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.

Produced by the NASA Center for Aerospace Information (CASI)
NUMERICAL CALCULATION OF THE PARAMETERS OF THE EFFLUX FROM A HELIUM DEWAR USED FOR COOLING OF HEAT SHEIELDS IN A SATELLITE

BY

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

July 1982
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
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>2. THEORETICAL CONSIDERATIONS</td>
<td>2</td>
</tr>
<tr>
<td>APPENDIX A: INPUTS AND OUTPUTS OF EFFLUXD AND EFFLUXM</td>
<td>10</td>
</tr>
<tr>
<td>APPENDIX B: CALCULATION OF INLET MACH NUMBER</td>
<td>41</td>
</tr>
<tr>
<td>APPENDIX C: FRICTION FLOW MACH NUMBER CALCULATION</td>
<td>43</td>
</tr>
<tr>
<td>APPENDIX D: STAGNATION TEMPERATURE CALCULATION FROM AN ENERGY BALANCE</td>
<td>44</td>
</tr>
<tr>
<td>APPENDIX E: HEAT TRANSFER WITH FRICTION</td>
<td>45</td>
</tr>
<tr>
<td>APPENDIX F: RUNGE-KUTTA SOLUTION</td>
<td>47</td>
</tr>
<tr>
<td>APPENDIX G: FLUID PROPERTIES</td>
<td>48</td>
</tr>
<tr>
<td>APPENDIX H: MACH NUMBER CALCULATION WITH AREA CHANGE</td>
<td>49</td>
</tr>
</tbody>
</table>
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_0  stagnation pressure
Pr  Prandtl number
Q_w  shield heat exchange
q  heat flux
Re_D  Reynolds number
St  Stanton number
T  temperature
T_0  stagnation temperature
T_w  wall (shield) temperature
T_{aw}  adiabatic wall temperature
V  velocity
x  distance

Greek Letters
\gamma  specific heat ratio
\epsilon  shield effectiveness
\rho  density
\rho_0  stagnation density
\mu  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.

\[ e = \frac{(T_{out} - T_{in})}{(T_w - T_{in})} \]  \hspace{1cm} (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.

\[ \dot{m}_j = \frac{\sum_{i=1}^{j} q_{wi}}{[C_p(T_{oj} - T_{in})]} \]  \hspace{1cm} (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 4

Viscosity versus Temperature

- Helium Data
- 3rd Order Fit

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 \textsc{EFFLUXD} and \textsc{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.
PROGRAM EFLUX (INPUT, OUTPUT, EFFIND1, EFFIND2, POUTD);
PROCEDURE INDAT(A(INDEX:INTEGER);
BEGIN (* INTERACTIVE INPUT *)
  IF INDEX = 1 THEN
    BEGIN
      (* READ IN DATA *)
      WRITE('INPUT DATA AS PROMPTED BY TERMINAL');
      WRITE('');
      WRITE('WHEN ASKED FOR A SERIES OF DATA SUCH AS THE SHIELD');
      WRITE('TEMPERATURES, INPUT THE DATA IN THE ORDER FROM DEVAR');
      WRITE('TO THE OUTER BOUNDARY.');
      WRITE('');
      WRITE('ENTER THE NUMBER OF RADIATION SHIELDS');
      READ(NWALLS);
      WRITE(EFFIND1,NWALLS);
      WRITE('NUMBER OF SHIELDS=',NWALLS:3);
      WRITE('');
      WRITE('ENTER SINGLE DIVISION LENGTH (M)');
      READ(LDIV);
      WRITE(EFFIND1,LDIV);
      WRITE('DIVISION LENGTH (M)=',LDIV:15:5);
      WRITE('');
      WRITE('ENTER PIPE LENGTHS NOT ON SHIELDS (M)');
      FOR I:=1 TO NWALLS+1 DO
        BEGIN
          READ(LSPACE(I));
          WRITE(EFFIND1,LSPACE(I));
        END;
      WRITE('ENTER TEMPERATURES OF RADIATION SHIELDS (R)');
      FOR I:=1 TO NWALLS DO
        BEGIN
          READ(TVII); WRITE(EffIND1,TVII);
        END;
      WRITE('ENTER HEAT INPUTS OF RADIATION SHIELDS (W)');
      FOR I:=1 TO NWALLS DO
        BEGIN
          READ(OVEII); WRITE(EffIND1,OVEII);
        END;
      WRITE('ENTER MAXIMUM HEAT EXCHANGER EFFECTIVENESS AS A FRACTION');
      READ(M1EFF);
      WRITE(EffIND1,M1EFF);
      WRITE('SHIELD EFFECTIVENESS=',M1EFF:15:7);
      WRITE('');
      WRITE('ENTER INITIAL STAGNATION T (K)');
      READ(TOEII);
      WRITE(EffIND1,TOEII);
      WRITE('INITIAL STAGNATION TEMPERATURE (K)=',TOEII:15:5);
      WRITE('');
      WRITE('ENTER INITIAL STAGNATION P (Pa)');
      READ(P0II);
      WRITE(EffIND1,P0II);
      WRITE('INITIAL STAGNATION PRESSURE (Pa)=',P0II:15:5);
      WRITE('');
      WRITE('ENTER NUMBER OF STEPS IN RUNGE-KUTTA METHOD');
      READ(H);
      WRITE(EffIND2,H);
      WRITE('STEPS IN RUNGE-KUTTA=',H:5);
    END;
  ELSE
    (* READ FILE *)
    BEGIN
      RESET(EffIND1);
    END;
  END;
PROCEDURE OUTDATA;
BEGIN
  WRITE(OUTFO, 'VALUES OF INPUT': 40);
  WRITE(OUTFO, 'VALUES OF OUTPUT': 40);
  WRITE(OUTFO, 'VALUES OF SECTION FLOW VARIABLES, INPUT 1.

  READ(F); IF P111 = 1 THEN
    BEGIN
      WRITE(OUTFO, 'LENGTH': 7, 'MACH NUMBER': 14, 'STAG TEMP': 14, 'STAG PRES': 14);
      WRITE(OUTFO, 'TEMP': 10, 'PRESSURE': 10, 'NUSSELT': 10, 'REYNOLD NUM': 15);
      WRITE(OUTFO, 'SECTION HEAT': 15, 'TOTAL HEAT': 15);
      FOR I := 1 TO NUMDIV DO
        BEGIN
          WRITE(OUTFO, 'SHEILD HEAT TRANSFER': 40);
          FOR I := 1 TO NUMVALLS DO
            BEGIN
              WRITE(OUTFO, 'SHIELD HEAT (V)': 71, '=': 1, OVALL[I]: 15: 11);
            END;
          END;
        END
      END;
    END;
  END;
END;(* INDATA *)
FUNCTION WK(P:REAL):REAL;
(* SOLVES FOR HKP *)
VAR A,REAL;
BEGIN
A:=F(K(P));
WK:=EXP(A);
END;(* WK *)

FUNCTION R(M:REAL):REAL;
(* SOLVES OBTAINED FUNCTION *)
BEGIN
R:=A*(1-E)*SQR(M)/2.8;
END;(* R *)

FUNCTION TEMP(TO,M:REAL):REAL;
(* CALCULATES LOCAL TEMPERATURE *)
BEGIN
TEMP:=TO/(1-E);
END;(* TEMP *)

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

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

FUNCTION RUT(R:REAL):REAL;
VAR A,B,C,D:REAL;
BEGIN(* RUT *)
A:=1.04191463318-06;
B:=1.857375534E-07;
C:=3.87827383402E-08;
END;(* RUT *)
FUNCTION COND(T:REAL):REAL;
VAR A,B,C,D:REAL;
BEGIN(*COND FUNCTION, CALCULATION*)
A:=6.73570437*1322E-13*T^2+SOR(T);
B:=A*B+C*D;
END(*COND*)

FUNCTION FRAUQTL(T:REAL):REAL;
VAR A,B,C,D,E,F:REAL;
BEGIN(*FRAQTL FUNCTION, CALCULATION*)
A:=1.31537158*T;
B:=4.94575312*87*SOR(T);
C:=4.529087942*10*T^2+SOR(T);
D:=A*B+C*D;
END(*FRAQTL*)

FUNCTION MOUT(M1,M2,D1,D2:REAL):REAL;
VAR A,C:REAL;
BEGIN(*MOUT FUNCTION, CALCULATION*)
A:=A(M2*B(M2));
C:=A(K2);(1.271 THEN (*FUNCTION (INSTABLE, XACH=1 1PPROICHED0
MOUT:=1.1;
END(*MOUT*)

FUNCTION XFRST(M,D,MDOT,Pl, TI: REAL):REAL;
VAR A,C:REAL;
BEGIN
C:=(i+1.0)+(2.0*(1.0-L));
MFRST:=(M1POtPlISOR(D)14.0)•SORT(II(RITI))t?VR(A,C)-?MOT;
END(*XFRST*)

FUNCTION MFRIC(M1,M2,F:REAL):REAL;
VAR MF:REAL;
BEGIN(*MFRIC FUNCTION, CALCULATION*)
MF:=SOR(M1)-SOR(M2)/(K*SOR(M1)*K*SOR(M2))+(K+1.1)*(K+1.1)*K*(MK2)*SOR(M1)/K*(M1)*SOR(M2))14.0*E/LD;
MF:=(K+1.1);
END(*MFRIC*)

FUNCTION XFRST(M,D,MDOT,Pl, TI: REAL):REAL;
VAR A,C:REAL;
BEGIN
C:=(i+1.0)+(2.0*(1.0-L));
MFRST:=(M1POtPlISOR(D)14.0)•SORT(II(RITI))t?VR(A,C)-?MOT;
END(*XFRST*)

FUNCTION MFRIC(M1,M2,F:REAL):REAL;
VAR MF:REAL;
BEGIN(*MFRIC FUNCTION, CALCULATION*)
MF:=(K+1.1)
END(*MFRIC*)

FUNCTION XFRST(M,D,MDOT,Pl, TI: REAL):REAL;
VAR A,C:REAL;
BEGIN
A:=A(K);
C:=(K+1.1);(2.0*(1.0-E));
MFRIC:=K*(K+3)*SOR(D)/4.0)1(SORT(K*(K+31))1*VR(A,C)-MOT;

FUNCTION MFRIC(M1,M2,F:REAL):REAL;
VAR MF:REAL;
BEGIN(*MFRIC FUNCTION, CALCULATION*)
MF:=(K+1.1);
END(*MFRIC*)
FUNCTION DNFST(M,D,Px,Tx:REAL):REAL;
(* USED IN CASE 3 OF NEWTON *)
VAR a,c,e:REAL;
BEGIN
  a:=x(M);
  c:=(x(M)+1.8)/(1.8-1.8);
  e:=(x(M)-1.8)/(1.8-1.8);
  DNFST:=(2.956*E4/(4.2))+(E2*(T2/10))+(PI*(A-C)-SQR(M)*(K-E))/2.0*PI*(A,E));
END(* DNFST *);

FUNCTION NEWTON(I:INTEGER; M,D,L,ITOT,Px,Tx:REAL):REAL;
VAR MNEW,MOLD,DIF,D2,A,Mdot,DHF:REAL;
BEGIN(* NEWTON *)
  DIF:=1.0;
  CASE I OF
    1: BEGIN
      (* CALCULATES EXIT MACH NUMBER FOR SECTION WITH AREA CHANGE *)
      MNEW:=MOLD-0.01(MOLD,Mdot);  
      WHILE (DIF > 0.001) DO
        BEGIN
          MNEW:=MOLD-Mdot(MOLD,DIF);  
          DIF:=ABS(MOLD-MNEW);
          MOLD:=MNEW;
        END(* WHILE *)
    END(* CASE 1 *);
    2: BEGIN
      (* CALCULATES EXIT MACH FOR SECTION WITH FRICTION ONLY *)
      MNEW:=MOLD-0.01(MOLD,D,HF);  
      WHILE (DIF > 0.00001) AND (COUNT < 10) DO
        BEGIN
          MNEW:=MOLD-Mdot(D,HF,1.0);  
          DIF:=ABS(MOLD-MNEW);
          IF DIF < 1.0 THEN (*FUNCTION UNSTABLE, EXIT LOOP*)
            BEGIN
              DIF:=0.00001;
              MNEW:=1.0;
            END(* IF *)
          COUNT:=COUNT+1;
          MOLD:=MNEW;
        END(* WHILE *)
    END(* CASE 2 *);
    3: BEGIN
      (* CALCULATES INLET MACH NUMBER *)
      MNEW:=MOLD-0.00001(MOLD,D,HF);  
      WHILE (DIF > 0.00001) DO
        BEGIN
          MNEW:=MOLD-Mdot(D,HF,1.0);  
          DIF:=ABS(MOLD-MNEW);
          MOLD:=MNEW;
        END(* WHILE *)
    END(* CASE 3 *);
    4: BEGIN
      (* CALCULATES INLET CONDITIONS *)
      MNEW:=MNEW;
      END(* CASE 4 *);
    END(* CASE *)
    END(* NEWTON *)
  END(* CASE *)

PROCEDURE INLETMACH(D,Mdot,Px,Tx:REAL; VAR R,H,F:REAL);
(* CALCULATES INLET CONDITIONS *)
VAR a:REAL;
BEGIN(* INLETMACH *)
  R:=NEWTON(D,Mdot,Px,Tx);
  T:=EXP(Tx,R);
  F:=PRES(F,T);
  H:=DENSE(F,T);
END(* INLETMACH *);
PROCEDURE AREACHANGE(D1,D2,T0,P0:REAL;VAR M,T,P,RHO,HE,F,NU,PR:REAL);
(* FOR DIAMETER CHANGE *)
VAR A:REAL;
BEGIN
M:=NEWTON(1.6,1.3,2.0,0.0,0.0);  
T:=TEMP(T0);  
P:=PRESS(P0);  
END:=DENSE(F,T);  
RE:=RE/D2/D2;
IF RE <= 200 THEN
BEGIN (* LAMINAR FLOW *)
  F:=16.8/RE;
  NU:=3.66;
  END(*IF*)
ELSE
BEGIN (* TURBULENT FLOW *)
  F:=0.366/PVR(HE,0.0);
  NU:=0.013*PVR(HE,0.0)*PVR(F,P0,0.0);
  END(*ELSE*)
END(*AREACHANGE*)

PROCEDURE FRICTION(T0,D1,F,L:REAL;VAR M,T,P,RHO,PR:REAL);
(* FOR FLOW WITH FRICTION ONLY *)
VAR A,C:REAL;
BEGIN
M:=NEWTON(2.25,D1,F,L,0.0,0.0);  
IF M<1 THEN IF M<1.8 THEN NU:=1.8;
A:=PETRO(0.0);
C:=(K+1)/((K+1)*K);
F:=F+M/A;
PR:=PVR(1.0,A,C);
END(*FRICTION*)

FUNCTION DIAMETER(MDOT,T0,P0,M:REAL):REAL;
(* CALCULATES DIAMETER GIVEN MDOT AND MACH *)
VAR A,C:REAL;
BEGIN(*DIAMETER*)
  A:=1.6*MDOT*SORT(T0/E)/(EI*MDOT);
  C:=(K+1)/(2.0*(K-1.0));
  DIAMETER:=SORT(A*PVR(1.0,C));
END(*DIAMETER*)

PROCEDURE SECTIOH(HUMTOT:INTEGER; LENGTH:DIVL:REAL; VAR I:DIVISIONS; VAR XM:N:INTEGER);
(* DIVIDES PIPE LENGTHS INTO SMALLER SECTIONS *)
VAR RESL,REALDIV:REAL;
I,J:INTEGER;
BEGIN(*SECTION*)
  IF HUMTOT = 1 THEN
    BEGIN
      HUMTOT:=0;
      XM:=0;
    END(*IF*)
  (* CALCULATE RESIDUAL SOLUTION SECTION LENGTH *)
  REALDIV:=LENGTH/DIVL;
  NUMDIV:=TRUNC(REALDIV);
  REAL:=REALDIV-TRUNC(REALDIV)*DIVL;
  IF REAL > 0 THEN
    BEGIN
      NUMDIV:=NUMDIV+1;
    END(*IF*)
  ELSE
    BEGIN
      RESL:=DIVL;
      END(*ELSE*)
  (* DEFINE LENGTH SECTION ARRAY *)
  NUMDIV:=HUMTOT+NUMDIV;
  FOR I:=1 TO NUMDIV DO (X[I]:=X[I-1]+DIVL)
  IF NUMDIV+1 THEN (X[NUMDIV+1]:=RESL)
  END(*SECTION*)
FUNCTION STAGTEMP(TS,TW,NU,EF,MDOT,L:REAL);
(* CALCULATES NEW STAGNATION TEMPERATURE FOR FLOW WITH HEAT TRANSFER *)
BEGIN
  STAGTEMP:=TS+(EF*(NU*EF*L)/(MDOT*C));
END;(*STAGTEMP*)

FUNCTION FUNCN(TS,TW,PR,NU:REAL; KEY:INTEGER:REAL);
(* SOLVES AN OPTIMIZED EQUATION IN PROCEDURE HEATTRANS *)
VAR A,C,D,E:REAL;
BEGIN
  A:=PR*(0.8*(1.0-SOR(M)));
  C:=L*(1.0-SOR(N))/E;
  D:=E*SOR(N);
  IF KEY = 1 THEN
    BEGIN
      E:=1.0*(T-W);
      END;(*IF*)
  ELSE
    BEGIN
      E:=16.0*PR/(NU*(T-W));
      END;(*ELSE*)
  END;(*IF*)
END;(*FUNCN*)

PROCEDURE HEATTRANS(T,L,DO,MDOT,NU:REAL; N:INTEGER; VAR M,TS,TW,PR,NU:REAL);
(* CALCULATES FLOW VARIABLES FOR FLOW WITH FRICTION AND HEAT TRANSFER *)
VAR
  C1,C2,C3,C4,STEP,A,C:REAL;
  TS1,TS2,NI,PF,KF,DELN,X:REAL;
  KEY:INTEGER;
BEGIN
  X:=0;
  (* FLAG KEY IF FLOW TURBULENT *)
  IF NU > 3.2 THEN KEY:=1;
  (* CALCULATE NEW STAGNATION TEMPERATURE *)
  TS:=TS2;
  TS2:=STAGTEMP(TS1,TW,NU,EF,MDOT,L);
  (* STEPWISE SOLUTION USING RUNGE-KUTTE METHOD *)
  (* CALCULATE STEP SIZE *)
  STEP:=ABS((TS2-TS1)/N);
  M:=N;
  I:=1;
  WHILE I < N DO
    BEGIN
      CI:=FUNCN(TS,TS1,TW,PR,NU,KEY);
      IF CI < 0 THEN CI:=0.0;(*FUNCTION UNSTABLE*)
      C1:=CFUNCI(TS+FIT(0.0),TS+STEP/2.0,TS1,TW,PR,NU,KEY);
      IF CI < 0 THEN CI:=0.0;(*FUNCTION UNSTABLE*)
      C2:=CFUNCI(TS+FIT(0.0),TS+STEP/2.0,TS1,TW,PR,NU,KEY);
      IF CI < 0 THEN CI:=0.0;(*FUNCTION UNSTABLE*)
      C3:=CFUNCI(TS+FIT(0.0),TS+STEP/2.0,TS1,TW,PR,NU,KEY);
      IF CI < 0 THEN CI:=0.0;(*FUNCTION UNSTABLE*)
      DELT:=STEP/4.0*(C1+2.0*C2+2.0*C3+C4);
      M:=M-DELN;
      TS:=TS+STEP;
      I:=I+1;
      IF M = 1 THEN (*FUNCTION UNSTABLE, EXIT LOOP*)
        BEGIN
          N:=N+1;
          M:=2;
          END;(*IF*)
    END;(*WHILE*)
  IF (C1 = 0) OR (C2 = 0) OR (C3 = 0) OR (C4 = 0) THEN
    (*FUNCTION UNSTABLE, EXIT LOOP*)
    BEGIN
      M:=N+1;
      N:=4;
      END;(*IF*)
  END;(*WHILE*)
  M2:=M;
  A:=M(2)/(M1);
  C:=(K1.0)*(C-1.0)));
PROCEDURE SETBOUND(KEY:INTEGER; VAR HIB,LOWB,DIA:REAL);
(* ALTERS DIAMETER SO THAT NO KEY 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)/2.8;
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 ARRT:COUNT; VAR CKT:COUNT);
(* USED IN FUNCTION HDOTHIN TO FIX OPTIMUM MASS FLOW *)
VAR
RCHT:COUNT;
INDEX:INTEGER;
SWITCH:BOOLEAN;
BEGIN(*BSORT*)
FOR I := START TO TOP DO RCHT[I] := CKT[I];
REPEAT(* UNTIL SORTED WITH SMALLEST AT TOP *)
SWITCH := FALSE;
FOR I := START TO TOP-1 DO
BEGIN
IF ARRT[I] > ARRT[I+1] THEN
BEGIN
SWAP(ARRT[I],ARRT[I+1]);
SWAP(RCHT[I],RCHT[I+1]);
SWITCH := TRUE;
END(*IF*)
END(*FOR*)
UNTIL SWITCH = FALSE;
FOR I := START TO TOP DO CKT[I] := ROUND(RCHT[I]);
END(*BSORT*)

FUNCTION HDOTHIN(GQ,T,V:COUNT; T01,EFF:REAL; N:INTEGER):REAL;
(* DETERMINES OPTIMUM MASS FLOW *)
VAR
QTOT,TOT,HDOT:COUNT;
INDEX:INTEGER;
I,CONT:INTEGER;
BEGIN(*MDOTMIN*)
(* INITIALIZE INDEX ARRAY *)
FOR I:=1 TO N DO INDEX(I):=1;
(* CALCULATE SHIELD EXIT STAGNATION TEMPERATURES *)
THMAX(I):=TV(I);
FOR I:=1 TO N DO THMAX(I):=EFF*(TV(I)-THMAX(I-I)+THMAX(I-I));
(* CALCULATE MASS FLOWS AS DETERMINED BY SECTIONS *)
QTOT(I):=0;
FOR I:=1 TO N DO
BEGIN
QTOT(I):=QTOT(I-I)+Q(I);
NVT(I):=QTOT(I)/CP(TMAX(I)-TH(I));
END;(*FOR*)
(* SEARCH FOR LARGEST MDOT *)
SORT(I,H,MDOT.INDEX);
MDOTMIN:=MDOT(INDEX);
END;(*MDOTMIN*)
PROCEDURE INREASSIGN;
(* DEFINES VALUES FOR USE IN PARAMETER LIST *)
BEGIN(*INREASSIGN*)
I:=I1;
T(I):=TV(I);
P(I):=P(I);
M(I):=M(I);
RHO(I):=RHO(I);
F(I):=F(I);
HE(I):=HE(I);
Q(I):=Q(I);
G(I):=G(I);
NU(I):=NU(I);
END;(*INREASSIGN*)
PROCEDURE OUTREASSIGN;
(* DEFINES NEW VALUES OBTAINED FROM PARAMETER LIST *)
BEGIN(*OUTREASSIGN*)
I:=1;
M(I):=M(I);
T(I):=T(I);
P(I):=P(I);
M(I):=M(I);
RHO(I):=RHO(I);
F(I):=F(I);
HE(I):=HE(I);
Q(I):=Q(I);
G(I):=G(I);
NU(I):=NU(I);
END;(*OUTREASSIGN*)
PROCEDURE INITIALIZE;
(* PREPARES VARIABLES FOR USE IN MAIN PROGRAM *)
BEGIN(*INITIALIZE*)
(* DETERMINE MINIMUM MASS FLOW *)
MDOT:=MDOTMIN(QV.TV,T(I),MASS,NUMWALLS);
(* DEFINE FIRST SECTION SIZE *)
I1:=DIVLENGTH;
(* DEFINE IMAGINARY LAST PIPE LENGTH *)
SPACE=NUMWALLS/2.1;=0;
(* START LOOP COUNTER *)
LOOPCOUNT:=4;
(* INITIAL MACH NUMBER GUESS *)
MI1:=d11;
(* INITIAL DIAMETER GUESS, ASSUMES INLET MACH OF 0.1 *)
D(I):=DIAMETER(MDOT,T(I),P(I),MI1);
(* CALCULATE MINIMUM DIAMETER, MACH = 1 *)
LOWBOUND:=DIAMETER(MDOT,T(I),P(I),1.0);
(* DEFINE MAXIMUM DIAMETER *)
HIGHBOUND:=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*)
\[ LTOTAL:=SPACE(1); \]

(*CALCULATE ENTRANCE CONDITIONS*)
\[ QEl(1):=0; QE(1):=0; \]
\[ RE(1):=6.0*H0H0/(P1*Q1*T01)\#DIAL(LOOPCOUNT); \]
\[ IF RE(1) (.0) THEN \]
\[ BEGIN (* LAMINAR FLOW*) \]
\[ FC1:=16/REV1; \]
\[ ML(1):=1.44; \]
\[ END(*IF*) \]

ELSE \[ BEGIN (* TURBULENT FLOW*) \]
\[ FC1:=0.0666/FVR(RE(1),0.33); \]
\[ ML(1):=0.233*FVR(RE(1),0.8)*FVR(RE(1),0.44); \]
\[ END;(*ELSE*) \]

(*CALCULATE INLET CONDITIONS*)
\[ INLETMACH(DIA(LOOPCOUNT),P01,T01,M01,T11,PI1,H01); \]

(*INITIALIZE CONTROL CARD*)
\[ MTEST:=1; \]
\[ END;(*REINITIALIZE*) \]

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

REPEAT (* UNTIL THE FLOW IS CHOKE *)
REINITIALIZE;
\[ WHILE (SPACECNT (= NUMWALLS+1) AND (MTEST(2)) DO \]
BEGIN (* TEST FOR CHANCE IN TEMPERATURE BOUNDARY *) \[ IF M01 (.0) THEN \]
\[ BEGIN (* DIVIDE FRICTION LENGTH INTO SECTIONS *) \]
\[ SECTION(1,SPACE(1),SPACECNT,DIVLENGTH,1,NUMDIV); \]
\[ (* DEFINE END MACH NUMBER FOR TEST AT END *) \]
\[ MINDIV:=-2; \]
\[ WHILE (1 ( NUMDIV) AND (MTEST(3)) DO \]
\[ BEGIN \]
\[ L:=M01-M01; \]
\[ INEREASSIGN; \]
\[ FRICTION(T01,DIA(LOOPCOUNT),PI1,L1,L1,P11,P01,Q01); \]
\[ OUTEREASSIGN; \]
\[ IF M01 (.0) THEN MTEST:=3 (*EXIT LOOP*) \]
\[ END;(*WHILE*) \]
\[ SPACECNT:=SPACECNT+1; \]
\[ PIPECNT:=PIPECNT+1; \]
\[ END (* FRICTION FLOW*) \]
ELSE \[ BEGIN (* TEMPERATURE BOUNDARY CHANGE *) \]
\[ QUALWALLCNT:=0; \]
\[ LTEMPWALLCNT:=0; \]
\[ EFFWALLCNT:=0; \]
\[ TOSAVE:=T01; \]
\[ WHILE (QUALWALLCNT) OR (EFFWALLCNT) AND (MTEST(2)) DO \]
BEGIN \[ INEREASSIGN; \]
\[ HEATRANS(TWALLCNT,DIVLENGTH,DIA(LOOPCOUNT),Q01,T01,P01,T11,P11,H01,HE1,F11,Q11,HE1,ML1); \]
\[ OUTEREASSIGN; \]
\[ QUALWALLCNT:=0; \]
\[ EFFWALLCNT:=0; \]
\[ TOSAVE:=TOSAVE; \]
\[ END;(*WHILE*) \]
\[ LTOTAL:=LTOTAL-LTEMPWALLCNT\#SPACE(SPACECNT); \]
\[ PIPECNT:=PIPECNT+1; \]
\[ WALLCNT:=WALLCNT+1; \]
\[ END (* TEMPERATURE BOUNDARY CHANGE*) \]
END;(*WHILE*);
DIAL1-DIALOGCOUNT;
SETNUMINTEST.MINBOUND,LOWBOUND,DIAL1;
DIALOGCOUNT+1:=DIAL1;
UNTIL (MINNUMDIVVT ) < (FF) AND (PIPECNT=2*NWalls+1) OR (LOOPCOUNT = 2D);

(*CALCULATE OVERALL EFFECTIVENESS *)
EFFECTOTAL+(1.0*TR(NUMDIVVT-TS1))/TU(NWalls)-TS(11));
(*CALCULATE TOTAL HEAT TRANSFER *)
TOTAL(NWalls+1):=NUMC*EP(1*NUMDIVVT-TS1));
WRITEML('OUTPUT STORED IN FILE EOUTUD. ');
WRITEML('INPUT STORED IN FILES EFIND1 AND EFIND2.');
END (*MAIN PROGRAM*)
TO ENTER DATA, ENTER 1. OTHERWISE, ENTER 0.

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

NUMBER OF SHIELDS = 5

ENTER SINGLE DIVISION LENGTH (M)

DIVISION LENGTH (M) = 0.00200

ENTER PIPE LENGTHS NOT ON SHIELDS (M)

LENGTHS = 0.019, 0.021, 0.024, 0.026, 0.034, 0.035

ENTER TEMPERATURES OF RADIATION SHIELDS (K)

TEMPERATURES = 21.7, 39.7, 65.0, 101.1

ENTER HEAT INPUTS OF RADIATION SHIELDS (W)

INPUTS = 0.084, 0.084, 0.094, 0.094, 0.094, 0.094, 0.094

ENTER MAXIMUM HEAT EXCHANGER EFFECTIVENESS AS A FRACTION

MAXIMUM EFFECTIVENESS = 0.9800000

ENTER INITIAL STAGNATION T (K)

STAGNATION TEMPERATURE (K) = 4.23600

ENTER INITIAL STAGNATION P (PA)

STAGNATION PRESSURE (PA) = 2100.00000

ENTER NUMBER OF STEPS IN RUNGE-KUTTA METHOD

STEPS = 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.

FOR A PLOT FILE ENTER 1, OTHERWISE ENTER 0

NUMBER OF INTERATIONS = 13

FOR INTERATION INFORMATION, INPUT 1. OTHERWISE INPUT 0

2.806 CF SECs, 1027168 CM USED.
**Values of Input**

- **Inlet Stagnation Temperature (K)**: 4.2300008000
- **Inlet Stagnation Pressure (Pa)**: 1160.00000000
- **Number of Radiation Shields**: 3

**Temperatures of Radiation Shields**

<table>
<thead>
<tr>
<th>Shield</th>
<th>Temperature (K)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>31.7000000000</td>
</tr>
<tr>
<td>2</td>
<td>39.7000000000</td>
</tr>
<tr>
<td>3</td>
<td>45.4000000000</td>
</tr>
<tr>
<td>4</td>
<td>105.1900000000</td>
</tr>
<tr>
<td>5</td>
<td>146.4000000000</td>
</tr>
</tbody>
</table>

**Heat Inputs to Radiation Shields**

<table>
<thead>
<tr>
<th>Shield</th>
<th>Heat (W)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.0000000000</td>
</tr>
<tr>
<td>2</td>
<td>0.5540000000</td>
</tr>
<tr>
<td>3</td>
<td>0.9710000000</td>
</tr>
<tr>
<td>4</td>
<td>0.1380000000</td>
</tr>
<tr>
<td>5</td>
<td>0.3540000000</td>
</tr>
</tbody>
</table>

**Pipe Lengths Between Shields**

<table>
<thead>
<tr>
<th>Space Length (H)</th>
<th>( H_1 = 0.0190000000 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( H_2 = 0.0230000000 )</td>
<td></td>
</tr>
<tr>
<td>( H_3 = 0.0240000000 )</td>
<td></td>
</tr>
<tr>
<td>( H_4 = 0.0340000000 )</td>
<td></td>
</tr>
<tr>
<td>( H_5 = 0.0355000000 )</td>
<td></td>
</tr>
</tbody>
</table>

**Solution Section Size (m)**: 0.0020000000

**Maximum Shield Exchanger Effectiveness**: 0.0000000000

**Values of Output**

- **Mass Flow (kg/s)**: 0.0000000000
- **Pipe Diameter (m)**: 0.0000000000
- **Total Pipe Length (m)**: 0.2530000000

**Shield Lengths**

<table>
<thead>
<tr>
<th>Shield</th>
<th>Length (m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.0320000000</td>
</tr>
<tr>
<td>2</td>
<td>0.0310000000</td>
</tr>
<tr>
<td>3</td>
<td>0.0010000000</td>
</tr>
<tr>
<td>4</td>
<td>0.0061000000</td>
</tr>
<tr>
<td>5</td>
<td>0.0310000000</td>
</tr>
</tbody>
</table>

**Total Heat Transfer (W)**: 1.1933333333

**Shield Heat Transfer**

<table>
<thead>
<tr>
<th>Shield</th>
<th>Heat (W)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.0965996340</td>
</tr>
<tr>
<td>2</td>
<td>0.0434681350</td>
</tr>
<tr>
<td>3</td>
<td>0.1081726340</td>
</tr>
<tr>
<td>4</td>
<td>0.1277294010</td>
</tr>
<tr>
<td>5</td>
<td>0.7747349749</td>
</tr>
</tbody>
</table>

**Overall Exchanger Effectiveness**: 0.9998444444

**Exit Stagnation Temperature (K)**: 145.1490257355

**Exit Stagnation Pressure (Pa)**: 467.3579230706

**Length** | **Number** | **Stag Temp** | **Stag Pres** | **Temp** | **Pressure** | **Muselit** | **Reynold Num** | **Section Heat** | **Total Heat** |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0.803000</td>
<td>0.80154443</td>
<td>4.230000</td>
<td>105.000000</td>
<td>3.2397</td>
<td>2075.193</td>
<td>3.640</td>
<td>712.65327</td>
<td>0.0000000000</td>
<td>0.0000000000</td>
</tr>
<tr>
<td>0.804000</td>
<td>0.80147979</td>
<td>4.235000</td>
<td>1075.77130</td>
<td>3.3735</td>
<td>2075.163</td>
<td>3.640</td>
<td>712.65327</td>
<td>0.0000000000</td>
<td>0.0000000000</td>
</tr>
<tr>
<td>0.804000</td>
<td>0.80157979</td>
<td>4.235000</td>
<td>1075.75806</td>
<td>3.3735</td>
<td>2075.140</td>
<td>3.640</td>
<td>712.65327</td>
<td>0.0000000000</td>
<td>0.0000000000</td>
</tr>
</tbody>
</table>

**Flow Output**
| 1.12106 | 0.01670343 | 35,327246 | 1948.76425 | 35.1741 | 1141.473 | 3.661 | 235.41742 | 1.0000100 | 1.2631203 |
| 0.10110 | 0.05627711 | 23.161313 | 1913.27337 | 13.1340 | 1971.040 | 3.660 | 235.41742 | 1.0000100 | 1.2631203 |
| 1.13210 | 1.06661421 | 33.327246 | 1959.97312 | 35.1741 | 1141.473 | 3.660 | 235.41742 | 1.0000100 | 1.2631203 |
| 1.13101 | 1.11611201 | 33.172266 | 1176.41111 | $3.1313 | 1144.104 | 3.660 | 174.61461 | 1.0000001 | 1.4201717 |
| 1.12401 | 1.06446119 | 35.317246 | 1116.26171 | 3.1757 | 1161.171 | 3.660 | 311.60051 | 1.0010000 | 1.1541477 |
| 1.11401 | 1.06446119 | 35.317246 | 1116.26171 | 3.1757 | 1161.171 | 3.660 | 311.60051 | 1.0010001 | 1.1549177 |
| 1.11401 | 1.06446119 | 35.317246 | 1116.26171 | 3.1757 | 1161.171 | 3.660 | 311.60051 | 1.0010001 | 1.1549177 |
| 1.14711 | 1.13114114 | 13.113576 | 2037.10573 | 13.9711 | 2035.111 | 3.661 | 412.23767 | 1.1133171 | 1.6511635 |
| 1.11711 | 1.12174724 | 11.171361 | 2061.61136 | 11.1171 | 2051.161 | 3.161 | 413.67537 | 1.1000101 | 1.1115370 |
| 0.89600 | 0.15131051 | 12.542701 | 1111.21160 | 12.3221 | 2116.661 | 3.660 | 311.61031 | 0.1000001 | 1.1341477 |
| 0.89811 | 1.13171693 | 21.111921 | 1111.10313 | 11.0851 | 2144.357 | 3.661 | 311.10051 | 1.1010000 | 1.1101111 |
| 0.92000 | 0.15131051 | 12.542701 | 1111.21160 | 12.3221 | 2116.661 | 3.660 | 311.61031 | 0.1000001 | 1.1341477 |
| 0.90800 | 0.15131051 | 12.542701 | 1111.21160 | 12.3221 | 2116.661 | 3.660 | 311.61031 | 0.1000001 | 1.1341477 |
| 0.90001 | 1.01107140 | 4.231011 | 2077.97171 | 4.2213 | 1077.157 | 3.661 | 713.13337 | 0.1000001 | 1.1400010 |
| 0.89611 | 1.13171693 | 21.111921 | 1111.10313 | 11.0851 | 2144.357 | 3.661 | 311.10051 | 1.1010000 | 1.1101111 |
| 1.13711 | 1.13341104 | 11.171361 | 2061.61136 | 11.1171 | 2051.161 | 3.161 | 413.67537 | 1.1000101 | 1.1115370 |
| 0.90001 | 1.01107140 | 4.231011 | 2077.97171 | 4.2213 | 1077.157 | 3.661 | 713.13337 | 0.1000001 | 1.1400010 |
| 1.13711 | 1.13341104 | 11.171361 | 2061.61136 | 11.1171 | 2051.161 | 3.161 | 413.67537 | 1.1000101 | 1.1115370 |
| 0.90001 | 1.01107140 | 4.231011 | 2077.97171 | 4.2213 | 1077.157 | 3.661 | 713.13337 | 0.1000001 | 1.1400010 |
| 1.13711 | 1.13341104 | 11.171361 | 2061.61136 | 11.1171 | 2051.161 | 3.161 | 413.67537 | 1.1000101 | 1.1115370 |
| 0.90001 | 1.01107140 | 4.231011 | 2077.97171 | 4.2213 | 1077.157 | 3.661 | 713.13337 | 0.1000001 | 1.1400010 |

**ORIGINAL PAGE IS OF POOR QUALITY**
THIS PROGRAM DETERMINES CERTAIN PARAMETERS OF THE EFFLUX
RESULTING FROM THE PARASITIC HEAT LOAD ON A HELIUM Dewar in SPICE. THE SHIELD SYSTEM CONSISTS OF CONSTANT TEMPERATURE SHIELDS WITH A GIVEN DIAMETER AND PIPE LENGTHS. THE MASS FLOW VARIABLES.

LABEL 18;

CONST
PI=3.141592653;
K=1877;(kJ/kg-K)
G=1.47;
CP=1288;

TYPE DIVISIONS=ARRAY(.1001 OF REAL;
COUNT=ARRAY(.101 OF REAL;
ICOUNT=ARRAY(.101 OF INTEGER;
REALOUT=FILE OF REAL;
INTOUT=FILE OF INTEGER;
VAR NUMWALLS,NUMDIV,N,TEST,PEST:INTEGER;
I,LOOPCOUNT,TAB,FIVECWT,WALLCNT,SPACECNT:INTEGER;
EFFOUT,POUT,TEST;
EFFINI,SEALOUT;
EFSINI,INTOUT;
F.T.R.H.:DIVISIONS;
I.G.N.CH.TO.O,RE,F.NW,MCT:DIVISIONS;
EFF,TV,SPACE,PIPE,LVALL,OSHEILD,COUNT;
% EDH,31A,LOGBOUND,LOGBOUND:REAL;
TSAVE,GSAVE,MCT1,MCTH MAX:REAL;
PROCEDURE INDATA(INDEX: INTEGER);
BEGIN
  IF INDEX = 0 THEN
    BEGIN (* READ IN DATA *)
      WRITE('INPUT DATA AS PROMPTED BY TERMINAL');
      WRITE(' '); 
      WRITE('WHEN ASKED FOR A SERIES OF DATA SUCH AS THE');
      WRITE('SHIELD TEMPERATURES, INPUT THE DATA IN');
      WRITE('ORDER FROM THE DEVAR TO THE OUTER BOUNDARY');
      WRITE(' '); 
      WRITE('ENTER THE NUMBER OF RADIATION SHIELDS');
      READLN;
      READ(NUWALLS);
      WRITE('NUMBER OF SHIELDS = ',NUWALLS:3); 
      WRITE('ENTER NUMBER OF SECTIONS FOR SOLUTION');
      READLN;
      READ(NUDIV);
      WRITE('NUMBER OF SOLUTION SECTIONS = ',NUDIV:3); 
      WRITE('ENTER DIAMETER (M)');
      READLN;
      READ(DIA);
      WRITE('DIAMETER (M) = ',DIA:15:2); 
      FOR I:=1 TO NUWALLS+1 DO
        BEGIN (* FOR *)
          READLN;
          READ(LSPACE(I));
          WRITE('LENGTH = ',LSPACE(I):15:2);
        END; (* FOR *)
      WRITE('ENTER TEMPERATURES OF RADIATION SHIELDS (E)');
      FOR I:=1 TO NUWALLS DO
        BEGIN (* FOR *)
          READLN;
          READ(T(V[I]));
          WRITE('TEMPERATURE = ',T(V[I]):15:2);
        END; (* FOR *)
      END; (* IF INDEX = 0 THEN *)
  END; (* IF INDEX = 0 THEN *)
ORIGINAL PAGE 13
OF POOR QUALITY

```
READLN;
READIN;
WRITE Eff1(1:M);
WRITE Eff2, 'NUMBER OF STEPS IN RUNGE-KUTTA', N:5;
END (*IP*)
ELSE
(* READ FILE *)
BEGIN
RESET (Eff1);
RESET (Eff2);
READ(eff1, NUMWALLS);
READ(eff1, NUMDIV);
READ(eff1, DI8);
FOR I := 1 TO NUMWALLS DO READ(eff1, LSPACE(I));
FOR I := 1 TO NUMWALLS DO READ(eff1, LVAL(I));
FOR I := 1 TO NUMWALLS DO READ(eff1, TVEI);
LEAD (Eff1, M1);
WRITE

PROCEDURE OUTDATA;
BEGIN
WRITE Eff1, 'FLAME OUTPUT', N:60;
WRITE Eff1, ' ', N:60;
WRITE Eff1, 'VALUES OF INPUT', N:60;
WRITE Eff1, ' ', N:60;
WRITE Eff1, 'NUMBER OF SECTIONS', N:60;
WRITE Eff1, 'DIVAR STAGNATION TEMPERATURE', N:60;
WRITE Eff1, 'DIVAR STAGNATION PRESSURE', N:60;
WRITE Eff1, 'MASS FLOW APPROXIMATION'; N:60;
WRITE Eff1, ' ', N:60;
WRITE Eff1, 'SHIELD TEMPERATURES', N:60;
FOR E := 1 TO MALLS DO WRITE Eff1, 'SHIELD TEMPERATURE', N:60;
WRITE Eff1, ' ', N:60;
WRITE Eff1, 'MASS FLOW', N:60;
WRITE Eff1, ' ', N:60;
WRITE Eff1, 'TOTAL HEAT TRANSFER', N:60;
WRITE Eff1, ' ', N:60;
WRITE Eff1, 'SHIELD HEAT TRANSFER', N:60;
WRITE Eff1, ' ', N:60;
WRITE Eff1, 'SHIELD HEAT EXCHANGE EFFECTIVENESS'; N:60;
WRITE Eff1, ' ', N:60;
WRITE Eff1, 'FOR AN OUTPUT OF SECTION FLOW VARIABLES, INPUT 1. OTHERWISE INPUT 0. ');
READIN;
READ (key);
IF key = 1 THEN
BEGIN
WRITE Eff1, 'LENGTH', N:10;
WRITE Eff1, 'MACH NUMBER', N:15;
WRITE Eff1, 'STAG TEM', N:15;
WRITE Eff1, 'STAG PRESSURE', N:15;
WRITE Eff1, 'TEMPERATURE', N:15;
WRITE Eff1, 'PRESSURE', N:15;
WRITE Eff1, 'NUSSELT', N:15;
WRITE Eff1, 'REYNOLDS', N:15;
WRITE Eff1, 'SECTION HEAT', N:15;
WRITE Eff1, 'TOTAL HEAT', N:15;
FOR I := 1 TO NUMDIV DO
BEGIN
WRITE Eff1, ' ', N:15;
END (*IF*)
END (*END*)
END (*BEGIN*)
```

```
FOR A PLOT FILE ENTER 1, OTHERWISE ENTER 0*);
READIN;
READ (Tab);
IF Tab = 0 THEN
BEGIN
WRITE Eff1, 'FOR PLOT OF STAGNATION TEMPERATURE ENTER 1.';
```
WRITELM('FOR PLOT OF STAGNATION PRESSURE ENTER 2,');
WRITELM('FOR PLOT OF HEAT TRANSFER ENTER 3,');
READIN:READ(TAB);
FOR 1:1 TO NGRID DO
BEGIN
CASE TAB OF
1: WRITELM('POUT,TRE(1)');
2: WRITELM('POUT,FAC(1)');
3: WRITELM('POUT,DI(1)');
END(*CASE*)
END(VFOR*)
WRITELM('FOR INTERACTION INFORMATION, INPUT 1. OTHERWISE INPUT 2.');
READIN:READ(FYRT);
IF FYRT = 1 THEN
FOR 1:1 TO LOOPCOUNT DO
WRITELM('EFFOUT,'INTERACTION#':11,1;1,'MASS FLOM':11,1;MOUT(1,1:11));
WRITELM('OUTPUT STORED IN FILE EFFOUT');
END(* OUTDATA *)

FUNCTION PVR(M,P:REAL):REAL;
(* SOLVES FOR M=P *)
VAR A:REAL;
BEGIN
A:=P*M;
PVR:=EXP(A);
END(* PVR *)

FUNCTION R(M:REAL):REAL;
(* SOLVES OFTEN OCCURRING FUNCTION *)
BEGIN
R:=1.8+(E^-4.4)*SQRT(M)/2;  
END(* R *)

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

FUNCTION PRES(P0,M:REAL):REAL;
(* CALCULATES LOCAL PRESSURE *)
VAR A,C:REAL;
BEGIN
C:=3*M;
A:=E/(1-E);  
PRES:=P0*PVR(C,A);  
END(* PRES *)

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

FUNCTION MOLT(R:REAL):REAL;
VAR A,B,C,D:REAL;
BEGIN(MOLT)
A:=1.0417*55531E-64;
B:=1.0447*5216E-67T;
C:=3.322*272341E-19*SQRT(T);
D:=4.7231002131229-131T12/3;
RT:=A+B+C+D;  
END(MOLT)

FUNCTION COND(T:REAL):REAL;
VAR A,B,C:REAL;
BEGIN(CONDUCTIVITY CALCULATION*)
A:=8.818728E01;
...
FUNCTION FRANDTL(T:REAL):REAL;
VAR A,B,C,D,E:REAL;
BEGIN(*FRANDTL NUMBER CALCULATION*)
A:=a; 3.333333333E+0;
B:=a; 855382197E-0;
C:=a; -0.9234521E-0;
D:=a; 3.953051E-0;
END;(*FRANDTL*)

FUNCTION HERIC(M1,M2,D,F,L:REAL):REAL;
VAR MRF:REAL;
(* USED IN CASE 1 OF NEWTON *)
BEGIN
MRF:=(SOR(M1))-SQR(M1))/(K*SQR(M1)+M1*(1.8)/(2.0*K)*LN(K*M1)*SQR(M1)/(K*M1)*SQR(M1))-1.0*L/D;
MRF:=NRF;
END;(* HERIC *)

FUNCTION DFRIC(M1,M2:REAL):REAL;
(* USED IN CASE 1 OF NEWTON *)
VAR DFR:REAL;
BEGIN
DFR:=(E*M1*SQR(M2))/((K+E*M1*SQR(M2))/((K*M1)+M1*(1.8)/(2.0*K)*LN(K*M1)*SQR(M1)/(K*M1)*SQR(M1))/K*M1)*SQR(M1); DFR:=DFR;
IF DFR < 1.274 THEN (*FUNCTION UNSTABLE, EXIT 1 APPROACHED*)
DFR:=1.1;
END;(* DFRIC *)

FUNCTION HERSTUN,D,HDT,F0,T0:REAL):REAL;
(* USED IN CASE 1 OF NEWTON *)
VAR A,C:REAL;
BEGIN
A:=a; 3.0;
C:=(K1.0)/(1.0+(1.0-K1.2));
HERST:=C*F0*F*SQR(D)/((K*1.0)*SQR(K)/(K*1.0))*FVR(A,C,HDT);
END;(* HERST *)

FUNCTION DHERST(N,D,F0,T0:REAL):REAL;
(* USED IN CASE 1 OF NEWTON *)
VAR A,C,E:REAL;
BEGIN
A:=a; 2.0;
C:=(K1.0)/(1.0+(1.0-K1.2));
E:=(3.0*K1.0)/(1.0+(1.0-K1.2));
DHERST:=(F0*SQR(D)/((K*1.0)*SQR(K)/(K*1.0))*FVR(A,C,SQR(K)*FVR(K1.0)+1.0*SQR(E);)
END;(* DHERST *)

FUNCTION NEWTON(KEY:INTEGER; M1,D,F,L,LTOT,F0,T0:REAL):REAL;
VAR HDTIV,HOLD,DIF,DI,T,T0,HDT,T1:REAL;
COUNT:INTEGER;
BEGIN(* NEWTON *)
DIF:=1.0;
CASE KEY OF
1: BEGIN (* CALCULATES EXIT RACE FOR SECTION WITH FRICTION ONLY *)
END;
COUNT:=1;
WHILE (DIFF >= 0.00005) AND (COUNT < 10) DO
BEGIN
DIF:=DHERST(M1,HDT,T0);
HDTIV:=HOLD-HERIC(M1,HOLD,D,F,L)/DIF;
DIF:=ABS(HDTIV-HDTIV);
END;
IF DIF = 1.0 THEN (*FUNCTION UNSTABLE, EXIT LOOP*);
BEGIN
DIFF:=0.00001;
NEWV:=A.B;
END;(* IF *)
COUNT:=COUNT+1;
HOLD:=NEWV;
END;(* WHILE *)
END;(* CASE *)

BEGIN
(* CALCU LATES INLET MACH NUMBER *)
NEWV:=A.B;
HOLD:=HOLD;
WHILE (DIFF > 0.80001) DO
BEGIN
MNEW:=NEW(MDOT,DP,F,C,IV,0.0,0.0);
DIFF:=ABS(MNEW-MNEW);
HOLD:=NEWV;
END;(* WHILE *)
END;(* CASE *)
NEWTON:=HOLD;
END;(* NEWTON *)

PROCEDURE INLETHMAC(D,M,DOT,F,C,IV,0.0,0.0,T):
VAR M,D,DOT,F,C,IV,0.0,0.0,T:REAL;
BEGIN
IF T = 0 THEN
BEGIN
NEWV:=MNEW;
END;(* NEWTON *)
END;(* INLETHMAC *)

PROCEDURE FRICTION(T,D,M,F,L:REAL;VAR M,D,EX:REAL);
(* FOR FLOW WITH FRICTION ONLY *)
VAR M,D,EX:REAL;
BEGIN
MNEW:=NEWTON(1.0,D,M,F,L,TOTAL,1.0,1.0);
A:=B(MNEW)/B(M);
C:=C(M,1.0)/(2.0*(1.0-A));
F,D:=F*F/M/MNEW+F/M+A,C;
END;(* FRICTION *)

FUNCTION STAGTEMPO(T,L,TV,N1T,nu):REAL;
(* CALCULATES NEW STAGNATION TEMPERATURE FOR FLOW WITH HEAT TRANSFER *)
VAR A,T,L,TV,N1T,nu:REAL;
BEGIN
TVNEW:=TV+(N1T*TV)/M(DOT+CP));
END;(* STAGTEMPO *)

FUNCTION RFUNCMT(T,TV,FR,NU,KET:REAL; KET:INTEGER):REAL;
(* SOLVES AN OFTEN USED FUNCTION IN PROCEDURE HEATTRANS *)
VAR A,T,L,TV,FR,NU,KET:REAL;
BEGIN
A:=H/(1.0-SQR(1.0-A));
C:=C*(1.0-SQR(1.0-A))/T;
D:=A-SQR(1.0-A);
IF KET = 1 THEN
BEGIN
E:=2.0*(TV-TB);
END;(* IF *)
ELSE BEGIN
E:=14.0*(TV-TB);
END;(* ELSE *)
RFUNC:=A*(D*C);
END;(* RFUNC *)
PROCEDURE HEATTANSITV.L.O,HOT; REAL; M: INTEGER; VAR M,T0,T,F,END,RE,F,G,NU:REAL;
(* CALCULATES FLOW VARIABLES FOR FLOW WITH FRICTION AND HEAT TRANSFER *)
VAR
C1,C2,C3,C4,STEP,A.C.R,NEV,FACOR:REAL;
T11,T12,M1,M2,F11K:REAL;
N1,N2:INTEGER;
BEGIN
EXX:=0;
TEST:=0;
(* FLAG EXIT IF FLOW IS TURBULENT *)
IF NU < 3.44 THEN EXIT:={};
(* CALCULATE NEW STAGNATION TEMPERATURE *)
F1:=F11K(T);
F2:=F11K(T);
T1:=T1;
T2:=STAGTEM(T1,T2,NU,1/K,1/O,HOT,L);
IF (T1-T2) < 0.001 THEN (* WALL TEMPERATURE APPROACHED, EXIT PROGRAM *)
BEGIN
F1:=(HOT+CF)/F1+NU*F1);
C2:=CFM(0.01);
F1:=F11K(T);
VOR:='FLUID TEMPERATURE APPROACHES THE WALL TEMPERATURE';
VOR:='AT AN INDETERMINATE LENGTH. INCREASE THE NUMBER OF';
VOR:='SOLUTION SECTIONS. SUGGESTED INCREASE BY FACTOR OF',FACOR11.',':1);
VOR:='OR MORE.);
END:=(VOR)
(* STEPWISE SOLUTION USING RUNGE-KUTTE METHOD *)
(* CALCULATE STEP SIZE *)
STEP:=ABS((T11-T12)/N1);
M1:=M1;
I:=1;
WHILE (I < N) AND (TEST = 0) DO
BEGIN
C1:=F11C1(T0,T,V,F,NU,KEY);
IF C1 = 0 THEN C1:=0;(*FUNCTION UNSTABLE*)
C1:=F11C1(STEP/2.0,T+STEP/2.0,TV,PR,NU,KEY);
IF C1 = 0 THEN C1:=0;(*FUNCTION UNSTABLE*)
C1:=F11C1(STEP/2.0,T+STEP/2.0,TV,PR,NU,KEY);
IF C1 = 0 THEN C1:=0;(*FUNCTION UNSTABLE*)
C1:=F11C1(STEP/2.0,T+STEP/2.0,TV,PR,NU,KEY);
IF C1 = 0 THEN C1:=0;(*FUNCTION UNSTABLE*)
C1:=F11C1(STEP/2.0,T+STEP/2.0,TV,PR,NU,KEY);
IF C1 = 0 THEN C1:=0;(*FUNCTION UNSTABLE*)
C1:=F11C1(STEP/2.0,T+STEP/2.0,TV,PR,NU,KEY);
IF C1 = 0 THEN C1:=0;(*FUNCTION UNSTABLE*)
C1:=F11C1(STEP/2.0,T+STEP/2.0,TV,PR,NU,KEY);
I:=I+1;
IF I = 1 THEN (*FUNCTION UNSTABLE, EXIT LOOP*)
BEGIN
TEST:=1;
M1:=M1;
END:=(VOR)
(* WHILE #)
IF (C1 < 0) OR (C2 < 0) OR (C3 < 0) OR (C4 < 0) THEN
(*FUNCTION UNSTABLE, EXIT LOOP*)
BEGIN
TEST:=1;
M1:=M1;
END:=(VOR)
M1:=M1;
END:=(VOR)
EXX:=H;
A:=R(M)/R(M);
C:=X(1.0)/C(1.0)-1.11);%
F:=FRES(R1(/M));
T:=TEMP(T11,11);
RHO:=DENSIT(T1);
EXX:=T21 Accident12;
IF RE < 2000 THEN
BEGIN (* LINEAR FLOW *)
F:=0.1638RE;
NU:=3.36;
END:=(VOR)
ELSE
BEGIN (* TURBULENT FLOW *)
F:=0.1638RE/(RE,F,G,NU,3.3);
NU:=0.0213RE(FR,F,G,NU,3.3);
END:=(VOR)
Q:=C2*HOT*(T21-T31);
Q:=Q1;
PROCEDURE SECTION(NWALL:INTEGER; VAR NDIV:INTEGER; LS,LV,COUNT: VAR I:DIVISIONS);
(* THIS PROCEDURE DIVIDES THE PIPE LENGTHS INTO NONOVERLAPPING SECTIONS *)
VAR
RESL,LTUT,L:REAL;
COUNT1,COUNT2,I,J,K:INTEGER;
BEGIN (* SECTION *)
(* CALCULATE NOMINAL SECTION SIZE *)
LTOT:=8;
FOR I:=1 TO NWALL DO LTOT:=LTOT+LV[I];
FOR I:=1 TO NWALL+1 DO LTOT:=LTOT+LS[I];
L:=LTOT/NDIV;
(* CALCULATE PIPE SECTIONS *)
LTOT:=0;
I:=0;
J:=0;
K:=0;
COUNT:=1;
COUNT2:=1;
FOR I:=1 TO NWALL+1 DO BEGIN
    I:=I+1;
    IF I = 1 THEN BEGIN
        LTOT:=LTOT+LS[COUNT1];
        COUNT1:=COUNT1+1;
        END (* IF *)
    ELSE BEGIN
        LTOT:=LTOT+LV[COUNT2];
        COUNT2:=COUNT2+1;
        END (* ELSE *)
    WHILE I<=I THEN BEGIN
        J:=J+1;
        END (* WHILE *)
    IF I THEN BEGIN
        RESL:=LTOT+LS[IJ];
        RESL:=RESL+X(J-1); RESL;
        END (* IF *)
    END (* FOR *)
NDIV:=J;
COUNT2:=I:=LTOT+1;
END (* SECTION *)

PROCEDURE SETBOUND(KEY:INTEGER; VAR HIB,LOW,MOOT:REAL);
BEGIN(*SETBOUND*)
    IF KEY = 1 THEN BEGIN
        LOW:=MOOT;
        END (* IF *)
    ELSE BEGIN
        HIB:=MOOT;
        END (* ELSE *)
    MOOT:=(1.*LOW+HIB)/2.0;
    END (*SETBOUND*)

PROCEDURE INREASSIGN;
(* DEFINES VALUES FOR USE IN PARAMETER LIST *)
BEGIN(*INREASSIGN*)
  L:=[XL-IX-11];
  XI:=[XI];
  TR:=[TR];
  PS:=[PS];
  P1:=[PI];
  ENH:=[ENH];
  END (*INREASSIGN*)
PROCEDURE OUTREASSIGN:
(* DEFINES NEW VALUES OBTAINED FROM PARAMETER LIST *)
BEGIN(*OUTREASSIGN*)
  I:=1;
  N1:=N1;
  T01:=T01;
  PI01:=PI01;
  TC01:=TC01;
  R101:=R101;
  H101:=H101;
  E101:=E101;
  N21:=N21;
  Q21:=Q21;
  Q22:=Q22;
END(*OUTREASSIGN*)

PROCEDURE INITIALIZE:
(* PREPARES VARIABLES FOR USE IN MAIN PROGRAM *)
VAR
  J:INTEGER;
BEGIN(*INITIALIZE*)
  NDOTC:=[1];
  (*Determine maximum mass flow*)
  A:=[4.3444];
  C:=[3.20];
  NDOTMAX:=[1*(SDC(DIA)/1.5)*P01*(LAMBDA/10)*T01]*P01*
  (*Check given mass flow and reduce if necessary*)
  (*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*)
  SECTION(NUMWALLS,NORDIV,LSPACE,LWALL,LTOTAL);
  (*Start loop counter*)
  LOOPCOUNT:=0;
  (*Initial Mach number guess*)
  N11:=8.1;
  (*Initialize end Mach number so that it may be used for test*)
  NMINDIV:=0.8;
  (*Initialize bounds*)
  HIROOM:=HROOM;
  LOWROOM:=LOWROOM;
  END(*INITIALIZE*)

PROCEDURE REINITIAL:
(* PREPARES VARIABLES FOR USE IN MAIN PROGRAM *)
VAR J:INTEGER;
BEGIN(*REINITIAL*)
  (*Initialize counters*)
  I:=1;
  PIPETC:=1;
  VALLN:=1;
  SPACECNT:=1;
  (*Increment loop counter*)
  LOOPCOUNT:=LOOPCOUNT+1;
  (*Calculate entrance conditions*)
  EF11:=LSPACE11;
  DX11:=X11-D11;
  RX11:=X11-MDOTC[1]*(LAMBDA(T01)*DIA);
  IF RX11 < 10 THEN
    BEGIN(*LAMINAR FLOW*)
    T11:=[4.0];
    N11:=8.46;
    END(*IF*)
  ELSE
    BEGIN(*Turbulent flow*)
    T11:=[16.0];
    N11:=8.46;
    END(*IF*)
  END(*REINITIAL*)
ORIGINAL PAGE IS OF POOR QUALITY

BEGIN (* TURBULENT FLOW *)
  F21:=[8.866/PVR(KE1(1),0.5)];
  NU1:=[8.823/PVR(KE1(1),0.5)PVR(PRANDTL(KE1(1)),0.8)];
END;(*ELSE*)

(*CALCULATE INLET CONDITIONS*)
INLETMACH(DIA,MDOTLOOPCOUNT),PS1(1),Thu(1),N(1),TE1(1),F1(1),RHO(1));
(*INITIALIZE CONTROL CARD*)
NTEST:=-1;
END;(*ENDINITIAL*)

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

REPEAT (* UNTIL THE FLOW IS CHOKE *)
  REINITIAL;
  WHILE (((NUMDIV) AND (NU1) (= 0.97)) DO
  BEGIN
  (* BEGIN FRICTION FLOW CALCULATIONS *)
  WHILE (XI1) (= LPIPE(FIRONMENT)) AND (NU1) (= 0.97) DO
  BEGIN
    INREASSGN:
    FRICTION(T9,DIA,T1-L1,T1,F1,RHO1,F1.01);
    OUTREASSGN;
    END;(*WHILE*)
  IF VALUE1 (= NUMWALLS THEN
  BEGIN
    FIRONMENT:=FICIENT-1;
    SPACENT:=SPACENT+1;
    LPIPE(FICIENT):=LPIPE(FICIENT-1)-LVALL(VALLCNT);
    END;(*IF*)
  (* BEGIN NEAR TRANSFERS CALCULATIONS *)
  TSAVE:=T1;
  EFF(VALLCNT):=0;
  GSAVE:=0;
  WHILE (EFF(VALLCNT) >= MAXEFF) AND (XI1) (= LPIPE(FICIENT)) AND (NU1) (= 0.97) DO
  BEGIN
    INREASSGN:
    NEATRANS(T(VALLCNT),L,DIA,MDOTLOOPCOUNT),N,T1,F1,RHO1,R1,F1.01,
    OUTREASSGN;
    EFF(VALLCNT):=T(VALLCNT)-TSAVE)/T(VALLCNT)-TSAVE;
    END;(*WHILE*)
  IF VALUE1 (= NUMWALLS THEN
  BEGIN
    FICIENT:=FICIENT+1;
    VALLCNT:=VALLCNT+1;
    LPIPE(FICIENT):=LPIPE(FICIENT-1)+LSPACE(SPACECNT);
    END;(*IF*)
  END;(*WHILE*)
  IF NUMDIV) (= THEN NTEST:=2;
  MDOTI:=MDOTLOOPCOUNT;
  SETBOUND(NTEST,HBOUND,LOBOUND,BOUND);
  MDOTLOOPCOUNT:=MDOTI;
  UNFIL (MINUMDIV) (= 0.97) AND (MINUMDIV) (= 1.63) OR (LOOPCOUNT = 30);
END;(*END MAIN PROGRAM *)
TO ENTER DATA, ENTER 1. OTHERWISE, ENTER 0.

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
? 0.024
? 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)
? 6.23

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)
? 0.00000143

MASS FLOW (KG/S) = 0.000001430

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 CF SECS, 1014668 CM USED.
### Values of Input

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dev. Stagnation Temperature (T)</td>
<td>1.3000000000000E+00</td>
</tr>
<tr>
<td>Dev. Stagnation Pressure (P)</td>
<td>2.0000000000000E-01</td>
</tr>
<tr>
<td>Mass Flow Approximation (kg/s)</td>
<td>1.0000000000000E+00</td>
</tr>
</tbody>
</table>

### Shield Temperatures (T)

<table>
<thead>
<tr>
<th>Section</th>
<th>Temperature</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>31.7000000000</td>
</tr>
<tr>
<td>2</td>
<td>39.7000000000</td>
</tr>
<tr>
<td>3</td>
<td>45.0000000000</td>
</tr>
<tr>
<td>4</td>
<td>101.0000000000</td>
</tr>
<tr>
<td>5</td>
<td>144.4000000000</td>
</tr>
</tbody>
</table>

### Values of Output

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mass Flow (kg/s)</td>
<td>0.00000000000000</td>
</tr>
<tr>
<td>Total Heat Transfer (W)</td>
<td>0.00000000000000</td>
</tr>
</tbody>
</table>

### Shield Heat Transfer (W)

<table>
<thead>
<tr>
<th>Section</th>
<th>Heat Transfer</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>20.1000000000</td>
</tr>
<tr>
<td>2</td>
<td>19.7000000000</td>
</tr>
<tr>
<td>3</td>
<td>19.1000000000</td>
</tr>
<tr>
<td>4</td>
<td>19.5000000000</td>
</tr>
<tr>
<td>5</td>
<td>19.9000000000</td>
</tr>
</tbody>
</table>

### Shield Heat Exchanger Effectiveness

<table>
<thead>
<tr>
<th>Section</th>
<th>Effectiveness</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.143247712</td>
</tr>
<tr>
<td>2</td>
<td>0.463401115</td>
</tr>
<tr>
<td>3</td>
<td>0.304015674</td>
</tr>
<tr>
<td>4</td>
<td>0.764015674</td>
</tr>
</tbody>
</table>

### Length, Mass Number, Stagnation Temperature, Stagnation Pressure, Temperature, Pressure, Nu, Reynold, Section Heat, Total Heat

<table>
<thead>
<tr>
<th>Length</th>
<th>Mass Number</th>
<th>Stag Temp</th>
<th>Stag Pres</th>
<th>Temp</th>
<th>Pressure</th>
<th>Nu</th>
<th>Reynold</th>
<th>Section Heat</th>
<th>Total Heat</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.06652</td>
<td>0.647998467</td>
<td>4.330000</td>
<td>1160.60000</td>
<td>4.32377</td>
<td>1379.4199</td>
<td>3.660</td>
<td>657.9517</td>
<td>0.00000000</td>
<td>0.10000000</td>
</tr>
<tr>
<td>0.06852</td>
<td>0.647998467</td>
<td>4.330000</td>
<td>1160.60000</td>
<td>4.32377</td>
<td>1379.4199</td>
<td>3.660</td>
<td>657.9517</td>
<td>0.00000000</td>
<td>0.10000000</td>
</tr>
<tr>
<td>0.06972</td>
<td>0.647998467</td>
<td>4.330000</td>
<td>1160.60000</td>
<td>4.32377</td>
<td>1379.4199</td>
<td>3.660</td>
<td>657.9517</td>
<td>0.00000000</td>
<td>0.10000000</td>
</tr>
<tr>
<td>0.11011</td>
<td>0.647998467</td>
<td>4.330000</td>
<td>1160.60000</td>
<td>4.32377</td>
<td>1379.4199</td>
<td>3.660</td>
<td>657.9517</td>
<td>0.00000000</td>
<td>0.10000000</td>
</tr>
</tbody>
</table>

...
<table>
<thead>
<tr>
<th>INTERATION</th>
<th>MASS FLOW</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
</tr>
<tr>
<td>2</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
</tr>
<tr>
<td>3</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
</tr>
<tr>
<td>4</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
</tr>
<tr>
<td>5</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
</tr>
<tr>
<td>6</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
</tr>
<tr>
<td>7</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
</tr>
<tr>
<td>8</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
</tr>
</tbody>
</table>
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

\[ \dot{m} = \rho VA. \]  

(B.1)

The definition of Mach number is

\[ M = \frac{V}{\sqrt{\gamma RT}} \]  

(B.2)

Combined with (B.1), this yields

\[ \dot{m} = (\gamma RT)^{1/2} \rho MA \]  

(B.3)

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

\[ \rho = \rho_0 \left( 1 + \frac{\gamma - 1}{2} M^2 \right)^{-1/(\gamma - 1)} \]  

(B.4)

The ideal gas equation states

\[ \rho_0 = \frac{P_0}{RT_0} \]  

(B.5)

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

\[ \rho = \frac{P_0}{RT_0} \left[ 1 + \frac{\gamma - 1}{2} M^2 \right]^{-1/(\gamma - 1)} \]  

(B.6)

Similarly, T may be expressed in terms of M and T₀

\[ T = T_0 \left( 1 + \frac{\gamma - 1}{2} M^2 \right)^{-1} \]  

(B.7)

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

\[ \dot{m} = \frac{(\gamma/RT_0) \rho_0}{\left( 1 + \frac{\gamma - 1}{2} M^2 \right)^{\left( \frac{\gamma + 1}{2} \right)}} \left( \frac{\gamma + 1}{2(1 - \gamma)} \right) \]  

(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

\[ M_{\text{new}} = M_{\text{old}} - \frac{f(M_{\text{old}})}{f'(M_{\text{old}})} \]  

(B.9)

where

\[ f(M) = \frac{(\gamma/RT_0)^{1/2} A_0}{1 + \frac{\gamma - 1}{2} M^2} \left( \frac{\gamma + 1}{2(1 - \gamma)} \right) - \dot{m} \]  

(B.10)
\[ f'(M) = \frac{A P_0}{(\gamma/R T_0)^{1/2}} B^{[\gamma+1]/2[1-\gamma]} \left[ 1 - M^2 \right] B^{-1} \]  \hspace{1cm} (B.11)

B is an often used function of M and \( \gamma \).

\[ B = 1 + \left[ (\gamma - 1)/2 \right] M^2 \]  \hspace{1cm} (B.12)
APPENDIX C: FRICTION FLOW MACH NUMBER CALCULATION

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

\[ \frac{dM}{M} = \left( \frac{\gamma M^2 B_1}{2(1 - M^2)} \right) \left( \frac{4f}{D} \right) dx \]  

(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

\[ \frac{4f(x_2 - x_1)}{D} = \left( \frac{M_2^2 - M_1^2}{\gamma M_1^2 M_2^2} \right) + \left( \frac{\gamma + 1}{2\gamma} \right) \ln \left( \frac{M_1 B_1}{M_2 B_2} \right) \]  

(C.2)

Solve for \( M_2 \) with the Newton-Raphson method using

\[ f(M_2) = \left[ \frac{(M_2^2 - M_1^2)}{\gamma M_1^2 M_2^2} \right] \]

\[ + \left( \frac{\gamma + 1}{2\gamma} \right) \ln \left( \frac{M_1 B_1}{M_2 B_2} \right) - \frac{4fL}{D} \]  

(C.3)

\[ f'(M_2) = \left[ \frac{2}{\gamma M_2^3} + \left( \frac{\gamma + 1}{2\gamma} \right) \frac{B_1^2}{M_1^4 B_2^2} \right] \]  

(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

\[ \dot{m} \, dq = h(T_w - T_{aw}) \pi D \, dx \]  \hspace{1cm} (D.1)

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

\[ dq = c_p \, dT_o \]  \hspace{1cm} (D.2)

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

\[ \dot{m} \, c_p \, dT_o = h(T_w - T_{aw}) \pi D \, dx \]  \hspace{1cm} (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

\[ \dot{m} \, c_p \, dT_o = h(T_w - T_o) \pi D \, dx \]  \hspace{1cm} (D.4)

One may recall that the Nusselt number is defined as

\[ Nu = hD/k_f \]  \hspace{1cm} (D.5)

Equation (D.4) may be then expressed as

\[ \dot{m} \, c_p \, dT_o = \pi \, Nu \, k_f (T_w - T_o) \, dx \]

\[ dT_o/(T_w - T_o) = (\pi \, Nu \, k_f)/(\dot{m} \, c_p) \, dx \]  \hspace{1cm} (D.6)

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

\[ T_o = T_{o1} \text{ at } x = x_1 \text{ and } T_o = T_{o2} \text{ at } x = x_2 \]

yields

\[ T_{o2} = T_w - \{[T_w - T_{o1}]/[\exp(\pi \, Nu \, k_f L/\dot{m} \, c_p)]\} \]  \hspace{1cm} (D.7)

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

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

One may express these relations in terms of dimensionless parameters.

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

For laminar flow,

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

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

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

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

\[
dT_o / (T_w - T_o) = 4(f/2) \, dx/D \tag{E.9}
\]

So that for turbulent flow,

\[
\frac{dM}{M} = \frac{\gamma M^2 B}{2(1 - M^2)} \frac{2 \, dT_o}{T_w - T_o} + \frac{(1 + \gamma M^2) B}{2(1 - M^2)} \frac{dT_o}{T_o} \tag{E.10}
\]
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.

\[ M_{n+1} = M_n + \Delta T_0 / 6 (c_1 + 2c_2 + 2c_3 + c_4) \]  

where

\[ c_1 = F(T_{on}, M_n) \]
\[ c_2 = F(T_{on} + \Delta T_0 / 2, M_n + c_1 \Delta T_0 / 2) \]
\[ c_3 = F(T_{on} + \Delta T_0 / 2, M_n + c_2 \Delta T_0 / 2) \]
\[ c_4 = F(T_{on} + \Delta T_0, M_n + c_3 \Delta T_0) \]  

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

\[ F(T_{on}, M_n) = \frac{dM}{dT_0} \mid_{n} \]  

For laminar flow,

\[ F = \frac{dM}{dT_0} = \frac{MB}{2(1 - M^2)} \left[ Y M^2 \left( \frac{16 Pr}{N u D (T_w - T_0)} \right) + \frac{1 + \gamma M^2}{T_0} \right] \]  

For turbulent flow,

\[ F = \frac{dM}{dT_0} = \frac{BM}{2(1 - M^2)} \left[ Y M^2 \left( \frac{2}{T_w - T_0} \right) + \frac{1 + \gamma M^2}{T_0} \right] \]
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:

\[ k_f = 0.0157218071 + 6.3416525 \times 10^{-4} T - 6.99407521 \times 10^{-7} T^2 + 4.52003096 \times 10^{-10} T^3 \]  
\[ (G.1) \]

Viscosity:

\[ \mu = 1.10419265531 \times 10^{-6} + 1.160677594648 \times 10^{-7} T - 3.222922924062 \times 10^{-10} T^2 + 4.725700293922 \times 10^{-13} T^3 \]  
\[ (G.2) \]

Prandtl number:

\[ Pr = 0.570232994 + 0.0055872291 T - 7.58324532 \times 10^{-5} T^2 + 4.32720501 \times 10^{-7} T^3 - 1.11661445 \times 10^{-9} T^4 + 1.07403542 \times 10^{-12} T^5 \]  
\[ (G.3) \]
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 $A_2$.

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

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

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

Solve for $M_2$ with a Newton-Raphson method.

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

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