ABS(MOLD-MNEW);
                    MOLD:=MNEW;
                    END(* WHILE *)
                END(* 2 *)
            END;(* CASE *)
        NEWTON:=MOLD;
        END;(* NEWTON *)


    PROCEDURE INLETMACH(D,MDOT,P0,T0:REAL; VAR M,T,P,RHO:REAL);
        (* CALCULATES INLET CONDITIONS *)
        VAR A:REAL;
        BEGIN(* INLETMACH *)
            M:=NEWTON(2,M,D,0,MDOT,0.0,P0,T0);
            T:=TEMP(T0,M);
            P:=PRES(P0,M);
            RHO:=DENSE(P,T);
        END;(* INLETMACH *)


    PROCEDURE FRICTION(T0,DIA,F,L:REAL;VAR M,T,P,RHO,P0,Q:REAL);
        (* FOR FLOW WITH FRICTION ONLY *)
        VAR A,C,NEWM:REAL;
        BEGIN
        NEWM:=NEWTON(1,M,DIA,F,L,LTOTAL,0,0);
        A:=R(NEWM)/R(M);
        C:=(K+1.0)/(2.0*(K-1.0));
        P0:=P0*M/NEWM*PVR(A,C);
        M:=NEWM;
        T:=TEMP(T0,M);
        P:=PRES(P0,M);
        RHO:=DENSE(P,T);
        Q:=0;
    END;(*FRICTION*)


    FUNCTION STAGTEMP(T0,TW,NU,KF,MDOT,L:REAL):REAL;
        (* CALCULATES NEW STAGNATION TEMPERATURE FOR FLOW WITH HEAT TRANSFER *)
        BEGIN
        STAGTEMP:=TW+((T0-TW)/EXP((NU*PI*KF*L)/(MDOT*CP)));
        END;(*STAGTEMP*)

    FUNCTION RFUNC(M,T0,TW,PR,NU:REAL; KEY:INTEGER):REAL;
        (* SOLVES AN OFTEN USED FUNCTION IN PROCEDURE HEATTRANS *)
        VAR A,C,D,E:REAL;
        BEGIN
        A:=M*R(M)/(2.0*(1.0-SQR(M)));
        C:=(1.0+K*SQR(M))/T0;
        D:=K*SQR(M);
        IF KEY = 1 THEN
            BEGIN
            E:=2.0/(TW-T0);
            END(*IF*)
        ELSE
            BEGIN
            E:=16.0*PR/(NU*(TW-T0));
            END;(*ELSE*)
        RFUNC:=A*(D*E+C);
        END;(* RFUNC *)
```

```
PROCEDURE HEATTRANS(TW,L,D,MDOT:REAL; N:INTEGER; VAR M,T0,P0,T,P,RHO,RE,F,Q,QX,MU:REAL);
    (* CALCULATES FLOW VARIABLES FOR FLOW WITH FRICTION AND HEAT TRANSFER *)
    VAR
        C1,C2,C3,C4,STEP,A,C,NEWL,FACTOR:REAL;
        T01,T02,M1,M2,PR,KF,DELM:REAL;
        KEY,I,TEST:INTEGER;
    BEGIN
    KEY:=0;
    TEST:=0;
    (* FLAG KEY IF FLOW IS TURBULENT *)
    IF MU ) 3.66 THEN KEY:=1;
    (* CALCULATE NEW STAGNATION TEMPERATURE *)
    PR:=PRANDTL(T);
    KF:=COND(T);
    T01:=T0;
    T02:=STAGTEMP(T01,TW,MU,KF,MDOT,L);
    IF (TW-T02) ( 0.0001 THEN
        (* WALL TEMPERATURE APPROACHED, EXIT PROGRAM *)
        BEGIN
        A:=-(MDOT*CP)/(PI*MU*KF);
        NEWL:=A*LN(0.03);
        FACTOR:=L/NEWL;
        WRITELN('THE FLUID TEMPERATURE APPROACHES THE WALL TEMPERATURE');
        WRITELN('AT AN INDETERMINATE LENGTH.  INCREASE THE NUMBER OF');
        WRITELN('SOLUTION SECTIONS.  SUGGESTED INCREASE BY FACTOR OF',FACTOR:15,',':1);
        WRITELN('OR MORE.');
        GOTO 10;
        END;(*IF*)
    (* STEPWISE SOLUTION USING RUNGE-KUTTE METHOD *)
    (* CALCULATE STEP SIZE *)
    STEP:=ABS((T02-T01)/N);
    M1:=M;
    I:=1;
    WHILE (I (= N) AND (TEST = 0) DO
        BEGIN
        C1:=RFUNC(M,T0,TW,PR,MU,KEY);
        IF C1 ( 0 THEN C1:=0;(*FUNCTION UNSTABLE*)
        C2:=RFUNC((M+C1*STEP/2.0),(T0+STEP/2.0),TW,PR,MU,KEY);
        IF C2 ( 0 THEN C2:=0;(*FUNCTION UNSTABLE*)
        C3:=RFUNC((M+C2*STEP/2.0),(T0+STEP/2.0),TW,PR,MU,KEY);
        IF C3 ( 0 THEN C3:=0;(*FUNCTION UNSTABLE*)
        C4:=RFUNC(M+C3*STEP,T0+STEP,TW,PR,MU,KEY);
        IF C4 ( 0 THEN C4:=0;(*FUNCTION UNSTABLE*)
        DELM:=STEP/6.0*(C1+2.0*C2+2.0*C3+C4);
        M:=M+DELM;
        T0:=T0+STEP;
        I:=I+1;
        IF M ) 1 THEN (*FUNCTION UNSTABLE, EXIT LOOP*)
            BEGIN
            TEST:=1;
            M:=1.0;
            END(*IF*)
        END;(* WHILE *)
        IF (C1 = 0) OR (C2 = 0) OR (C3 = 0) OR (C4 = 0) THEN
            (*FUNCTION UNSTABLE, EXIT LOOP*)
            BEGIN
            TEST:=1;
            M:=1;
            END;(*IF*)
    M2:=M;
    A:=R(M2)/R(M1);
    C:=(K+1.0)/(2.0*(K-1.0));
    P0:=P0*(K1/M2)*SQRT(T02/T01)*PWR(A,C);
    P:=PRES(P0,M2);
    T:=TEMP(T01,M2);
    RHO:=DENSE(P,T);
    RE:=4.0*MDOT/(PI*D*MU(T));
    IF RE (= 2000 THEN
        BEGIN (* LAMINAR FLOW *)
        F:=16.0/RE;
        MU:=3.66;
        END(*IF*)
    ELSE
        BEGIN (* TURBULENT FLOW *)
        F:=0.046/PWR(RE,0.2);
        MU:=0.023*PWR(RE,0.8)*PWR(PR,0.4);
        END;(*ELSE*)
    Q:=CP*MDOT*(T02-T01);
    QX:=QX+Q;
```

```
    END;(* HEATTRANS *)


PROCEDURE SECTION(NWALL:INTEGER; VAR NDIV:INTEGER; LS,LW:COUNT; VAR X:DIVISIONS);
(* THIS PROCEDURE DIVIDES THE PIPE LENGTHS INTO NONOVERLAPPING SECTIONS *)
    VAR
        RESL,LTOT,L:REAL;
        COUNT1,COUNT2,I,J,KEYINTEGER;

    BEGIN (* SECTION *)
    (* CALCULATE NOMINAL SECTION SIZE *)
    LTOT:=0;
    FOR I:=1 TO NWALL DO LTOT:=LTOT+LW[I];
    FOR I:=1 TO NWALL+1 DO LTOT:=LTOT+LS[I];
    L:=LTOT/NUMDIV;
    (* CALCULATE PIPE SECTIONS *)
    LTOT:=0;
    X[0]:=0;
    J:=0;
    KEY:=0;
    COUNT1:=1;
    COUNT2:=1;
    FOR I:=1 TO 2*NWALL+1 DO
        BEGIN
        KEY:=KEY+1;
        IF KEY = 1 THEN
            BEGIN
            LTOT:=LTOT+LS[COUNT1];
            COUNT1:=COUNT1+1;
            END(*IF*)
        ELSE
            BEGIN
            LTOT:=LTOT+LW[COUNT2];
            COUNT2:=COUNT2+1;
            KEY:=0;
            END;(*ELSE*)
        WHILE X[J] < LTOT DO
            BEGIN
            J:=J+1;
            X[J]:=X[J-1]+L;
            END;(*WHILE*)
        IF X[J] > LTOT THEN
            BEGIN
            RESL:=LTOT-X[J-1];
            X[J]:=X[J-1]+RESL;
            END;(*IF*)
        END;(*FOR*)
    NDIV:=J;
    X[NDIV+1]:=LTOT+1;
    END;(*SECTION*)



PROCEDURE SETBOUND(KEY:INTEGER; VAR HIB,LOWB,MDOT:REAL);
    BEGIN(*SETBOUND*)
    IF KEY = 1 THEN
        BEGIN
        LOWB:=MDOT;
        END(*IF*)
    ELSE
        BEGIN
        HIB:=MDOT;
        END;(*ELSE*)
    MDOT:=(2.0*LOWB+HIB)/3.0;
    END;(*SETBOUND*)



PROCEDURE INREASSIGN;
    (*DEFINES VALUES FOR USE IN PARAMETER LIST *)
    BEGIN(*INREASSIGN*)
    L:=X[I]-X[I-1];
    M1:=M[I];
    T01:=T0[I];
    P01:=P0[I];
    T1:=T[I];
    P1:=P[I];
    RHO1:=RHO[I];
```

```
      F1:=F[I];
      RE1:=RE[I];

      MU1:=MU[I];
      Q1:=Q[I];
      QX1:=QX[I];
      END;(*INREASSIGN*)


PROCEDURE OUTREASSIGN;
    (* DEFINES NEW VALUES OBTAINED FROM PARAMETER LIST *)
    BEGIN(*OUTREASSIGN*)
    I:=I+1;
    M[I]:=M1;
    T0[I]:=T01;
    P0[I]:=P01;
    T[I]:=T1;
    P[I]:=P1;
    RHO[I]:=RHO1;
    F[I]:=F1;
    RE[I]:=RE1;
    MU[I]:=MU1;
    Q[I]:=Q1;
    QX[I]:=QX1;
    END;(*OUTREASSIGN*)


PROCEDURE INITIALIZE;
    (* PREPARES VARIABLES FOR USE IN MAIN PROGRAM *)
    VAR
        J:INTEGER;
        A,C:REAL;
    BEGIN(*INITIALIZE*)
    MDOT[0]:=0;
    (*DETERMINE MAXIMUM MASS FLOW*)
    A:=1.0+0.405*(K-1.0);
    C:=(K+1.0)/(2.0*(1.0-K));
    MDOTMAX:=(PI*SQR(DIA)/4.0)*P0[1]*SQRT(K/(R*T0[1]))*PWR(A,C);
    (*CHECK GIVEN MASS FLOW AND REDUCE IF NECESSARY*)
    IF MDOT[1] > MDOTMAX THEN MDOT[1]:=0.7*MDOTMAX;
    (* CALCULATE TOTAL PIPE LENGTH *)
    LTOTAL:=0;
    FOR I:=1 TO NUMWALLS DO LTOTAL:=LTOTAL+LWALL[I];
    FOR I:=1 TO NUMWALLS+1 DO LTOTAL:=LTOTAL+LSPACE[I];
    (*CALCULATE SECTION LENGTHS*)
    (*INITIALIZE SECTION ARRAY*)
    SECTION(NUMWALLS,NUMDIV,LSPACE,LWALL,X);
    (*START LOOP COUNTER*)
    LOOPCOUNT:=0;
    (*INITIAL MACH NUMBER GUESS*)
    M[1]:=0.01;
    (*INITIALIZE END MACH NUMBER SO THAT IT MAY BE USED FOR TEST*)
    M[NUMDIV]:=0.0;
    (*INITIALIZE BOUNDS *)
    HIBOUND:=MDOTMAX;
    LOWBOUND:=0.0;
    END;(*INITIALIZE*)


PROCEDURE REINITIAL;
    (* PREPARES VARIABLES FOR USE IN MAIN PROGRAM *)
    VAR J:INTEGER;
    BEGIN(*REINITIAL*)
    (*INITIALIZE COUNTERS*)
    I:=1;
    PIPECNT:=1;
    WALLCNT:=1;
    SPACECNT:=1;
    (*INCREMENT LOOP COUNTER*)
    LOOPCOUNT:=LOOPCOUNT+1;
    (*CALCULATE ENTRANCE CONDITIONS*)
    LPIPE[1]:=LSPACE[1];
    QX[1]:=0;Q[1]:=0;
    RE[1]:=4.0*MDOT[LOOPCOUNT]/(PI*MU(T0[1])*DIA);
    IF RE[1] <= 2000 THEN
        BEGIN (* LAMINAR FLOW *)
        F[1]:=16.0/RE[1];
        MU[1]:=3.66;
        END(*IF*)
    ELSE
```

```
            BEGIN (* TURBULENT FLOW *)
            F[I]:=0.046/PWR(RE[I],0.2);
            NU[I]:=0.023*PWR(RE[I],0.8)*PWR(PRANDTL(T0[I]),0.4);
            END;(*ELSE*)
        (*CALCULATE INLET CONDITIONS*)
        INLETMACH(DIA,MDOT[LOOPCOUNT],P0[1],T0[1],M[1],T[1],P[1],RHO[1]);
        (*INITIALIZE CONTROL CARD*)
        MTEST:=1;
        END;(*REINITIAL*)


    BEGIN(* MAIN PROGRAM *)
        WRITELN('TO ENTER DATA, ENTER 1.  OTHERWISE,ENTER 0.');
        READLN;
        READ(TAB);
        INDATA(TAB);
        INITIALIZE;

        REPEAT (* UNTIL THE FLOW IS CHOKED *)
        REINITIAL;
        WHILE (I < NUMDIV) AND (M[I] (= 0.97) DO
            BEGIN
            (* BEGIN FRICTION FLOW CALCULATIONS *)
            WHILE (X[I] (= LPIPE[PIPECNT]) AND (M[I] (= 0.97) DO
                BEGIN
                INREASSIGN;
                FRICTION(T01,DIA,F1,L,M1,T1,P1,RHO1,P01,Q1);
                OUTREASSIGN;
                END;(*WHILE*)
            IF WALLCNT (= NUMWALLS THEN
                BEGIN
                PIPECNT:=PIPECNT+1;
                SPACECNT:=SPACECNT+1;
                LPIPE[PIPECNT]:=LPIPE[PIPECNT-1]+LWALL[WALLCNT];
                END;(*IF*)
            (*  BEGIN HEAT TRANSFER CALCULATIONS *)
            T0SAVE:=T0[I];
            EFF[WALLCNT]:=0;
            QSAVE:=QX[I];
            WHILE (EFF[WALLCNT] < MAXEFF) AND (X[I] < LPIPE[PIPECNT]) AND (M[I] (= 0.97) DO
                BEGIN
                INREASSIGN;
                HEATTRANS(TW[WALLCNT],L,DIA,MDOT[LOOPCOUNT],N,M1,T01,P01,T1,P1,RHO1,RE1,F1,Q1,QX1,NU1);
                OUTREASSIGN;
                EFF[WALLCNT]:=(T0[I]-T0SAVE)/(TW[WALLCNT]-T0SAVE);
                END;(*WHILE*)
            QSHIELD[WALLCNT]:=QX[I]-QSAVE;
            IF WALLCNT (= NUMWALLS THEN
                BEGIN
                PIPECNT:=PIPECNT+1;
                WALLCNT:=WALLCNT+1;
                LPIPE[PIPECNT]:=LPIPE[PIPECNT-1]+LSPACE[SPACECNT];
                END(*IF*)
            END;(*WHILE*)
        IF NUMDIV ) I THEN MTEST:=2;
        MDOT1:=MDOT[LOOPCOUNT];
        SETBOUND(MTEST,HIBOUND,LOWBOUND,MDOT1);
        MDOT[LOOPCOUNT+1]:=MDOT1;
        UNTIL (M[NUMDIV] ) 0.97) AND (M[NUMDIV] < 1.03) OR (LOOPCOUNT = 30);

        OUTDATA;
    10: END.(* MAIN PROGRAM *)
    /
```

```
 $INM.
TO ENTER DATA, ENTER 1.' OTHERWISE,ENTER 0.
? 1
INPUT DATA AS  PROMPTED BY TERMINAL

WHEN ASKED FOR A SERIES OF DATA SUCH AS THE
SHIELD TEMPERATURES, INPUT THE DATA IN
ORDER FROM THE DEWAR TO THE OUTER BOUNDARY

ENTER THE NUMBER OF RADIATION SHIELDS
? 5
NUMBER OF SHIELDS=     5

ENTER NUMBER OF SECTIONS FOR SOLUTION
? 125
NUMBER OF SOLUTION SECTIONS=   125

ENTER DIAMETER (M)
? 0.0016
DIAMETER (M)=       0.0016000

INPUT DISTANCES BETWEEN SHIELDS (M)
? 0.019
? 0.021
? 0.024
? 0.026
? 0.034
? 0.035
ENTER PIPE LENGTHS ALONG SHIELDS (M)
? 0.032
? 0.01
? 0.008
? 0.006
? 0.038
ENTER TEMPERATURES OF RADIATION SHIELDS (K)
? 21.7
? 39.7
? 65.4
? 101.1
? 146.6
ENTER MAXIMUM SHIELD EXCHANGER EFFECTIVENESS
? 0.98
MAXIMUM SHIELD EFFECTIVENESS=     0.980000000

ENTER INITIAL STAGNATION T (K)
? 4.23
INITIAL STAGNATION TEMPERATURE (K)=     4.230000000

ENTER INITIAL STAGNATION P (PA)
? 2100.0
INITIAL STAGNATION PRESSURE (PA)= 2100.000000000

ENTER MASS FLOW APPROXIMATION (KG/S)
? 0.00000163
MASS FLOW (KG/S)=     0.000001630

ENTER NUMBER OF STEPS IN RUNGE-KUTTA METHOD
? 4
NUMBER OF STEPS IN RUNGE-KUTTA=     4
FOR AN OUTPUT OF SECTION FLOW VARIABLES, INPUT 1. OTHERWISE INPUT 0.
? 1

FOR A PLOT FILE ENTER 1, OTHERWISE ENTER 0
? 0
FOR INTERATION INFORMATION, INPUT 1.  OTHERWISE INPUT 0.
? 1
OUTPUT STORED IN FILE EFFOUTM
     1.824 CP SECS, 101466B CM USED.
/
```

TYPE,EFFOUTH

LAMFLOW OUTPUT

VALUES OF INPUT

NUMBER OF SECTIONS = 120
DEVAR STAGNATION TEMPERATURE (K)= 4.23000000000E+000
DEVAR STAGNATION PRESSURE (PA)= 2.10000000000E+003
MASS FLOW APPROXIMATION (KG/S)= 1.63000000000E-006

SHIELD TEMPERATURES (K)
SHIELD TEMPERATURE 1= 21.700000000
SHIELD TEMPERATURE 2= 39.700000000
SHIELD TEMPERATURE 3= 65.400000000
SHIELD TEMPERATURE 4= 101.100000000
SHIELD TEMPERATURE 5= 146.400000000

VALUES OF OUTPUT

MASS FLOW (KG/S)= 0.000000858
TOTAL HEAT TRANSFER (W)= 0.630151115

SHIELD HEAT TRANSFER (W)
SHIELD HEAT (W) 1= 0.065715165
SHIELD HEAT (W) 2= 0.042899052
SHIELD HEAT (W) 3= 0.071678217
SHIELD HEAT (W) 4= 0.095742199
SHIELD HEAT (W) 5= 0.354116483

SHIELD HEAT EXCHANGER EFFECTIVENESS
SHIELD EFFECTIVENESS 1= 0.443247782
SHIELD EFFECTIVENESS 2= 0.463717702
SHIELD EFFECTIVENESS 3= 0.436381394
SHIELD EFFECTIVENESS 4= 0.380185874
SHIELD EFFECTIVENESS 5= 0.986241974

LENGTH	MACH NUMBER	STAG TEMP	STAG PRES	TEMP	PRESSURE	NUSSELT #	REYNOLD #	SECTION HEAT	TOTAL HEAT
0.00102	0.01473867	4.230000	2100.00000	4.2297	2099.619	3.660	429.58017	0.0000000	0.0000000
0.00405	0.01478766	4.230000	2093.04477	4.2297	2092.663	3.660	429.58017	0.0000000	0.0000000
0.00607	0.01483681	4.230000	2086.11226	4.2297	2085.729	3.660	429.58017	0.0000000	0.0000000
0.00810	0.01488614	4.230000	2079.20237	4.2297	2078.818	3.660	429.58017	0.0000000	0.0000000
0.01012	0.01493562	4.230000	2072.31506	4.2297	2071.929	3.660	429.58017	0.0000000	0.0000000
0.01214	0.01498528	4.230000	2065.45022	4.2297	2065.063	3.660	429.58017	0.0000000	0.0000000
0.01417	0.01503510	4.230000	2058.60788	4.2297	2058.219	3.660	429.58017	0.0000000	0.0000000
0.01619	0.01508509	4.230000	2051.78772	4.2297	2051.398	3.660	429.58017	0.0000000	0.0000000
0.01822	0.01513525	4.230000	2044.98990	4.2297	2044.599	3.660	429.58017	0.0000000	0.0000000
0.01900	0.01518558	4.230000	2038.21427	4.2297	2037.822	3.660	429.58017	0.0000000	0.0000000
0.02102	0.01523585	4.230000	2031.49128	4.2297	2031.098	3.660	429.58017	0.0000000	0.0000000
0.02305	0.01788757	5.828338	2031.22578	5.8277	2030.683	3.660	385.73793	0.0071299	0.0071299
0.02507	0.02009920	7.355642	2030.92014	7.3546	2030.135	3.660	351.78038	0.0068131	0.0139430
0.02710	0.02198954	8.800362	2030.57312	8.7989	2029.753	3.660	324.94689	0.0064447	0.0203877
0.02912	0.02362583	10.153898	2030.18400	10.1520	2029.238	3.660	303.43258	0.0060379	0.0264256
0.03114	0.02505172	11.410416	2029.75286	11.4082	2028.698	3.660	285.98168	0.0056058	0.0320314
0.03317	0.02629846	12.567686	2029.28046	12.5648	2028.109	3.660	271.69305	0.0051615	0.0371932
0.03519	0.02739014	13.624788	2028.76618	13.6214	2027.498	3.660	259.90346	0.0047154	0.0419087
0.03722	0.02834634	14.583734	2028.21784	14.5798	2026.858	3.660	250.11411	0.0042777	0.0461864
0.03924	0.02918360	15.448058	2027.62165	15.4436	2026.190	3.660	241.94298	0.0038556	0.0500420
0.04126	0.02991624	16.222553	2027.01205	16.2177	2025.498	3.660	235.09248	0.0034549	0.0534970
0.04329	0.03055680	16.912962	2026.36161	16.9877	2024.783	3.660	229.32831	0.0030798	0.0565768
0.04531	0.03111639	17.525545	2025.68296	17.5199	2024.044	3.660	224.46321	0.0027326	0.0593094
0.04734	0.03160489	18.066828	2024.97871	18.0608	2023.291	3.660	220.34627	0.0024146	0.0617240
0.04936	0.03203107	18.543362	2024.25138	18.5370	2022.518	3.660	216.85481	0.0021257	0.0638497
0.05100	0.03240272	18.961539	2023.50334	18.9549	2021.730	3.660	213.88828	0.0018654	0.0657152
0.05302	0.03247127	18.961539	2019.23756	18.9548	2017.461	3.660	213.88828	0.0000000	0.0657152
0.05505	0.03254168	18.961539	2014.87486	18.9548	2013.094	3.660	213.88828	0.0000000	0.0657152
0.05707	0.03261228	18.961539	2010.51919	18.9548	2008.735	3.660	213.88828	0.0000000	0.0657152
0.05910	0.03268307	18.961539	2006.17053	18.9548	2004.382	3.660	213.88828	0.0000000	0.0657152
0.06112	0.03275406	18.961539	2001.82884	18.9547	2000.037	3.660	213.88828	0.0000000	0.0657152
0.06314	0.03282524	18.961539	1997.49419	18.9547	1995.698	3.660	213.88828	0.0000000	0.0657152
0.06517	0.03289661	18.961539	1993.16647	18.9547	1991.367	3.660	213.88828	0.0000000	0.0657152
0.06719	0.03296818	18.961539	1988.84571	18.9546	1987.042	3.660	213.88828	0.0000000	0.0657152
0.06922	0.03303995	18.961539	1984.53187	18.9546	1982.724	3.660	213.88828	0.0000000	0.0657152
0.07124	0.03311192	18.961539	1980.22495	18.9546	1978.413	3.660	213.88828	0.0000000	0.0657152

0.07200	0.03318408	18.961539	1975.92494	18.9545	1974.109	3.660	213.88828	0.0000000	0.0657152
0.07402	0.03325054	18.961539	1971.98175	18.9545	1970.162	3.660	213.88828	0.0000000	0.0657152
0.07605	0.03562057	21.732703	1970.91570	21.7235	1968.829	3.660	196.24027	0.0123617	0.0780769
0.07807	0.03766901	24.269389	1969.69965	24.2578	1967.368	3.660	182.68396	0.0113154	0.0893923
0.08010	0.03943231	26.552986	1968.33701	26.5392	1965.784	3.660	172.11265	0.0101871	0.0995795
0.08208	0.04094309	28.578338	1966.83487	28.5423	1964.085	3.660	163.88802	0.0090346	0.1086142
0.08402	0.04101142	28.578338	1963.55595	28.5622	1960.801	3.660	163.88802	0.0000000	0.1086142
0.08605	0.04108186	28.578338	1960.20638	28.5622	1957.447	3.660	163.88802	0.0000000	0.1086142
0.08807	0.04115231	28.578338	1956.85817	28.5621	1954.094	3.660	163.88802	0.0000000	0.1086142
0.09010	0.04122297	28.578338	1953.51154	28.5621	1950.742	3.660	163.88802	0.0000000	0.1086142
0.09212	0.04129384	28.578338	1950.16642	28.5620	1947.392	3.660	163.88802	0.0000000	0.1086142
0.09414	0.04136492	28.578338	1946.82279	28.5620	1944.044	3.660	163.88802	0.0000000	0.1086142
0.09617	0.04143622	28.578338	1943.48064	28.5619	1940.697	3.660	163.88802	0.0000000	0.1086142
0.09819	0.04150773	28.578338	1940.13996	28.5618	1937.352	3.660	163.88802	0.0000000	0.1086142
0.10022	0.04157946	28.578338	1936.80076	28.5618	1934.008	3.660	163.88802	0.0000000	0.1086142
0.10224	0.04165140	28.578338	1933.46301	28.5617	1930.665	3.660	163.88802	0.0000000	0.1086142
0.10426	0.04172357	28.578338	1930.12672	28.5617	1927.324	3.660	163.88802	0.0000000	0.1086142
0.10629	0.04179595	28.578338	1926.79187	28.5616	1923.984	3.660	163.88802	0.0000000	0.1086142
0.10802	0.04186469	28.578338	1923.63548	28.5616	1920.823	3.660	163.88802	0.0000000	0.1086142
0.11005	0.04602557	34.446317	1921.45571	34.4219	1918.061	3.660	144.11636	0.0261762	0.1347904
0.11207	0.04958113	39.845476	1918.80013	39.8127	1914.867	3.660	130.17586	0.0240848	0.1588752
0.11400	0.05257984	44.646622	1915.66882	44.6053	1911.254	3.660	120.14158	0.0214172	0.1802924
0.11602	0.05270749	44.646622	1911.04652	44.6051	1906.621	3.660	120.14158	0.0000000	0.1802924
0.11805	0.05283932	44.646622	1906.29424	44.6049	1901.859	3.660	120.14158	0.0000000	0.1802924
0.12007	0.05297186	44.646622	1901.54437	44.6047	1897.096	3.660	120.14158	0.0000000	0.1802924
0.12210	0.05310511	44.646622	1896.79087	44.6045	1892.332	3.660	120.14158	0.0000000	0.1802924
0.12412	0.05323908	44.646622	1892.03571	44.6043	1887.565	3.660	120.14158	0.0000000	0.1802924
0.12614	0.05337378	44.646622	1887.27886	44.6041	1882.797	3.660	120.14158	0.0000000	0.1802924
0.12817	0.05350921	44.646622	1882.52028	44.6038	1878.027	3.660	120.14158	0.0000000	0.1802924
0.13019	0.05364539	44.646622	1877.75994	44.6036	1873.255	3.660	120.14158	0.0000000	0.1802924
0.13222	0.05378231	44.646622	1872.99779	44.6034	1868.482	3.660	120.14158	0.0000000	0.1802924
0.13424	0.05391999	44.646622	1868.23381	44.6032	1863.706	3.660	120.14158	0.0000000	0.1802924
0.13626	0.05405843	44.646622	1863.46795	44.6030	1858.929	3.660	120.14158	0.0000000	0.1802924
0.13829	0.05419764	44.646622	1858.70018	44.6027	1854.149	3.660	120.14158	0.0000000	0.1802924
0.14008	0.05433763	44.646622	1853.93046	44.6025	1849.368	3.660	120.14158	0.0000000	0.1802924
0.14202	0.05446584	44.646622	1849.58354	44.6023	1845.010	3.660	120.14158	0.0000000	0.1802924
0.14405	0.06113698	55.909883	1844.87728	55.8460	1839.132	3.660	102.45910	0.0502437	0.2305361
0.14600	0.06673151	66.109399	1838.80167	66.0109	1831.982	3.660	91.05174	0.0454985	0.2760346
0.14802	0.06699717	66.109399	1831.55385	66.0101	1824.707	3.660	91.05174	0.0000000	0.2760346
0.15005	0.06727264	66.109399	1824.09879	66.0093	1817.224	3.660	91.05174	0.0000000	0.2760346
0.15207	0.06755094	66.109399	1816.62923	66.0085	1809.726	3.660	91.05174	0.0000000	0.2760346
0.15410	0.06783212	66.109399	1809.14496	66.0077	1802.213	3.660	91.05174	0.0000000	0.2760346
0.15612	0.06811622	66.109399	1801.64575	66.0068	1794.685	3.660	91.05174	0.0000000	0.2760346
0.15814	0.06840329	66.109399	1794.13138	66.0059	1787.141	3.660	91.05174	0.0000000	0.2760346
0.16017	0.06869340	66.109399	1786.60163	66.0051	1779.581	3.660	91.05174	0.0000000	0.2760346
0.16219	0.06898660	66.109399	1779.05626	66.0042	1772.006	3.660	91.05174	0.0000000	0.2760346
0.16422	0.06928295	66.109399	1771.49504	66.0033	1764.415	3.660	91.05174	0.0000000	0.2760346
0.16624	0.06958250	66.109399	1763.91772	66.0023	1756.807	3.660	91.05174	0.0000000	0.2760346
0.16826	0.06988531	66.109399	1756.32406	66.0014	1749.182	3.660	91.05174	0.0000000	0.2760346
0.17029	0.07019146	66.109399	1748.71381	66.0005	1741.540	3.660	91.05174	0.0000000	0.2760346
0.17231	0.07050099	66.109399	1741.08671	65.9995	1733.882	3.660	91.05174	0.0000000	0.2760346
0.17434	0.07081397	66.109399	1733.44250	65.9985	1726.205	3.660	91.05174	0.0000000	0.2760346
0.17636	0.07113048	66.109399	1725.78092	65.9975	1718.512	3.660	91.05174	0.0000000	0.2760346
0.17838	0.07145058	66.109399	1718.10170	65.9965	1710.800	3.660	91.05174	0.0000000	0.2760346
0.18000	0.07177433	66.109399	1710.40457	65.9955	1703.069	3.660	91.05174	0.0000000	0.2760346
0.18202	0.07205161	66.109399	1703.86744	65.9946	1696.504	3.660	91.05174	0.0000000	0.2760346
0.18405	0.08278777	86.078887	1693.99074	85.8817	1684.335	3.660	75.89737	0.0890809	0.3651155
0.18607	0.09162007	103.554686	1680.61638	103.2643	1668.894	3.660	67.10205	0.0779569	0.4430724
0.18810	0.09840659	117.378678	1663.97609	116.9976	1650.543	3.660	61.89685	0.0616667	0.5047391
0.19012	0.10404068	127.456459	1644.57728	126.9959	1627.807	3.660	58.77742	0.0449555	0.5496946
0.19214	0.10831156	134.371112	1622.99295	133.8451	1607.203	3.660	56.89448	0.0308452	0.5805398
0.19417	0.11178771	138.919605	1599.73146	138.3405	1583.160	3.660	55.75387	0.0202901	0.6008299
0.19619	0.11476388	141.829058	1575.18431	141.2060	1557.994	3.660	55.06225	0.0129786	0.6138086
0.19822	0.11745487	143.656902	1549.62479	142.9960	1531.917	3.660	54.64333	0.0081537	0.6219623
0.20024	0.12000984	144.792292	1523.22924	144.0971	1505.864	3.660	54.39047	0.0050648	0.6270271
0.20226	0.12253069	145.492608	1496.10248	144.7645	1477.518	3.660	54.23891	0.0031240	0.6301511
0.20429	0.12488703	145.492608	1468.44279	144.7364	1449.492	3.660	54.23891	0.0000000	0.6301511
0.20631	0.12738408	145.492608	1440.25988	144.7060	1420.929	3.660	54.23891	0.0000000	0.6301511
0.20834	0.13003653	145.492608	1411.52143	144.6731	1391.787	3.660	54.23891	0.0000000	0.6301511
0.21036	0.13286134	145.492608	1382.19170	144.6373	1362.027	3.660	54.23891	0.0000000	0.6301511
0.21238	0.13591128	145.492608	1351.90952	144.5978	1331.281	3.660	54.23891	0.0000000	0.6301511
0.21441	0.13918265	145.492608	1320.92227	144.5545	1299.796	3.660	54.23891	0.0000000	0.6301511
0.21643	0.14270385	145.492608	1289.17683	144.5068	1267.514	3.660	54.23891	0.0000000	0.6301511
0.21800	0.14650870	145.492608	1256.61308	144.4539	1234.370	3.660	54.23891	0.0000000	0.6301511
0.22002	0.14968950	145.492608	1230.67897	144.4086	1207.952	3.660	54.23891	0.0000000	0.6301511
0.22205	0.15414059	145.492608	1196.21187	144.3437	1172.806	3.660	54.23891	0.0000000	0.6301511
0.22407	0.15902029	145.492608	1160.67863	144.2785	1136.529	3.660	54.23891	0.0000000	0.6301511
0.22610	0.16440293	145.492608	1123.97137	144.1871	1099.081	3.660	54.23891	0.0000000	0.6301511
0.22812	0.17041765	145.492608	1085.74689	144.0987	1059.858	3.660	54.23891	0.0000000	0.6301511
0.23014	0.17716113	145.492608	1046.03785	143.9788	1019.119	3.660	54.23891	0.0000000	0.6301511

0.23217	0.18479595	145.492608	1084.65232	143.8470	976.567	3.660	54.23891	0.0000000	0.6301511
0.23419	0.19354111	145.492608	961.35274	143.4895	931.931	3.660	54.23891	0.0000000	0.6301511
0.23622	0.20370017	145.492608	915.83922	143.4979	884.862	3.660	54.23891	0.0000000	0.6301511
0.23824	0.21570889	145.492608	867.72364	143.2595	836.987	3.660	54.23891	0.0000000	0.6301511
0.24026	0.23022073	145.492608	816.48883	142.9544	781.445	3.660	54.23891	0.0000000	0.6301511
0.24229	0.24827182	145.492608	761.41943	142.5491	723.601	3.660	54.23891	0.0000000	0.6301511
0.24431	0.27163139	145.492608	701.47791	141.9831	660.859	3.660	54.23891	0.0000000	0.6301511
0.24634	0.30364898	145.492608	635.06363	141.1333	588.691	3.660	54.23891	0.0000000	0.6301511
0.24836	0.35172387	145.492608	559.50349	139.7038	505.646	3.660	54.23891	0.0000000	0.6301511
0.25038	0.43696755	145.492608	469.96922	136.7457	402.672	3.660	54.23891	0.0000000	0.6301511
0.25241	0.66028718	145.492608	360.66127	126.9510	256.761	3.660	54.23891	0.0000000	0.6301511
0.25300	1.00000000	145.492608	322.76830	108.9832	157.881	3.660	54.23891	0.0000000	0.6301511

INTERATION= 1 MASS FLOW= 1.630000E-006
INTERATION= 2 MASS FLOW= 5.433333E-007
INTERATION= 3 MASS FLOW= 9.055556E-007
INTERATION= 4 MASS FLOW= 6.440741E-007
INTERATION= 5 MASS FLOW= 7.445679E-007
INTERATION= 6 MASS FLOW= 7.982305E-007
INTERATION= 7 MASS FLOW= 8.340055E-007
INTERATION= 8 MASS FLOW= 8.578555E-007

APPENDIX B: CALCULATION OF INLET MACH NUMBER

The stagnation conditions of the dewar are equivalent to the entrance stagnation conditions. The continuity equation for one-dimensional flow states that

$$\dot{m} = \rho VA. \tag{B.1}$$

The definition of Mach number is

$$M = V/[(\gamma RT)^{1/2}] \tag{B.2}$$

Combined with (B.1), this yields

$$\dot{m} = (\gamma RT)^{1/2} \rho MA \tag{B.3}$$

The density may be expressed in terms of the stagnation density and the Mach number.

$$\rho = \rho_0\{1 + [(\gamma - 1)/2] M^2\}^{-1/(\gamma-1)} \tag{B.4}$$

The ideal gas equation states

$$\rho_0 = P_0/RT_0 \tag{B.5}$$

Combining Eqs. (B.4) and (B.5) yields

$$\rho = [P_0/(RT_0)][1 + (\gamma - 1)/2\ M^2]^{-1/(\gamma-1)} \tag{B.6}$$

Similarly, T may be expressed in terms of M and T_0

$$T = T_0\{1 + (\gamma - 1)/2] M^2\}^{-1} \tag{B.7}$$

Combining Eqs. (B.3), (B.6), and (B.7) yields

$$\dot{m} = (\gamma/RT_0)\ AMP_0\{1 + [(\gamma - 1)/2] M^2\}^{\{[\gamma+1]/[2(1-\gamma)]\}} \tag{B.8}$$

We wish to solve for the Mach number. Since the equation is nonlinear in M, the Newton-Raphson method of solution is invoked. Starting from some initial condition, new values of M are calculated iteratively until convergence is obtained. Using the expression

$$M_{new} = M_{old} - [f(M_{old})]/[f'(M_{old})] \tag{B.9}$$

where

$$f(M) = (\gamma/RT_0)^{1/2}\ AMP_0\{1 + [(\gamma - 1)/2] M^2\}^{\{[\gamma+1]/2[1-\gamma]\}} - \dot{m} \tag{B.10}$$

and

$$f'(M) = AP_o(\gamma/RT_o)^{1/2} B^{\{[\gamma+1]/2[(1-\gamma]\}}[1 - M^2[(\gamma+1)/2] B^{-1}\} \quad (B.11)$$

B is an often used function of M and γ.

$$B = 1 + [(\gamma - 1)/2] M^2 \quad (B.12)$$

APPENDIX C: FRICTION FLOW MACH NUMBER CALCULATION

The differential equation relating the change in Mach number to the change in distance is

$$dM/M = \{[\gamma M^2 B]/[2(1 - M^2)]\}[(4f\, dx)/D] \qquad (C.1)$$

Solving with the imposed boundary conditions of $M = M_1$ at $x = x_1$ and $M = M_2$ at $x = x_2$ yields

$$[4f(x_2 - x_1)]/D = \{(M_2^2 - M_1^2)/(\gamma M_1^2 M_2^2) + [(\gamma + 1)/2\gamma] * \ln[(M_1^2 B_2)/(M_2^2 B_1)]\} \qquad (C.2)$$

Solve for M_2 with the Newton-Raphson method using

$$f(M_2) = [(M_2^2 - M_1^2)/(\gamma M_1^2 M_2^2)] + [(\gamma+1)/2\gamma] \ln[(M_1^2 B_2)/(M_2^2 B_1)] - 4fL/D \qquad (C.3)$$

$$f'(M_2) = [2/(\gamma M_2^3) + \{[(\gamma+1) M_2]/\gamma\}[B_1^2/(M_1^4 B_2^3)]] \qquad (C.4)$$

where $L = x_2 - x_1$.

APPENDIX D: STAGNATION TEMPERATURE CALCULATION FROM AN ENERGY BALANCE

The heat transferred to a fluid as it travels through a pipe within isothermal walls is

$$\dot{m}\, dq = h(T_w - T_{aw})\, \pi D\, dx \qquad (D.1)$$

The differential change in heat of an ideal compressible gas is a function of the stagnation temperature.

$$dq = c_p\, dT_o \qquad (D.2)$$

Combining Eqs. (D.1) and (D.2) yields

$$\dot{m}\, c_p\, dT_o = h(T_w - T_{aw})\, \pi D\, dx \qquad (D.3)$$

The adiabatic wall temperature of a laminar flow is equal to the stagnation temperature. For turbulent flow, the slope of the temperature profile of a fully developed fluid is too steep to apply a recovery factor; therefore, the adiabatic wall temperature will be assumed to be also equal to the stagnation temperature for turbulent fluid. Equation (D.3) may then be expressed as

$$\dot{m}\, c_p\, dT_o = h(T_w - T_o)\, \pi\, D\, dx \qquad (D.4)$$

One may recall that the Nusselt number is defined as

$$Nu = hD/k_f \qquad (D.5)$$

Equation (D.4) may be then expressed as $\dot{m}\, c_p\, dT_o = \pi\, Nu\, k_f (T_w - T_o)\, dx$.

$$dT_o/(T_w - T_o) = (\pi\, Nu\, k_f)/(\dot{m}\, c_p)\, dx \qquad (D.6)$$

Integrating Eq. (D.6) with the boundary conditions

$T_o = T_{o1}$ at $x = x_1$ and $T_o = T_{02}$ at $x = x_2$ yields

$$T_{02} = T_w - \{[T_w - T_{01}]/[\exp(\pi\, Nu\, k_f\, L/\dot{m}\, c_p)]\} \qquad (D.7)$$

The exit stagnation temperature of a length L of pipe is therefore a function of known quantitites.

APPENDIX E: HEAT TRANSFER WITH FRICTION

The change in Mach number in a pipe is a function of the change in length and the change in stagnation temperature.

$$\frac{dM}{M} = \frac{\gamma M^2 B}{2(1-M^2)} \frac{4f\,dx}{D} + \frac{(1+\gamma M^2)B}{2(1-M^2)} \frac{dT_o}{T_o} \tag{E.1}$$

From the energy balance in APPENDIX D, we have

$$\dot{m} C_p dT_o = \rho V \pi/4 D^2 C_p dT_o = h(T_w - T_o) \pi D dx \tag{E.2}$$

One may express these relations in terms of dimensionless parameters.

$$dT_o/(T_w - T_o) = [4Nu_D/Pr(Re_D)]\, dx/D \tag{E.3}$$

For laminar flow,

$$Re_D = 16/f \tag{E.4}$$

$$[dT_o/(T_w - T_o)] = [4\,Nu_D]/[Pr(16/f)]\, dx/D \tag{E.5}$$

$$(4f\,dx/D) = (16\,Pr/Nu_D)[dT_o/(T_w - T_o)] \tag{E.6}$$

For laminar flow the differential equation is then,

$$\frac{dM}{M} = \frac{\gamma M^2 B}{2(1-M^2)} \frac{16\,Pr}{Nu_D} \frac{dT_o}{T_w - T_o} + \frac{(1+\gamma M^2) B}{2(1-M^2)} \frac{dT_o}{T_o} \tag{E.7}$$

For turbulent flow, one may invoke the Reynolds analogy

$$St = f/2 \tag{E.8}$$

and

(E.9)

$$dT_o/(T_w - T_o) = 4(f/2)\, dx/D \tag{E.10}$$

So that for turbulent flow,

$$\frac{dM}{M} = \frac{\gamma M^2 B}{2(1-M^2)} \frac{2\,dT_o}{T_w - T_o} \frac{(1+\gamma M^2) B}{2(1-M^2)} \frac{dT_o}{T_o} \tag{E.11}$$

Equations (E.7) and (E.11) are solved numerically with the Runge-Kutta method, using about four iterations, as shown in APPENDIX F.

APPENDIX F: RUNGE-KUTTA SOLUTION

The basic Runge-Kutta equation is rather simple.

$$M_{n+1} = M_n + \Delta T_o/6(c_1 + 2c_2 + 2c_3 + c_4) \qquad (F.1)$$

where

$$c_1 = F(T_{on}, M_n)$$

$$c_2 = F(T_{on} + \Delta T_o/2,, M_n + c_1 \Delta T_o/2)$$

$$c_3 = F(T_{on} + \Delta T_o/2, M_n + c_2 \Delta T_o/2)$$

$$c_4 = F(T_{on} + \Delta T_o, M_n + c_3 \Delta T_o) \qquad (F.2)$$

The function F is simply a form of the differential equation we wish to solve for

$$F(T_{on}, M_n) = dM/dT_o|_n \qquad (F.3)$$

For laminar flow,

$$F = \frac{dM}{dT_o} = \frac{MB}{2(1 - M^2)} \left[\gamma M^2 \left(\frac{16 Pr}{Nu_D (T_w - T_o)}\right) + \frac{1 + \gamma M^2}{T_o}\right] \qquad (F.4)$$

For turbulent flow,

$$F = \frac{dM}{dT_o} = \frac{BM}{2(1 - M^2)} \left[\gamma M^2 \left(\frac{2}{T_w - T_o}\right) + \frac{1 + \gamma M^2}{T_o}\right] \qquad (F.5)$$

APPENDIX G: FLUID PROPERTIES

All curve fits shown below were performed by a standard IMSL Math Library routine on the University of Illinois Cyber 175. They are valid only in a temperature range of 4.2 K to 350 K.

Thermal conductivity:

$$k_f = 0.0\ 157218071 + 6.3416525*10^{-4}*T - 6.99407521*10^{-7}*T^2 + 4.52003096*10^{-10}*T^3 \qquad (G.1)$$

Viscosity:

$$\mu = 1.10419265531*10^{-6} + 1.160677594648*10^{-7}*T - 3.222922924062*10^{-10}*T^2 + 4.725700293922*10^{-13}*T^3 \qquad (G.2)$$

Prandtl number:

$$Pr = 0.570232994 + 0.0055872291*T - 7.58324532*10^{-5}*T^2 + 4.32720501*10^{-7}*T^3 - 1.11661445*10^{-9}*T^4 + 1.07403542*10^{-12}*T^5 \qquad (G.3)$$

APPENDIX H: MACH NUMBER CALCULATION WITH AREA CHANGE

In most texts, the area ratio is given in terms of the area at the throat, A*. The following merely algebraically solves for some arbitrary A_2.

$$A_1/A^* = \{1/M_1\} \{[2/(\gamma + 1)] B_1\}^{\{[\gamma+1]/[2(\gamma-1)]\}} \quad (H.1)$$

$$A_2/A^* = \{1/M_2\} \{[2/(\gamma + 1)] B_2\}^{\{[\gamma+1]/[2(\gamma-1)]\}} \quad (H.2)$$

$$A_2/A_1 = \{M_1/M_2\} (B_2/B_1)^{\{[\gamma+1]/[2(\gamma-1)]\}} \quad (H.3)$$

Solve for μ_2 with a Newton-Raphson method.

$$f(M_2) = \{M_1/M_2\} (B_2/B_1)^{\{[\gamma+1]/[2(\gamma-1)]\}} - A_2/A_1 \quad (H.4)$$

$$f'(M_2) = M_1 \left(\frac{B_2}{B_1}\right)^{\{[\gamma+1]/[2(\gamma-1)]\}} \left[\frac{\gamma + 1}{2} \left(\frac{B_1}{B_2}\right) - \frac{1}{M_2^2}\right] \quad (H.5)$$