

# ornl

ORNL/TM-9957

OAK RIDGE  
NATIONAL  
LABORATORY

MARTIN MARIETTA

**Improvements and Enhancements  
of the Absorb Computer Program  
for Modeling Chemical Absorption  
Heat Pump Systems**

Richard L. Cox

OPERATED BY  
MARTIN MARIETTA ENERGY SYSTEMS, INC.  
FOR THE UNITED STATES  
DEPARTMENT OF ENERGY

ORNL/TM-9957

Energy Division

IMPROVEMENTS AND ENHANCEMENTS OF THE *ABSORB* COMPUTER PROGRAM  
FOR MODELING CHEMICAL ABSORPTION HEAT PUMP SYSTEMS

Richard L. Cox  
Computing and Telecommunications Division

Date of Issue-July 1986

Prepared for the  
Office of Industrial Programs

Prepared by the  
Oak Ridge National Laboratory  
Oak Ridge, Tennessee 37831  
operated by  
MARTIN MARIETTA ENERGY SYSTEMS, INC.  
for the  
U.S. DEPARTMENT OF ENERGY  
under Contract No. DE-AC05-84OR21400



## CONTENTS

	Page
ABSTRACT .....	1
INTRODUCTION .....	1
DISCUSSION .....	1
SOLUTION OF EQUATIONS .....	1
MODULARIZATION OF THE CODE .....	3
ECONOMIC OPTIMIZATION OF THE SYSTEM DESIGN .....	4
RESULTS AND CONCLUSIONS .....	4
ACKNOWLEDGMENTS .....	7
REFERENCES .....	9
Appendix. LISTING OF ABSORB COMPUTER PROGRAM .....	11



IMPROVEMENTS AND ENHANCEMENTS OF THE *ABSORB* COMPUTER PROGRAM  
FOR MODELING CHEMICAL ABSORPTION HEAT PUMP SYSTEMS

Richard L. Cox

ABSTRACT

The computer code **ABSORB** is a simulation program specifically designed to model chemical absorption heat pump systems of varying configuration. In a continuing effort to improve the robustness, flexibility, and applicability of this code, a number of improvements and enhancements have been recently incorporated into the code. It is the purpose of this report to update the documentation of the code by describing these modifications which include: (1) a revised strategy of solving the system equations, (2) increased modularization of the program, and (3) the first efforts to employ the code to determine an optimum economic design of a heat pump system.

INTRODUCTION

The computer program **ABSORB** (for simulating chemical absorption heat pumps) has been described previously<sup>1</sup>. The present paper is concerned with describing modifications to this code. The alterations to the code can be generally categorized as follows:

1. incorporation of an alternative approach to the solution of the nonlinear algebraic system of equations modeling the heat pump;
2. further modularization of the code serving to streamline the program, improve its readability, and increase the ease of modification and enhancement; and
3. modifications to permit use of the program to perform economic optimization of the heat absorber system.

DISCUSSION

SOLUTION OF EQUATIONS

The previous approach to the solution of the system equations, while generally effective, possessed certain shortcomings. The greatest of these centered around the measures required to guarantee a physically meaningful mathematical solution. These measures involved intervening in the solution process to ensure that the variables satisfied certain bound and inequality constraints in addition to the equality constraints solved by a nonlinear equation solver. The new approach uses a nonlinear optimization program rather than a nonlinear equation solver to solve the system equations. The nonlinear optimization routine chosen, NPSOL<sup>2</sup>, permits one to specify bounds on the variables and inequality relationships among the variables as a part of the problem statement. Thus, bound and

inequality constraints, which previously were imposed external to a nonlinear equation solver and were troublesome to handle and difficult to ensure, are now an integral part of the problem specification to an optimizer specifically designed to efficiently treat such constraints.

A second shortcoming of the previous approach was the need to recognize and eliminate redundant constraints, a task that becomes increasingly difficult as the number of components (absorbers, desorbers, condensers, evaporators, etc.) making up the process system increases. Whereas the elimination of redundant constraints is imperative when using a nonlinear equation solver, redundant constraints are of much less concern and often are readily handled by a robust optimization program. This fact makes it possible to combine the equations modeling individual components into a system of equations modeling the heat pump without concern for the independence of the resulting set. This is a major advantage since the basic motivation for the development of the code was the desire to produce a simulation code to model heat pump systems of arbitrary configuration, predicated upon viewing each heat pump as consisting of a number of standard components.

If the optimizer is being used to simply solve a set of nonlinear equations with constraints, and if the bound and inequality constraints ensure a unique solution, then the feasible region for the optimization is a single point. Hence, an objective function is immaterial to the problem; and the usual two-part process (within the optimizer) of, first, determining a feasible point and of, second, determining that feasible point which minimizes (or maximizes) the objective function reduces to that of finding a feasible solution. Because of the irrelevance of the objective function, it was set, for simplicity, to 1 in the problem specification.

The use of NPSOL does, however, in contrast to HYBRD1 (the nonlinear equation solver used in the previous version of the code), require the user to supply the first derivatives of all the constraint functions and also of the objective function. The fact that the system equations are relatively simple algebraic expressions and the fact that analytical expressions are available for the thermodynamic properties of lithium bromide-water solutions (the working fluid) made it comparatively easy to determine all the required first derivatives. Although the version of the code presented in this report is restricted to lithium bromide-water solutions, this does not represent an inherent limitation of the code, but rather the lack of analytical expressions for the thermodynamic properties of other working media of interest. The modeling of working fluids for which the thermodynamic property data are available only in tabular form will require either that the tabular data be numerically differentiated or that the data be approximated by differentiable analytical functions.

Actually, first derivatives are also required by HYBRD1 but are calculated internal to HYBRD1 by finite differencing. (There also exists a version of HYBRD1, namely HYBRJ1, which requires the user to supply a subroutine to calculate first derivatives analytically. When this is feasible to do, HYBRJ1 is to be preferred to HYBRD1 as it may be expected to be more robust, avoiding the inaccuracies associated with finite difference approximation of derivatives.) Seen from this point of view, the optimization package requires no more information than the nonlinear equation solver. Also, it may be pointed out that there exist optimization routines which do not require user specification of the first derivatives, but at present they represent an older generation of codes which are far inferior to newer codes such as NPSOL.

In order to use the optimization package effectively, it was necessary to scale the variables, the constraints, and the objective function. The desirability and, often, the necessity for scaling arises from the fact that within optimization routines convergence tolerances and other criteria are necessarily based upon an implicit definition of "small" and "large", with the consequence that variables of widely varying orders of magnitude cause difficulties for the algorithms. The ideal usually sought in scaling is for the variables to be of order unity and the constraint and objective functions to vary by approximately one unit when a variable is changed by one unit.

The simplest procedure for scaling the variables and that used in the current version of the code is to use a linear transformation of the form

$$y = \lambda x$$

for each of the variables, where  $x$  is the original variable; and  $\lambda$  is a scale factor chosen to yield a transformed variable  $y$  of order unity. A further simplification is to choose the same value of  $\lambda$  for variables of the same type as, for example, a fixed value of  $\lambda$  for each of the temperature variables. Using this approach to scaling, it was fairly straightforward to rederive the system equations in terms of transformed variables. Indeed, for the linear constraints and even for some of the nonlinear constraints, the form of the original and transformed equations were identical with only the name of the variables changing. Thus, for example, consider the overall material balance for any unit in the system which may be written

$$\sum_{i=1}^N F_i = 0 ,$$

where the summation is over the  $N$  flows  $F_i$  entering and exiting the unit, with the flows appropriately signed. Then we have

$$\sum_{i=1}^N (\lambda_F F_i) = 0 ,$$

or

$$\sum_{i=1}^N \hat{F}_i = 0 ,$$

where  $\hat{F}_i$  denotes a transformed flow variable; and  $\lambda_F$  is the scaling factor for the flows. Once the derivation of the transformed equations has been performed and programmed, the use of scaled variables and constraints is invisible to the user with the exception that he must supply, as part of the input data set, appropriate values of the scale factors--a total of five values currently, one each for the temperature, flow, concentration, pressure, and vapor-fraction variables.

## MODULARIZATION OF THE CODE

In the process of modifying the code to compute first derivatives, it was recognized that the program could be revised to increase its modularity and thereby its readability and, particularly, its adaptability in regard to future modifications and enhancements. This was achieved principally by employing separate routines to generate the constraint equations and their first derivatives. Thus, the overall material balance, component material balance, enthalpy balance, heat exchanger energy balance, and the equilibria calculations are performed in separate subroutines called by the unit (absorber, desorber, etc.) subroutines instead of being calculated in line in each unit subroutine.

## ECONOMIC OPTIMIZATION OF THE SYSTEM DESIGN

As a first step in the development and demonstration of a computer code to perform an optimal detailed economic design of a heat pump system, the code was enhanced to optimize a heat pump system operating as a temperature amplifier (or heat transformer). In this type of system, the heat pump takes in waste heat at a low temperature, rejects a portion to a heat sink, and upgrades the remainder to a higher temperature for use as process heat.

A measure of the efficiency of a temperature amplifier is the coefficient of performance (COP) of the system which is defined as the ratio of the output process heat to the input waste heat. The financial return for a heat transformer is directly proportional to the units of process heat delivered and may be quantified in terms of the cost of supplying the process heat by alternate means as, for example, using steam. The capital cost of the system is essentially the sum of the capital costs of each of the component units. The capital cost of each of the components, in turn, may be taken as directly proportional to the heat exchanger area of the component, this being a measure of the size of the component.

Increasing the areas of the heat exchangers can increase the COP, but this also increases the capital cost of the system. The optimization problem consists of selecting heat exchanger areas for each of the system components so as to provide the optimum combination of capital cost and COP, thus minimizing the delivered energy cost. One possible measure of optimality is payback time which is defined as

$$\text{payback time, years} = \frac{\text{capital cost of heat pump system}}{\text{value of process heat produced/year}}$$

If, as a first approximation, the cost-per-unit of heat exchanger area is assumed constant for the various components in the system, then the payback time is directly proportional to the ratio

$$\frac{\sum \text{heat exchanger areas}}{\text{rate of process heat production}}$$

It is the above ratio which, at present, is minimized by the code in the search for the optimum economic operating conditions for a heat pump employed as a heat transformer. Estimates of the per-unit-cost of heat exchanger area and of the per-unit-value of process heat would, of course, allow one to quantify the payback period.

## RESULTS AND CONCLUSIONS

The new version of the code was used to repeat calculations which had been performed previously by the old version and reported<sup>1</sup>. In each instance, the computed results were identical and were obtained in less computer time when the same initial guesses were chosen, thus proving the efficacy of the new approach. Moreover, the new code has the added capability of performing optimization calculations. A FORTRAN listing of the code (excluding the optimizing package NPSOL which is proprietary and must be obtained under license from Stanford University) is given in the Appendix.

One area in which more work appears to be needed on the code is that of greater robustness as regards convergence from poor starting guesses. Actually, more use of the code is needed to define the extent to which this added robustness is necessary, the underlying question being whether the user will have available, or should be expected to supply, a "reasonably good" estimate of the solution. The older version of the code was somewhat more robust than the present version in that it incorporated a preprocessor to massage the input data and thereby permitted poorer initial guesses than the present version. In the interest of simplifying the code and because the preprocessor was somewhat heuristic, it was stripped from the new version of the code, with the intent of restoring it only if it proved to be a clearly desirable feature. In many instances, a good starting estimate for a given problem can be defined from the solution of a previous problem.

Although the need for continued development of the code in terms of incorporating models of new system components, thermodynamic description of other working fluids, and more detailed economic analysis is readily recognized, it is believed that the basic framework of a code, sound in strategy, efficient in execution, and flexible in application is now in place. Clearly, application of the program to problems of interest will guide the immediate and future evolution of the code.



#### ACKNOWLEDGMENTS

Appreciation is expressed to Gershon Grossman (Haifa Technion University) for involving and guiding the author in this work on his ABSORB code. Appreciation is also expressed to Moonis Ally and Stephen Kaplan (Energy Division) for their editorial comments which served to strengthen the report, and to Irene Rose (C&TD) for her fine job of typesetting the report.

This work was funded by the Office of Industrial Programs of the Department of Energy.



## REFERENCES

1. G. Grossman and E. Michelson, *Absorption Heat Pump Simulation and Studies, Part I: A Modular Computer Simulation of Absorption Systems*, ORNL/Sub/43337/2, to be published at Oak Ridge National Laboratory.
2. P. Gill, W. Murray, M. Saunders, and M. Wright, *User's Guide for NPSOL (Version 2.1): A Fortran Package for Nonlinear Programming*, SOL 84-7, Stanford University, Department of Operations Research, Stanford, Calif., September 1984.



## **APPENDIX**

### **LISTING OF *ABSORB* COMPUTER PROGRAM**

```

C THIS IS THE ABSORB COMPUTER PROGRAM FOR MODELING
C HEAT PUMP SYSTEMS OF VARYING CONFIGURATION
C
C VERSION OF APRIL 21, 1986
C
C NSP      = NUMBER OF STATE POINTS
C NUNITS   = NUMBER OF UNITS IN SYSTEM
C N          = NUMBER OF VARIABLES
C NCLIN    = NUMBER OF LINEAR CONSTRAINTS
C NCNLN    = NUMBER OF NONLINEAR CONSTRAINTS
C NCON     = NCLIN + NCNLN
C NCTOTL   = N + NCLIN + NCNLN
C
C DIMENSIONING INFORMATION:
C
C NROWA    = DECLARED ROW DIMENSION OF ARRAY A WHICH MUST BE
C             GREATER THAN OR EQUAL TO (.GE.) NCLIN
C NROWJ    = DECLARED ROW DIMENSION OF ARRAY CJAC WHICH MUST BE
C             .GE. NCNLN
C NROWR    = DECLARED ROW DIMENSION OF ARRAY R WHICH MUST BE
C             .GE. N
C LENIW    = DECLARED LENGTH OF VECTOR IWORK WHICH MUST BE .GE.
C             3*N + NCLIN + NCNLN
C LENW     = DECLARED LENGTH OF VECTOR WORK WHICH MUST BE .GE.
C             2*N*N + N*(NCON + 12) + 7*NCNLN + NROWA
C LENB     = INTEGER VARIABLE WHICH MUST BE SET .GE. NCTOTL
C
C VECTORS IN COMMON BLOCKS AA, BB, CC, DD AND THE VECTORS
C JC, JF, JP, JT, JW MUST BE DIMENSIONED .GE. NSP
C VECTORS IN COMMON BLOCKS EE, FF (FIRST DIMENSION OF ISP)
C MUST BE DIMENSIONED .GE. NUNITS
C VECTORS IN COMMON BLOCK GG AND THE VECTORS FEATOL, CLAMDA,
C ISTATE MUST BE DIMENSIONED .GE. NCTOTL
C ARRAY A MUST BE DIMENSIONED A(NRCWA,NCOLA), NCOLA .GE. N
C ARRAY CJAC MUST BE DIMENSIONED CJAC(NROWJ,NCCLJ) WITH
C NCOLJ .GE. N
C ARRAY R MUST BE DIMENSIONED R(NROWR,NCOLR), NCOLR .GE. N
C VECTOR FUN MUST BE DIMENSIONED .GE. NCNLN
C VECTORS OBJGRD, X MUST BE DIMENSIONED .GE. N
C VECTOR WORK MUST BE DIMENSIONED EQUAL TO LENW
C VECTOR IWORK MUST BE DIMENSIONED EQUAL TO LENIW
C
C INTEGER VARIABLES NROWA, NROWJ, NRCWR, LENIW, LENW, LENB
C MUST BE EXPLICITLY SET BELOW
C
C DIMENSIONS BELOW ARE FOR MAX. OF 50 STATE PTS. AND 20 UNITS.
C HOWEVER, SUBJECT ONLY TO THE LIMITATIONS OF COMPUTER MEMORY,
C THE PROGRAM MAY BE USED FOR SYSTEMS OF AN ARBITRARY NUMBER
C OF UNITS AND STATE POINTS BY APPROPRIATE REDIMENSIONING IN
C ACCORDANCE WITH THE INSTRUCTIONS GIVEN ABOVE.
C
C IMPLICIT REAL*8 (A-H,O-Z)
C LOGICAL COLD,ORTHO
C COMMON//AA/ C(50),F(50),P(50),T(50),Y(50),IVARC(50),
C > IVARF(50),IVARP(50),IVART(50),IVARY(50),KFAZ(50),MSUB
C COMMON//BB/ ICFX(50),IFFX(50),IPFX(50),ITFX(50),IYFX(50),
C > IVC(50),IVF(50),IVP(50),IVT(50),IVY(50),IVA(50)
C COMMON//CC/ H(50),DHC(50),DHP(50),DHT(50),DHY(50)
C COMMON//DD/ GRAD(50),OBJECT,ICBJ
C COMMON//EE/ Q(20),UA(20),IREV(20),IAFX(20),IVARA(20)
C COMMON//FF/ IDUNIT(20),ISP(20,7),NSP,NUNITS,NUC,NUF,
C > NUP,NUT,NUY,NUA
C COMMON//GG/ BL(125),BU(125)
C COMMON//NN/ NROWA,NROWJ,NL,NNL,INIT
C COMMON//SS/ CSCALE,FSCALE,PSCALE,TSCALE,YSCALE
C COMMON//ZZ/ UROUND
C
C DIMENSION A(25,50),CJAC(50,50),R(50,50)

```

```

DIMENSION FUN(50),OBJGRD(50),X(50)
DIMENSION WORK(7000),IWORK(225)
DIMENSION FEATOL(125),CLAMDA(125),ISTATE(125)
DIMENSION JC(50),JF(50),JP(50),JT(50),JY(50)
DIMENSION ANAME(8)

C      EXTERNAL CONFUN,OBJFUN
DATA BIGBND/2.0D0/,ETA/0.9D0/,COLD/.TRUE./,ORTHOG/.TRUE./
DATA ANAME/1HA,1HD,1HR,1HC,1HE,1HV,1FM,1HS/

C FOLLOWING STATEMENT IS NEEDED FOR THE CRAY COMPUTER
C CALL LINK ("UNITS=INFILE,UNIT6=TERMINAL //")

C SET DIMENSIONING DECLARATIONS
NROWA = 25
NROWJ = 50
NROWR = 50
LENIW = 225
LENW = 7000
LENB = 125

C CALCULATE UNIT ROUNDOFF OF COMPUTER (MACHINE EPSILON)
UROUND = D1MACH(4)

C INPUT AND PRINT OPTIMIZER FLAGS AND PUNCH FLAG
C IOBJ = OBJECTIVE FUNCTION FLAG
C     = 1 IF OBJ. IS ONE
C     = 2 IF OBJ. IS SUM OF SQUARES OF THE EQUALITY
C          CONSTRAINTS
C     = 3 IF OBJ. IS SUM OF UNKNOWN UA'S
C     = 4 IF OBJ. IS PROPORTIONAL TO PAYBACK TIME
C ITMAX = ALLOWABLE NUMBER OF ITERATIONS
C MSGLVL = PRINT LEVEL FLAG FOR INTERMEDIATE OUTPUT
C IPUN = PUNCH FLAG
C     = 0 NO PUNCHED OUTPUT
C     = 1 PUNCHED OUTPUT
READ (5,1000) IOBJ,ITMAX,MSGLVL,IPUN
WRITE (6,2000) IOBJ,ITMAX,MSGLVL,IPUN
IF (IPUN .EQ. 1) WRITE (7,1000) IOBJ,ITMAX,MSGLVL,IPUN

C INPUT AND PRINT ERROR TOLERANCES.  IF ZERO VALUES ARE INPUT,
C DEFAULT VALUES WILL BE CALCULATED AND USED.
C CTOL = ERROR TOLERANCE FOR CONSTRAINT FUNCTIONS
C FTOL = ERROR TOLERANCE FOR OBJECTIVE FUNCTION AT SOLUTION
C EPSAF = ESTIMATED ERROR IN COMPUTING OBJECTIVE FUNCTION
READ (5,1100) CTOL,FTOL,EPSAF
IF (CTOL .EQ. 0.0D0) CTOL = 10.0D0*DSQRT(UROUND)
IF (EPSAF .EQ. 0.0D0) EPSAF = 10.0D0*UROUND
IF (FTOL .EQ. 0.0D0) FTOL = DSQRT(EPSAF)
WRITE (6,2101) CTOL,FTOL,EPSAF,UROUND
IF (IPUN .EQ. 1) WRITE (7,1000) CTOL,FTOL,EPSAF,UROUND

C INPUT AND PRINT SCALE FACTORS FOR CONCENTRATIONS, FLOWS,
C PRESSURES, TEMPERATURES, VAPOR FRACTIONS.  SCALE FACTORS
C SHOULD BE CHOSEN SUCH THAT PRODUCT OF SCALE FACTOR AND
C VARIABLE IS OF ORDER OF MAGNITUDE ONE.
READ (5,1100) CSCALE,FSCALE,PSCALE,TSCALE,YSCALE
WRITE (6,2102) CSCALE,FSCALE,PSCALE,TSCALE,YSCALE
IF (IPUN .EQ. 1) WRITE (7,1100) CSCALE,FSCALE,PSCALE,
> TSCALE,YSCALE

C INPUT AND PRINT HEAT PUMP TYPE AND WORKING FLUID
C MAN=1 - ONE STAGE HEAT PUMP   : FOR PRINTING
C MAN=2 - ONE STAGE CHILLER    : OUT TITLE ONLY
C MAN=3 - TWO-STAGE HEAT PUMP  :
C MAN=4 - TWC-STAGE CHILLER   "
C

C MSUB=1 - LI-BR/WATER

```

```

C  MSUB=2 - ALTERNATIVE FLUID (NOT VALID AT PRESENT)
C      READ (5,1000) MAN,MSUB
C      GO TO (1,2,3,4), MAN
C
C      1 WRITE (6,2001)
C          GO TO 5
C      2 WRITE (6,2002)
C          GO TO 5
C      3 WRITE (6,2003)
C          GO TO 5
C      4 WRITE (6,2004)
C
C      5 GO TO (6,7), MSUB
C
C      6 WRITE (6,2006)
C          GO TO 8
C      7 WRITE (6,2007)
C
C INPUT AND PRINT NUMBER OF UNITS AND NUMBER OF STATE POINTS
C 8 READ (5,1000) NUNITS,NSP
C          WRITE (6,2008) NUNITS,NSP
C
C INPUT AND PRINT UNIT VARIABLES
C IUNIT = UNIT NUMBER (IN ORDER OF INPUT)
C IDUNIT = UNIT IDENTIFICATION NUMBER
C          = 1 FOR ABSORBER
C          = 2 FOR DESORBER
C          = 3 FOR RECUPERATOR
C          = 4 FOR CONDENSER
C          = 5 FOR EVAPORATOR
C          = 6 FOR VALVE
C          = 7 FOR MIXER
C          = 8 FOR SPLITTER
C UA    = OVERALL HEAT TRANSFER COEFF. TIMES HEAT EX. AREA
C IREV   = FLAG TO INDICATE REVERSED HEAT FLOW IN RECUPERATOR
C          = 0 IF HEAT FLOW IN FORWARD DIRECTION
C          = 1 IF HEAT FLOW IN REVERSE DIRECTION
C IAFX   = FLAG TO INDICATE STATUS OF UA VALUE
C          = 0 IF UA VALUE IS FIXED
C          = 1 IF UA VALUE IS UNKNOWN
C ISP    = VECTOR OF STATE POINTS FOR UNIT IN THE ORDER
C          CORRESPONDING TO THE TEMPLATE FOR THE UNIT
C          WRITE (6,2010)
C          DO 11 I=1,NUNITS
C
C          READ (5,1011) IUNIT, IDUNIT(I), IREV(I), IAFX(I), UA(I),
C          > (ISP(I,J), J=1,7)
C          WRITE (6,2011) IUNIT, IDUNIT(I), IREV(I), IAFX(I), UA(I)
C          IF (IPUN .EQ. 1) WRITE (7,1011) IUNIT, IDUNIT(I),
C          > IREV(I), IAFX(I), UA(I), (ISP(I,J), J=1,7)
C
C          DO 10 J=1,7
C          IF (ISP(I,J) .NE. 0) GO TO 10
C          NSPU = J - 1
C          WRITE (6,2012) (ISP(I,JJ), JJ=1,NSPU)
C          GO TO 11
C 10 CONTINUE
C
C 11 CONTINUE
C
C INPUT AND PRINT STATE POINT VARIABLES
C ISPT   = STATE POINT NUMBER (IN ORDER OF INPUT)
C KFAZ   = FLAG TO INDICATE PHASE TYPE
C          = 1 IF SOLUTION
C          = 2 IF VAPOR
C          = 3 IF PURE LIQUID

```

```

C      = 4 IF SATURATED LIQUID-VAPOR MIXTURE
C ITFX  = FLAG TO INDICATE TEMPERATURE STATUS
C      = 0 IF TEMPERATURE IS FIXED
C      = 1 IF TEMPERATURE IS UNKNOWN AND .NE. TO ANY OTHER
C      •GE. 2 IF TEMPERATURE IS UNKNOWN. STATE POINTS FOR
C          WHICH ITFX IS EQUAL HAVE IDENTICAL TEMPS.
C IFFX  = FLAG TO INDICATE FLOW STATUS
C      = 0 IF FLOW RATE IS FIXED
C      •GE. 1 IF FLOW RATE IS UNKNOWN. STATE POINTS FOR
C          WHICH IFFX IS EQUAL HAVE IDENTICAL FLOW RATES.
C ICFX  = FLAG TO INDICATE CONCENTRATION STATUS
C      = 0 IF CONCENTRATION IS FIXED
C      •GE. 1 IF CONCENTRATION IS UNKNOWN. STATE POINTS FOR
C          WHICH ICFX IS EQUAL HAVE IDENTICAL CONCS.
C IPFX  = FLAG TO INDICATE PRESSURE STATUS
C      = 0 IF PRESSURE IS FIXED
C      •GE. 1 IF PRESSURE IS UNKNWN. STATE POINTS FOR
C          WHICH IPFX IS EQUAL HAVE IDENTICAL PRESSURES.
C IYFX  = FLAG TO INDICATE VAPOR FRACTION STATUS
C      = 0 IF VAPOR FRACTION IS FIXED
C      •GE. 1 IF VAPOR FRACTION IS UNKNOWN. STATE PCINTS FOR
C          WHICH IYFX IS EQUAL HAVE IDENTICAL VAP. FRACS.
C T      = TEMPERATURE, DEG F
C F      = FLOW RATE, LB/MIN
C C      = CONCENTRATION, PER CENT SOLUTE
C P      = PRESSURE, PSI
C Y      = VAPOR FRACTION
C
C      WRITE (6,2015)
C      DO 16 I=1,NSP
C      READ (5,1016) ISPT,KFAZ(I),ITFX(I),T(I),IFFX(I),F(I),
C      > ICFX(I),C(I),IPFX(I),P(I),IYFX(I),Y(I)
C      WRITE (6,2016) ISPT,KFAZ(I),ITFX(I),T(I),IFFX(I),F(I),
C      > ICFX(I),C(I),IPFX(I),P(I),IYFX(I),Y(I)
C      16 CONTINUE
C
C      SCALE STATE POINT VARIABLES, ZERC ASSOCIATED INDEX VECTORS
C      DO 21 I=1,NSP
C      C(I) = C(I)*CSCALE
C      F(I) = F(I)*FSCALE
C      P(I) = P(I)*PSCALE
C      T(I) = T(I)*TSCALE
C      Y(I) = Y(I)*YSCALE
C      IVARC(I) = 0
C      IVARF(I) = 0
C      IVARP(I) = 0
C      IVART(I) = 0
C      IVARY(I) = 0
C      IVC(I) = 0
C      IVF(I) = 0
C      IVP(I) = 0
C      IVT(I) = 0
C      IVY(I) = 0
C      JC(I) = 0
C      JF(I) = 0
C      JP(I) = 0
C      JT(I) = 0
C      21 JY(I) = 0
C
C      SCALE UNIT VARIABLES AND ZERO ASSOCIATED INDEX VECTCRS
C      DO 22 I=1,NUNITS
C      UA(I) = UA(I)*FSCALE
C      Q(I) = 0•DO
C      IVARA(I) = 0
C      22 IVA(I) = 0
C
C      INITIALIZE COUNTER FOR NUMBER OF VARIABLES TO ZERC
C      IV = 0
C

```

```

C INDEX TEMPERATURE UNKNOWNS
DO 32 I=1,NSP
K= ITFX(I)
IF (K .EQ. 0) GO TO 32
IF (JT(K) .EQ. 0) GO TO 31
IVART(I)= JT(K)
GO TO 32
31 IV= IV + 1
IF (K .NE. 1) JT(K)=IV
IVART(I)= IV
IVT(IV)= I
X(IV)= T(I)
32 CONTINUE
NUT= IV
C
C INDEX CONCENTRATION UNKNOWNS
DO 42 I=1,NSP
K= ICFX(I)
IF (K .EQ. 0) GO TO 42
IF (JC(K) .EQ. 0) GO TO 41
IVARC(I)= JC(K)
GO TO 42
41 IV= IV + 1
JC(K)= IV
IVARC(I)= IV
IVC(IV)= I
X(IV)= C(I)
42 CONTINUE
NUC= IV-NUT
C
C INDEX FLOW UNKNOWNS
DO 52 I=1,NSP
K= IFFX(I)
IF (K .EQ. 0) GO TO 52
IF (JF(K) .EQ. 0) GO TO 51
IVARF(I)= JF(K)
GO TO 52
51 IV= IV + 1
JF(K)= IV
IVARF(I)= IV
IVF(IV)= I
X(IV)= F(I)
52 CONTINUE
NUF= IV-(NUT+NUC)
C
C INDEX PRESSURE UNKNOWNS
DO 62 I=1,NSP
K= IPFX(I)
IF (K .EQ. 0) GO TO 62
IF (JP(K) .EQ. 0) GO TO 61
IVARP(I)= JP(K)
GO TO 62
61 IV= IV + 1
JP(K)= IV
IVARP(I)= IV
IVP(IV)= I
X(IV)= P(I)
62 CONTINUE
NUP= IV-(NUT+NUC+NUF)
C
C INDEX VAPOR FRACTION UNKNOWNS
DO 72 I=1,NSP
K= IYFX(I)
IF (K .EQ. 0) GO TO 72
IF (JY(K) .EQ. 0) GO TO 71
IVARY(I)= JY(K)
GO TO 72
71 IV= IV + 1

```

```

JY(K)= IV
IVARY(I)= IV
IVY(IV)= I
X(IV)= Y(I)
72 CONTINUE
NUY= IV-(NUT+NUC+NUF+NUP)

C INDEX UA UNKNOWN
DO 82 I=1,NUNITS
K= IAFX(I)
IF (K .EQ. 0) GO TO 82
IV= IV + 1
IVARA(I)= IV
IVA(IV)= I
X(IV)= UA(I)
82 CONTINUE
NUA= IV-(NUT+NUC+NUF+NUP+NUY)

C N = IV
C WRITE (6,2084) N,NUT,NUC,NUF,NUP,NUY,NUA
C ZERO A AND CJAC MATRICES
DO 93 J=1,N
DO 91 I=1,NROWA
 91 A(I,J) = 0.D0
DO 92 I=1,NROWJ
 92 CJAC(I,J) = 0.D0
93 CONTINUE

C INITIALIZE LOWER AND UPPER BCUNDS
DO 95 I=1,LENB
  BL(I) = 0.D0
  95 BU(I) = BIGBND

C CALCULATE ENTHALPIES AT INPUT CONDITIONS
CALL ENTHAL

C INIT = 0
NL = 0
NNL = 0

C MAKE INITIAL PASS (INIT = 0) THRCUGH SYSTEM TO GENERATE
C LINEAR INEQUALITY AND BOUND CONSTRAINTS, AND TO DETERMINE
C NUMBER OF LINEAR AND NONLINEAR CCNSTRAINTS
C
DO 110 I=1,NUNITS
ID= IDUNIT(I)
IUNIT= I
GO TO (101,102,103,104,105,106,107,108), ID

C 101 CALL ABSORB (IUNIT,ISP(I,1),ISP(I,2),ISP(I,3),ISP(I,4),
> ISP(I,5),ISP(I,6),FUN,A,CJAC)
GO TO 110

C 102 CALL DESCRB (IUNIT,ISP(I,1),ISP(I,2),ISP(I,3),ISP(I,4),
> ISP(I,5),ISP(I,6),FUN,A,CJAC)
GO TO 110

C 103 CALL RECUP (IUNIT,ISP(I,1),ISP(I,2),ISP(I,3),ISP(I,4),
> IREV(I),FUN,A,CJAC)
GO TO 110

C 104 CALL CGND (IUNIT,ISP(I,1),ISP(I,2),ISP(I,3),ISP(I,4),
> ISP(I,5),FUN,A,CJAC)
GO TO 110

C 105 CALL EVAP (IUNIT,ISP(I,1),ISP(I,2),ISP(I,3),ISP(I,4),

```

```

> ISP(I,5),FUN,A,CJAC)
GO TO 110
C 106 CALL VALVE (IUNIT,ISP(I,1),ISP(I,2),FUN,CJAC)
GO TO 110
C 107 CALL MIX (IUNIT,ISP(I,1),ISP(I,2),ISP(I,3),FUN,CJAC)
GO TO 110
C 108 CALL SPLIT (IUNIT,ISP(I,1),ISP(I,2),ISP(I,3),FUN,CJAC)
C 110 CONTINUE
C
INIT = 1
NCLIN = NL
NCNLN = NNL
NCTOTL = N + NCLIN + NCNLN
C SET UPPER BOUNDS FOR NONLINEAR EQUALITY CONSTRAINTS TO ZERO
I1 = N + NCLIN + 1
DO 121 I=I1,NCTOTL
121 BU(I) = 0.D0
C SET MAXIMUM ALLOWABLE CONSTRAINT VIOLATIONS
DO 122 I=1,NCTOTL
122 FEATOL(I) = CTOL
C
C NAG ROUTINE E04ZCF MAY BE USED TO CHECK THE CODING OF
C FIRST DERIVATIVES OF CONSTRAINT AND OBJECTIVE FUNCTIONS.
C USE OF E04ZCF REQUIRES LINKING THE NAG LIBRARY.
IFAIL = 0
CALL E04ZCF (N,NCNLN,NROWJ,CCNFUN,OBJFUN,FUN,
> CJAC,CBJF,OBJGRD,X,WORK,LENW,IFAIL)
C WRITE (6,2125) IFAIL
C
C INVOKE OPTIMIZER PACKAGE TO SOLVE SYSTEM EQUATIONS
CALL NPSCL (ITMAX,MSGVL,N,NCLIN,NCNLN,NCTOTL,NROWA,
> NROWJ,NROWR,BIGBND,EPSAF,ETA,FTOL,A,BL,BU,FEATCL,
> CCNFUN,OBJFUN,CCLD,ORTHOG,INFORM,ITER,ISTATE,FUN,CJAC,
> CLAMDA,OBJF,OBJGRD,R,X,IWORK,LENIW,WORK,LENW)
C
C UNSCALE AND PRINT STATE POINT VARIABLES
WRITE (6,2130)
DO 131 I=1,NSP
C1 = C(I)/CSCALE
F1 = F(I)/FSCALE
P1 = P(I)/PSCALE
T1 = T(I)/TSCALE
Y1 = Y(I)/YSCALE
H1 = H(I)/TSCALE
WRITE (6,2131) I,T1,H1,F1,C1,P1,Y1
IF (IPUN .EQ. 1) WRITE (7,1016) I,KFAZ(I),ITFX(I),T1,
> IFFX(I),F1,ICFX(I),C1,IPFX(I),P1,IYFX(I),Y1
131 CONTINUE
C
C UNSCALE AND PRINT UNIT VARIABLES
WRITE (6,2140)
DO 141 I=1,NUNITS
ID = IDUNIT(I)
Q1 = Q(I)/(FSCALE*TSCALE)
UA1 = UA(I)/FSCALE
141 WRITE (6,2141) I,ANAME(ID),Q1,UA1
C
1000 FORMAT(16I5)
1011 FORMAT(2I5,3X,2I1,F10.0,7I5)
1016 FORMAT(2I5,5(I4,F6.1))
1100 FORMAT(8F10.0)
2000 FORMAT(

```

```

> * IOBJ    = OBJECTIVE FUNCTION FLAG      =',I4,/
> * ITMAX   = ALLOWABLE NUMBER OF ITERATIONS =',I4,/
> * MSGLVL  = OPTIMIZER MESSAGE LEVEL FLAG  =',I4,/
2001 FORMAT('1SINGLE STAGE HEAT TRANSFORMER')
2002 FORMAT('1SINGLE STAGE CHILLER')
2003 FORMAT('1DOUBLE STAGE HEAT TRANSFORMER')
2004 FORMAT('1DOUBLE STAGE CHILLER')
2006 FORMAT(' LI-BR/WATER',/)
2007 FORMAT(' INVALID WORKING FLUID CODE SPECIFIED',/)
2008 FORMAT(' NO. OF UNITS      ',I5,/
>     ' NO. OF STATE POINTS',IS,/)
2010 FORMAT(' UNIT SPECIFICATIONS',/,/
>     ' NO. ID IREV IAFX      UA          STATE POINTS',/,/
>           BTU/(M-F)',/)
2011 FORMAT(1X,I2,2I4,I5,1PE12.3)
2012 FORMAT(1H+,29X,7I4)
2015 FORMAT(//,' STATE POINT SPECIFICATIONS',/,/
>     ' NO. KFAZ ITFX TEMP. IFFX FLOW ICFX CONC.',/
>     ' IPFX PRESS IYFX VAPOR',/,16X,
>     ' F          LB/M        %          PSI          FRAC.',/)
2016 FORMAT(1X,I2,I4,I6,F7.1,I4,F7.1,I4,F7.2,I4,F7.2,I4,F7.3)
2084 FORMAT(//,' NO. OF VARIABLES      ',I5,/
>     ' NO. OF UNKNOWN TEMPERATURES      ',I5,/
>     ' NO. OF UNKNOWN CONCENTRATIONS    ',I5,/
>     ' NO. OF UNKNOWN FLOWS            ',I5,/
>     ' NO. OF UNKNOWN PRESSURES       ',I5,/
>     ' NO. OF UNKNOWN VAFOR FRACTIONS ',I5,/
>     ' NO. OF UNKNOWN HEAT EX. AREAS  ',I5)
2101 FORMAT(
>     ' CTOL    = CONSTRAINT FUNCTION TOLERANCE =',1PE9.1,/,/
>     ' FTOL    = OBJECTIVE FUNCTION TOLERANCE =',E9.1,/,/
>     ' EPSAF   = ERROR IN COMPUTING OBJ. FUNCT. =',E9.1,/,/
>     ' UROUND = UNIT ROUNDOFF CF COMPUTER     =',E9.1,/)
2102 FORMAT(' CSCALE = CONCENTRATION SCALE FACTOR =',F6.3,/,/
>     ' FSCALE = FLOW          SCALE FACTOR =',F6.3,/,/
>     ' PSCALE = PRESSURE      SCALE FACTOR =',F6.2,/,/
>     ' TSCALE = TEMPERATURE    SCALE FACTOR =',F6.3,/,/
>     ' YSCALE = VAPOR FRACTI CN SCALE FACTOR =',F6.1,/)
2125 FORMAT(' SUBROUTINE E04ZCF,  IFAIL =',I4)
2130 FORMAT(//,' STATE TEMP. ENTHALPY FLOW RATE',/
>     ' CONCENTR. PRESSURE VAPCR FRAC.',/,/
>     ' POINT    DEG F      BTU/LB      LB/M      % SOLUTE',/
>           PSIA',/)
2131 FORMAT(1X,I3,2X,1P8E11.4)
2140 FORMAT(//,' UNIT UNIT HEAT TRANSFER      UA',/,/
>           ' NO. TYPE      BTU/M      BTU/(M-F)',/)
2141 FORMAT(1X,I3,5X,A4,1P2E14.4)

C
STOP
END

```

```
FUNCTION D1MACH (IDUM)
IMPLICIT REAL*8 (A-H,O-Z)
U = 1.D0
10 U = 0.5D0*U
UPLUS1 = U + 1.D0
IF (UPLUS1 .NE. 1.D0) GO TO 10
D1MACH = 2.D0*U
RETURN
END
```

```

SUBROUTINE CONFUN ( MODE, NCNLN, N, NRJ, X, FUN, CJAC, NSTATE)
C CALCULATE CONSTRAINT FUNCTIONS AND THEIR FIRST DERIVATIVES
  IMPLICIT REAL*8 ( A-H,O-Z)
  COMMON/AA/ C(50),F(50),P(50),T(50),Y(50),IVARC(50),
  > IVARF(50),IVARP(50),IVART(50),IVARY(50),KFAZ(50),MSUB
  COMMON/BB/ ICFX(50),IFFX(50),IPFX(50),ITFX(50),IYFX(50),
  > IVC(50),IVF(50),IVP(50),IVT(50),IVY(50),IVA(50)
  COMMON/DD/ GRAD(50),OBJECT,ICBJ
  COMMON/EE/ Q(20),UA(20),IREV(20),IAFX(20),IVARA(20)
  COMMON/FF/ IDUNIT(20),ISP(20,7),NSP,NUNITS,NUC,NUF,
  > NUP,NUT,NUY,NUA
  COMMON/NN/ NROWA,NROWJ,NL,NNL,INIT
  DIMENSION CJAC(NROWJ,1),FUN(1),X(1)

C UPDATE TEMPERATURE VECTOR
  IF (NUT .EQ. 0) GO TO 40
  DO 31 IV=1,NUT
  31 T(IVT(IV)) = X(IV)
  DO 33 I=1,NSP
  IF (ITFX(I) .LT. 2) GO TO 33
  DO 32 J=1,NSP
  32 IF (ITFX(J) .EQ. ITFX(I) ) T(J)= T(I)
  33 CONTINUE

C UPDATE CONCENTRATION VECTOR
  40 IF (NUC .EQ. 0) GO TO 50
  IV1= NUT + 1
  IV2= IV1 + NUC - 1
  DO 41 IV= IV1,IV2
  41 C(IVC(IV))= X(IV)
  DO 43 I=1,NSP
  IF (ICFX(I) .EQ. 0) GO TO 43
  DO 42 J= 1,NSP
  42 IF (ICFX(J) .EQ. ICFX(I) ) C(J)= C(I)
  43 CONTINUE

C UPDATE FLOW VECTOR
  50 IF (NUF .EQ. 0) GO TO 60
  IV1= IV2 + 1
  IV2= IV1 + NUF - 1
  DO 51 IV= IV1,IV2
  51 F(IVF(IV))= X(IV)
  DO 53 I=1,NSP
  IF (IFFX(I) .EQ. 0) GO TO 53
  DO 52 J= 1,NSP
  52 IF (IFFX(J) .EQ. IFFX(I) ) F(J)= F(I)
  53 CONTINUE

C UPDATE PRESSURE VECTOR
  60 IF (NUP .EQ. 0) GO TO 70
  IV1= IV2 + 1
  IV2= IV1 + NUP - 1
  DO 61 IV= IV1,IV2
  61 P(IVP(IV))= X(IV)
  DO 63 I=1,NSP
  IF (IPFX(I) .EQ. 0) GO TO 63
  DO 62 J= 1,NSP
  62 IF (IPFX(J) .EQ. IPFX(I) ) P(J)= P(I)
  63 CONTINUE

C UPDATE VAPOR FRACTION VECTOR
  70 IF (NUY .EQ. 0) GO TO 80
  IV1= IV2 + 1
  IV2= IV1 + NUY - 1
  DO 71 IV= IV1,IV2
  71 Y(IVY(IV))= X(IV)
  DO 73 I=1,NSP
  IF (IYFX(I) .EQ. 0) GO TO 73

```

```

DO 72 J=1,NSP
72 IF (IYFX(J) .EQ. IYFX(I) ) Y(J)= Y(I)
73 CONTINUE
C
C UPDATE UA VECTOR
80 IF (NUA .EQ. 0) GO TO 90
  IV1= IV2 + 1
  IV2= IV1 + NUA - 1
  DO 81 IV= IV1,IV2
81 UA(IVA(IV))= X(IV)
C
C CALCULATE ENTHALPIES
90 CALL ENTHAL
C
C ZERO OUT THE CONSTRAINT JACOBIAN MATRIX
  DO 92 J=1,N
    DO 92 I=1,NCNLN
92 CJAC(I,J) = 0.D0
C
  NL = 0
  NNL = 0
C
  DO 110 I=1,NUNITS
    ID = IDUNIT(I)
    IUNIT = I
C
  GO TO (101,102,103,104,105,106,107,108), ID
C
101 CALL ABSORB (IUNIT,ISP(I,1),ISP(I,2),ISP(I,3),ISP(I,4),
> ISP(I,5),ISP(I,6),FUN,A,CJAC)
  GO TO 110
C
102 CALL DESORB (IUNIT,ISP(I,1),ISP(I,2),ISP(I,3),ISP(I,4),
> ISP(I,5),ISP(I,6),FUN,A,CJAC)
  GO TO 110
C
103 CALL RECUP (IUNIT,ISP(I,1),ISP(I,2),ISP(I,3),ISP(I,4),
> IREV(I),FUN,A,CJAC)
  GO TO 110
C
104 CALL COND (IUNIT,ISP(I,1),ISP(I,2),ISP(I,3),ISP(I,4),
> ISP(I,5),FUN,A,CJAC)
  GO TO 110
C
105 CALL EVAP (IUNIT,ISP(I,1),ISP(I,2),ISP(I,3),ISP(I,4),
> ISP(I,5),FUN,A,CJAC)
  GO TO 110
C
106 CALL VALVE (IUNIT,ISP(I,1),ISP(I,2),FUN,CJAC)
  GO TO 110
C
107 CALL MIX (IUNIT,ISP(I,1),ISP(I,2),ISP(I,3),FUN,CJAC)
  GO TO 110
C
108 CALL SPLIT (IUNIT,ISP(I,1),ISP(I,2),ISP(I,3),FUN,CJAC)
C
110 CONTINUE
C
  GO TO (140,120,140,140), IOBJ
C
C CALCULATION OF OBJECTIVE FUNCTION AND ITS GRADIENT WHEN THE
C OBJECTIVE IS THE SUM OF SQUARES OF THE CONSTRAINT FUNCTIONS
120 OBJECT = 0.D0
  DO 130 I=1,NCNLN
130 OBJECT = OBJECT + FUN(I)*FUN(I)
C
  DO 132 I=1,N
    GRADI = 0.D0

```

```
      DO 131 J=1,NCNLN
131 GRADI = GRADI + FUN(J)*CJAC(J,I)
132 GRAD(I) = 2.D0*GRADI
C
140 RETURN
END
```

```

SUBROUTINE OBJFUN ( MODE, N, X, OBJF, OBJGRD, NSTATE)
C CALCULATION OF OBJECTIVE FUNCTION AND ITS FIRST DERIVATIVES
IMPLICIT REAL*8 (A-H,O-Z)
DIMENSION X(1),OBJGRD(1)
COMMON/AA/ C(50),F(50),P(50),T(50),Y(50),IVARC(50),
> IVARF(50),IVARP(50),IVART(50),IVARY(50),KFAZ(50),MSUB
COMMON/EB/ ICFX(50),IFFX(50),IPFX(50),ITFX(50),IYFX(50),
> IVC(50),IVF(50),IVP(50),IVT(50),IVY(50),IVA(50)
COMMON/DD/ GRAD(50),OBJECT,ICBJ
COMMON/EE/ Q(20),UA(20),IREV(20),IAFX(20),IVARA(20)
COMMON/FF/ IDUNIT(20),ISP(20,7),NSP,NUNITS,NUC,NUF,
> NUP,NUT,NUY,NUA
C
C      GO TO (10,20,30,40),IOBJ
C
C OBJECTIVE FUNCTION IS ONE
10 OBJF = 1.0D0
DO 11 I=1,N
11 OBJGRD(I) = 0.0D0
RETURN
C
C OBJECTIVE FUNCTION IS SUM OF SQUARES OF CONSTRAINT FUNCTIONS
20 OBJF = OBJECT
DO 21 I=1,N
21 OBJGRD(I) = GRAD(I)
RETURN
C
C OBJECTIVE FUNCTION IS SUM OF UNKNOWN UA'S
30 OBJF = 0.0D0
DO 31 I=1,NUNITS
IF (IAFX(I) .EQ. 0) GO TO 31
OBJF = OBJF + UA(I)
31 CONTINUE
DO 32 I=1,N
OBJGRD(I) = 0.0D0
32 IF (IVA(I) .NE. 0) OBJGRD(I) = 1.0D0
RETURN
C
C OBJECTIVE FUNCTION IS PROPORTIONAL TO PAYBACK TIME
40 OBJF = 0.0D0
DO 41 I=1,NUNITS
41 OBJF = OBJF + UA(I)
DENOM = F(3)*(T(9)-T(3))
OBJF = 0.1D0*OBJF/DENOM
DO 42 I=1,N
OBJGRD(I) = 0.0D0
42 IF (IVA(I) .NE. 0) OBJGRD(I) = 0.1D0/DENOM
IF (IFFX(3) .EQ. 0) GO TO 43
I = IVARF(3)
OBJGRD(I) = -OBJF/F(3)
43 IF (ITFX(3) .EQ. 0) GO TO 44
I = IVART(3)
OBJGRD(I) = OBJF/(T(9)-T(3))
44 IF (ITFX(9) .EQ. 0) GO TO 45
I = IVART(9)
OBJGRD(I) = -OBJF/(T(9)-T(3))
45 CONTINUE
RETURN
C
END

```

```
SUBROUTINE BNDFUN (IV1,IV2,V1,V2,A)
C LINEAR INEQUALITY AND BOUND CONSTRAINTS
IMPLICIT REAL*8 (A-H,O-Z)
COMMON/GG/ BL(125),BU(125)
COMMON/NN/ NROWA,NROWJ,NL,NNL,INIT
DIMENSION A(NROWA,1)
IF (IV1 .GT. 0 .AND. IV2 .GT. 0) GO TO 10
IF (IV1 .GT. 0 .AND. IV2 .EQ. 0) GO TO 20
IF (IV1 .EQ. 0 .AND. IV2 .GT. 0) GO TO 30
RETURN
C LINEAR INEQUALITY CONSTRAINT, V1(UNKNOWN) - V2(UNKNCWN) > 0
10 NL=NL+1
CALL MATRIX (NL,IV1, 1.D0,A,NROWA)
CALL MATRIX (NL,IV2,-1.D0,A,NROWA)
RETURN
C LOWER BOUND CONSTRAINT, V1(UNKNCWN) > V2(FIXED)
20 BL(IV1)=V2
RETURN
C UPPER BOUND CONSTRAINT, V2(UNKNOWN) < V1(FIXED)
30 BU(IV2)=V1
RETURN
END
```

```
SUBROUTINE ENTHAL
C CALCULATE ENTHALPIES OF STREAMS AT STATE POINTS
  IMPLICIT REAL*8 (A-H,O-Z)
  COMMON/AA/ C(50),F(50),P(50),T(50),Y(50),IVARC(50),
  > IVARF(50),IVARP(50),IVART(50),IVARY(50),KFAZ(50),MSUB
  COMMON/CC/ H(50),DHC(50),DHP(50),DHT(50),DHY(50)
  COMMON/FF/ IDUNIT(20),ISP(20,7),NSP,NUNITS,NUC,NUF,
  > NUP,NUT,NUY,NUA
C
C SET ICALC TO SIGNAL THAT ENTHALPY CALCS. ARE TO BE DONE
  ICALC = 2
C
C PERFORM ENTHALPY CALCULATIONS FOR EACH STATE POINT
  DO 50 I=1,NSP
C
    GO TO (10,20),MSUB
C
C LIBR-WATER
  10 CALL EQB1 (P(I),C(I),T(I),Y(I),KFAZ(I),ICALC,TE,
  > DFP,DFT,DFC,H(I),DHP(I),DHT(I),DHC(I),DHY(I))
  GO TO 50
C
C SYSTEM NOT PRESENTLY CODED
  20 CALL EQB2 (P(I),C(I),T(I),Y(I),KFAZ(I),ICALC,TE,
  > DFP,DFT,DFC,H(I),DHP(I),DHT(I),DHC(I),DHY(I))
C
  50 CONTINUE
  RETURN
  END
```

```

SUBROUTINE ABSORB (IUNIT,I1,I2,I3,I4,I5,I6,FUN,A,CJAC)
C ABSORBER CONSTRAINTS
  IMPLICIT REAL*8 (A-H,O-Z)
  COMMON/AA/ C(50),F(50),P(50),T(50),Y(50),IVARC(50),
  > IVARF(50),IVARP(50),IVART(50),IVARY(50),KFAZ(50),MSUB
  COMMON/CC/ H(50),DHC(50),DHP(50),DHT(50),DHY(50)
  COMMON/NN/ NROWA,NROWJ,NL,NNL,INIT
  DIMENSION A(NROWA,1),CJAC(NRCWJ,1),FUN(1),IVEC(7)
C
C TOTAL MASS BALANCE, F1 + F2 - F5 = 0
  IVEC(1)= I1
  IVEC(2)= I2
  IVEC(3)=-I5
  NS=3
  CALL MBAL (NS,IVEC,FUN,CJAC)
C
C SOLUTE MASS BALANCE, F1*C1 + F2*C2 - F5*C5 = 0
  CALL CBAL (NS,IVEC,FUN,CJAC)
C
C ENTHALPY BALANCE, F1*H1 + F2*H2 + F3*H3 - F4*H4 - F5*H5 = 0
  IVEC(4)= I3
  IVEC(5)=-I4
  NS=5
  CALL HBAL (NS,IVEC,FUN,CJAC)
C
C SOLUTE MASS BALANCE AROUND TOP OF ABSORBER,
C F1*C1 + (F6 - F1)*C2 - F6*C6 = 0
  NNL = NNL + 1
  FUN(NNL) = F(I1)*C(I1) + (F(I6)-F(I1))*C(I2) -
  > F(I6)*C(I6)
  CALL MATRIX (NNL,IVARF(I1),C(I1)-C(I2),CJAC,NROWJ)
  CALL MATRIX (NNL,IVARF(I6),C(I2)-C(I6),CJAC,NROWJ)
  CALL MATRIX (NNL,IVARC(I1),F(I1),CJAC,NROWJ)
  CALL MATRIX (NNL,IVARC(I2),F(I6)-F(I1),CJAC,NROWJ)
  CALL MATRIX (NNL,IVARC(I6),-F(I6),CJAC,NROWJ)
C
C SOLUTE MASS BALANCE AROUND TOP OF ABSORBER
  NNL = NNL + 1
  FUN(NNL) = F(I1)*H(I1) + (F(I6)-F(I1))*H(I2) -
  > F(I6)*H(I6)
  CALL MATRIX (NNL,IVARF(I1),H(I1)-H(I2),CJAC,NROWJ)
  CALL MATRIX (NNL,IVARF(I6),H(I2)-H(I6),CJAC,NROWJ)
  CALL MATRIX (NNL,IVARC(I1),F(I1)*DHC(I1),CJAC,NROWJ)
  CALL MATRIX (NNL,IVARC(I2),(F(I6)-F(I1))*DHC(I2),
  > CJAC,NRCWJ)
  CALL MATRIX (NNL,IVARC(I6),-F(I6)*DHC(I6),CJAC,NROWJ)
  CALL MATRIX (NNL,IVARP(I1),F(I1)*DHP(I1),CJAC,NROWJ)
  CALL MATRIX (NNL,IVARP(I2),(F(I6)-F(I1))*DHP(I2),
  > CJAC,NRCWJ)
  CALL MATRIX (NNL,IVARP(I6),-F(I6)*DHP(I6),CJAC,NROWJ)
  CALL MATRIX (NNL,IVART(I1),F(I1)*DHT(I1),CJAC,NROWJ)
  CALL MATRIX (NNL,IVART(I2),(F(I6)-F(I1))*DHT(I2),
  > CJAC,NRCWJ)
  CALL MATRIX (NNL,IVART(I6),-F(I6)*DHT(I6),CJAC,NROWJ)
C
C HEAT BALANCE
  CALL QBAL (IUNIT,I6,I5,I3,I4,FUN,CJAC)
C
C EQUIL. VAPOR PRESS. RELATION FOR SOLUTION ENTERING ABSORBER
  CALL VBAL (I6,FUN,CJAC)
C
C EQUIL. VAPOR PRESS. RELATION FOR SOLUTION EXITING ABSORBER
  CALL VBAL (I5,FUN,CJAC)
C
C IF (INIT .EQ. 1) RETURN
C
C LINEAR INEQUALITY AND BOUND CONSTRAINTS ON TEMPERATURES
C T6 > T4

```

```
C   CALL BNDFUN (IVART(16),IVART(14),T(16),T(14),A)
C T5 > T3
C   CALL BNDFUN (IVART(15),IVART(13),T(15),T(13),A)
C T4 > T3
C   CALL BNDFUN (IVART(14),IVART(13),T(14),T(13),A)
C
C   RETURN
C   END
```

```

SUBROUTINE DESORB (IUNIT,I1,I2,I3,I4,I5,I6,FUN,A,CJAC)
C DESORBER CONSTRAINTS
  IMPLICIT REAL*8 (A-H,O-Z)
  COMMON/AA/ C(50),F(50),P(50),T(50),Y(50),IVARC(50),
  > IVARF(50),IVARP(50),IVART(50),IVARY(50),KFAZ(50),MSUB
  COMMON/CC/ H(50),DHC(50),DHP(50),DHT(50),DHY(50)
  COMMON/NN/ NROWA,NROWJ,NL,NNL,INIT
  DIMENSION A(NROWA,1),CJAC(NRCWJ,1),FUN(1),IVEC(7)

C TOTAL MASS BALANCE, F1 - F2 - F5 = 0
  IVEC(1)= I1
  IVEC(2)=-I2
  IVEC(3)=-I5
  NS=3
  CALL MBAL (NS,IVEC,FUN,CJAC)

C SOLUTE MASS BALANCE, F1*C1 - F2*C2 - F5*C5 = 0
  CALL CBAL (NS,IVEC,FUN,CJAC)

C ENTHALPY BALANCE, F1*H1 - F2*H2 + F3*H3 - F4*H4 - F5*H5 = 0
  IVEC(4)= I3
  IVEC(5)=-I4
  NS=5
  CALL HBAL (NS,IVEC,FUN,CJAC)

C SOLUTE MASS BALANCE AROUND TOP OF DESORBER,
C F1*C1 + (F6 - F1)*C2 - F6*C6 = 0
  NNL = NNL + 1
  FUN(NNL) = F(I1)*C(I1) + (F(I6)-F(I1))*C(I2) -
  > F(I6)*C(I6)
  CALL MATRIX (NNL,IVARF(I1),C(I1)-C(I2),CJAC,NROWJ)
  CALL MATRIX (NNL,IVARF(I6),C(I2)-C(I6),CJAC,NROWJ)
  CALL MATRIX (NNL,IVARC(I1),F(I1),CJAC,NROWJ)
  CALL MATRIX (NNL,IVARC(I2),F(I6)-F(I1),CJAC,NROWJ)
  CALL MATRIX (NNL,IVARC(I6),-F(I6),CJAC,NROWJ)

C PARTIAL ENTHALPY BALANCE
  NNL = NNL + 1
  FUN(NNL) = F(I1)*H(I1) + (F(I6)-F(I1))*H(I2) -
  > F(I6)*H(I6)
  CALL MATRIX (NNL,IVARF(I1),H(I1)-H(I2),CJAC,NROWJ)
  CALL MATRIX (NNL,IVARF(I6),H(I2)-H(I6),CJAC,NROWJ)
  CALL MATRIX (NNL,IVARC(I1),F(I1)*DHC(I1),CJAC,NROWJ)
  CALL MATRIX (NNL,IVARC(I2),(F(I6)-F(I1))*DHC(I2),
  > CJAC,NRCWJ)
  CALL MATRIX (NNL,IVARC(I6),-F(I6)*DHC(I6),CJAC,NROWJ)
  CALL MATRIX (NNL,IVARP(I1),F(I1)*DHP(I1),CJAC,NROWJ)
  CALL MATRIX (NNL,IVARP(I2),(F(I6)-F(I1))*DHP(I2),
  > CJAC,NRCWJ)
  CALL MATRIX (NNL,IVARP(I6),-F(I6)*DHP(I6),CJAC,NROWJ)
  CALL MATRIX (NNL,IVART(I1),F(I1)*DHT(I1),CJAC,NROWJ)
  CALL MATRIX (NNL,IVART(I2),(F(I6)-F(I1))*DHT(I2),
  > CJAC,NRCWJ)
  CALL MATRIX (NNL,IVART(I6),-F(I6)*DHT(I6),CJAC,NROWJ)

C HEAT BALANCE
  CALL QBAL (IUNIT,I6,I5,I3,I4,FUN,CJAC)

C EQUIL. VAPOR PRESS. RELATION FOR SOLUTION ENTERING DESORBER
  CALL VBAL (I6,FUN,CJAC)

C EQUIL. VAPOR PRESS. RELATION FOR SOLUTION EXITING DESORBER
  CALL VBAL (I5,FUN,CJAC)

C IF (INIT .EQ. 1) RETURN

C LINEAR INEQUALITY AND BOUND CONSTRAINTS ON TEMPERATURES
  C T4 > T6

```

```
C    CALL BNDFUN (IVART(I4),IVART(I6),T(I4),T(I6),A)
C    T3 > T5
C    CALL BNDFUN (IVART(I3),IVART(I5),T(I3),T(I5),A)
C    T3 > T4
C    CALL BNDFUN (IVART(I3),IVART(I4),T(I3),T(I4),A)
C
C    RETURN
C    END
```

```

SUBROUTINE RECUP (IUNIT,I1,I2,I3,I4,IREV,FUN,A,CJAC)
C RECUPERATOR CONSTRAINTS
  IMPLICIT REAL*8 (A-H,O-Z)
  COMMON/AA/ C(50),F(50),P(50),T(50),Y(50),IVARC(50),
  > IVARF(50),IVARP(50),IVART(50),IVARY(50),KFAZ(50),MSUB
  COMMON/NN/ NROWA,NROWJ,NL,NNL,INIT
  DIMENSION A(NROWA,1),CJAC(NRCWJ,1),FUN(1),IVEC(7)
C
C ENTHALPY BALANCE, F1*H1 - F2*H2 + F3*H3 - F4*H4 = 0
  IVEC(1) = I1
  IVEC(2) = -I2
  IVEC(3) = I3
  IVEC(4) = -I4
  NS = 4
  CALL HBAL (NS,IVEC,FUN,CJAC)
C
C HEAT BALANCE
  CALL QBAL (IUNIT,I1,I2,I3,I4,FUN,CJAC)
C
  IF (INIT .EQ. 1) RETURN
C
C CHECK DIRECTION OF HEAT FLOW IN RECUPERATOR
  IF (IREV .EQ. 1) GO TO 10
C
C LINEAR INEQUALITY AND BOUND CONSTRAINTS ON TEMPERATURES
C WHEN HEAT FLOW IS IN FORWARD DIRECTION
  C T3 > T2
    CALL BNDFUN (IVART(I3),IVART(I2),T(I3),T(I2),A)
  C T4 > T1
    CALL BNDFUN (IVART(I4),IVART(I1),T(I4),T(I1),A)
C
C LINEAR INEQUALITY AND BOUND CONSTRAINTS ON TEMPERATURES
C WHEN HEAT FLOW IS IN REVERSE DIRECTION
  C T2 > T3
    10 CALL BNDFUN (IVART(I2),IVART(I3),T(I2),T(I3),A)
  C T1 > T4
    CALL BNDFUN (IVART(I1),IVART(I4),T(I1),T(I4),A)
C
  RETURN
END

```

```

SUBROUTINE COND (IUNIT,I1,I2,I3,I4,I5,FUN,A,CJAC)
C CONDENSER CONSTRAINTS
  IMPLICIT REAL*8 (A-H,O-Z)
  COMMON/AA/ C(50),F(50),P(50),T(50),Y(50),IVARC(50),
  > IVARF(50),IVARP(50),IVART(50),IVARY(50),KFAZ(50),MSUB
  COMMON/NN/ NROWA,NROWJ,NL,NNL,INIT
  DIMENSION A(NROWA,1),CJAC(NRCWJ,1),FUN(1),IVEC(7)
C
C ENTHALPY BALANCE, F1*H1 - F2*H2 + F3*H3 - F4*H4 = 0
  IVEC(1) = I1
  IVEC(2) = -I2
  IVEC(3) = I3
  IVEC(4) = -I4
  NS=4
  CALL HBAL (NS,IVEC,FUN,CJAC)
C
C HEAT BALANCE
  CALL QBAL (IUNIT,I5,I2,I3,I4,FUN,CJAC)
C
C EQUIL. VAPOR PRESS. RELATION FOR VAPOR ENTERING CCNDENSER
  CALL VBAL (I5,FUN,CJAC)
C
C IF ( INIT .EQ. 1) RETURN
C
C LINEAR INEQUALITY AND BOUND CONSTRAINTS ON TEMPERATURES
C T5 > T4
  CALL BNDFUN (IVART(I5),IVART(I4),T(I5),T(I4),A)
C T2 > T3
  CALL BNDFUN (IVART(I2),IVART(I3),T(I2),T(I3),A)
C T4 > T3
  CALL BNDFUN (IVART(I4),IVART(I3),T(I4),T(I3),A)
C
C RETURN
END

```

```

SUBROUTINE EVAP (IUNIT,I1,I2,I3,I4,I5,FUN,A,CJAC)
C EVAPORATOR CONSTRAINTS
  IMPLICIT REAL*8 (A-H,O-Z)
  COMMON/AA/ C(50),F(50),P(50),T(50),Y(50),IVARC(50),
> IVARF(50),IVARP(50),IVART(50),IVARY(50),KFAZ(50),MSUB
  COMMON/NN/ NROWA,NROWJ,NL,NNL,INIT
  DIMENSION A(NROWA,1),CJAC(NRCWJ,1),FUN(1),IVEC(7)
C
C ENTHALPY BALANCE, F1*H1 - F2*H2 + F3*H3 - F4*H4 = 0
  IVEC(1) = I1
  IVEC(2) = -I2
  IVEC(3) = I3
  IVEC(4) = -I4
  NS=4
  CALL HBAL (NS,IVEC,FUN,CJAC)
C
C HEAT BALANCE
  CALL QBAL (IUNIT,I5,I2,I3,I4,FUN,CJAC)
C
C EQLIL. VAFOR PRESS. RELATION FOR LIQUID ENTERING EVAPORATOR
  CALL VBAL (I5,FUN,CJAC)
C
  IF (INIT .EQ. 1) RETURN
C
C LINEAR INEQUALITY AND BOUND CONSTRAINTS ON TEMPERATURES
C  T4 > T5
  CALL BNDFUN (IVART(I4),IVART(I5),T(I4),T(I5),A)
C  T3 > T2
  CALL BNDFUN (IVART(I3),IVART(I2),T(I3),T(I2),A)
C  T3 > T4
  CALL BNDFUN (IVART(I3),IVART(I4),T(I3),T(I4),A)
C
  RETURN
END

```

```
SUBROUTINE VALVE (IUNIT,I1,I2,FUN,CJAC)
C VALVE CONSTRAINTS
  IMPLICIT REAL*8 (A-H,O-Z)
  COMMON/AA/ C(50),F(50),P(50),T(50),Y(50),IVARC(50),
  > IVARF(50),IVARP(50),IVART(50),IVARY(50),KFAZ(50),MSUB
  COMMON/NN/ NROWA,NROWJ,NL,NNL,INIT
  DIMENSION CJAC(NROWJ,1),FUN(1),IVEC(7)
C
C ENTHALPY BALANCE, F1*H1 - F2*H2 = 0
  IVEC(1)= I1
  IVEC(2)=-I2
  NS=2
  CALL HBAL (NS,IVEC,FUN,CJAC)
C
  RETURN
END
```

```

SUBROUTINE MIX (IUNIT,I1,I2,I3,FUN,CJAC)
C MIXER CONSTRAINTS
  IMPLICIT REAL*8 (A-H,O-Z)
  COMMON/AA/ C(50),F(50),P(50),T(50),Y(50),IVARC(50),
  > IVARF(50),IVARP(50),IVART(50),IVARY(50),KFAZ(50),MSUB
  COMMON/BB/ ICFX(50),IFFX(50),IPFX(50),ITFX(50),IYFX(50),
  > IVC(50),IVF(50),IVP(50),IVT(50),IVY(50),IVA(50)
  COMMON/NN/ NROWA,NROWJ,NL,NNL,INIT
  DIMENSION CJAC(NROWJ,1),FUN(1),IVEC(7)

C TOTAL MASS BALANCE, F1 + F2 - F3 = 0
  IVEC(1)= I1
  IVEC(2)= I2
  IVEC(3)=-I3
  NS=3
  IF (IFFX(I1) .EQ. 0 .AND. IFFX(I2) .EQ. 0) GO TO 20
  CALL MBAL (NS,IVEC,FUN,CJAC)
C COMPONENT MASS BALANCE, F1*C1 + F2*C2 - F3*C3 = 0
  20 IF (C(I3) .EQ. 0.D0) GO TO 30
  CALL CBAL (NS,IVEC,FUN,CJAC)
C ENTHALPY BALANCE, F1*H1 + F2*H2 -F3*H3 = 0
  30 CALL HBAL (NS,IVEC,FUN,CJAC)
C
  RETURN
END

```

```
SUBROUTINE SPLIT (IUNIT,I1,I2,I3,FUN,CJAC)
C SPLITTER CONSTRAINTS
  IMPLICIT REAL*8 (A-H,O-Z)
  COMMON/AA/ C(50),F(50),P(50),T(50),Y(50),IVARC(50),
  > IVARF(50),IVARP(50),IVART(50),IVARY(50),KFAZ(50),MSUB
  COMMON/BB/ ICFX(50),IFFX(50),IPFX(50),ITFX(50),IYFX(50),
  > IVC(50),IVF(50),IVP(50),IVT(50),IVY(50),IVA(50)
  COMMON/NN/ NROWA,NROWJ,NL,NNL,INIT
  DIMENSION CJAC(NROWJ,1),FUN(1),IVEC(7)
C
  IF (IFFX(I1) .EQ. 0 .AND. IFFX(2) .EQ. 0) RETURN
C
C TOTAL MASS BALANCE, -F1 - F2 + F3 = 0
  IVEC(1)=-I1
  IVEC(2)=-I2
  IVEC(3)= I3
  NS=3
  CALL MEAL (NS,IVEC,FUN,CJAC)
C
  RETURN
  END
```

```
SUBROUTINE CBAL (NS, IVEC, FUN, CJAC)
C COMPONENT MASS BALANCE
IMPLICIT REAL*8 (A-H,O-Z)
COMMON/AA/ C(50),F(50),P(50),T(50),Y(50),IVARC(50),
> IVARF(50),IVARP(50),IVART(50),IVARY(50),KFAZ(50),MSUB
COMMON/NN/ NROWA,NROWJ,NL,NNL,INIT
DIMENSION CJAC(NROWJ,1),FUN(1),IVEC(7)
NNL=NNL+1
SUM=0.D0
DO 10 J=1,NS
IS=IVEC(J)
I=IABS(IS)
SIGN=IS/I
SUM=SUM+SIGN*F(I)*C(I)
CALL MATRIX (NNL,IVARC(I),SIGN*F(I),CJAC,NROWJ)
10 CALL MATRIX (NNL,IVARF(I),SIGN*C(I),CJAC,NROWJ)
FUN(NNL)=SUM
RETURN
END
```

```
SUBROUTINE MBAL (NS, IVEC, FUN, CJAC)
C TOTAL MASS BALANCE
IMPLICIT REAL*8 (A-H,O-Z)
COMMON/AA/ C(50),F(50),P(50),T(50),Y(50),IVARC(50),
> IVARF(50),IVARP(50),IVART(50),IVARY(50),KFAZ(50),MSUB
COMMON/NN/ NROWA,NROWJ,NL,NNL,INIT
DIMENSION CJAC(NROWJ,1),FUN(1),IVEC(7)
NNL=NNL+1
SUM=0.D0
DO 10 J=1,NS
IS=IVEC(J)
I=IABS(IS)
SIGN=IS/I
SUM=SUM+SIGN*F(I)
10 CALL MATRIX (NNL,IVARF(I),SIGN,CJAC,NROWJ)
FUN(NNL)=SUM
RETURN
END
```

```

SUBROUTINE HBAL (NS, IVEC, FUN, CJAC)
C ENTHALPY BALANCE
  IMPLICIT REAL*8 (A-H,O-Z)
  COMMON/AA/ C(50),F(50),P(50),T(50),Y(50),IVARC(50),
> IVARF(50),IVARP(50),IVART(50),IVARY(50),KFAZ(50),MSUB
  COMMON/CC/ H(50),DHC(50),DHP(50),DHT(50),DHY(50)
  COMMON/NN/ NROWA,NROWJ,NL,NNL,INIT
  DIMENSION CJAC(NROWJ,1),FUN(1),IVEC(7)
  NNL=NNL+1
  SUM=0.D0
  DO 10 J=1,NS
  IS=IVEC(J)
  I=IABS(IS)
  SIGN=IS/I
  SUM=SUM+SIGN*F(I)*H(I)
  CALL MATRIX (NNL,IVARF(I),SIGN*H(I),CJAC,NROWJ)
  CALL MATRIX (NNL,IVARC(I),SIGN*F(I)*DHC(I),CJAC,NROWJ)
  CALL MATRIX (NNL,IVARP(I),SIGN*F(I)*DHP(I),CJAC,NROWJ)
  CALL MATRIX (NNL,IVART(I),SIGN*F(I)*DHT(I),CJAC,NROWJ)
10 CALL MATRIX (NNL,IVARY(I),SIGN*F(I)*DHY(I),CJAC,NROWJ)
  FUN(NNL)=SUM
  RETURN
  END

```

```

SUBROUTINE QBAL (IUNIT,I1,I2,I3,I4,FUN,CJAC)
C HEAT BALANCE FOR HEAT EXCHANGER
IMPLICIT REAL*8 (A-H,O-Z)
COMMON/AA/ C(50),F(50),P(50),T(50),Y(50),IVARC(50),
> IVARF(50),IVARP(50),IVART(50),IVARY(50),KFAZ(50),MSUB
COMMON/CC/ H(50),DHC(50),DHP(50),DHT(50),DHY(50)
COMMON/EE/ Q(20),UA(20),IREV(20),IAFX(20),IVARA(20)
COMMON/NN/ NROWA,NROWJ,NL,NNL,INIT
COMMON/ZZ/ UROUND
DIMENSION CJAC(NROWJ,1),FUN(1)

C LET I1 = STATE PT. OF WORKING FLUID ENTERING HEAT EXCHANGER
C LET I2 = STATE PT. OF WORKING FLUID EXITING HEAT EXCHANGER
C LET I3 = STATE PT. OF NON-WORKING FLUID ENTERING EXCHANGER
C LET I4 = STATE PT. OF NON-WORKING FLUID EXITING EXCHANGER
C
NNL = NNL + 1
UA1 = UA(IUNIT)
TD1 = T(I1) - T(I4)
TD2 = T(I2) - T(I3)
IF (DABS(TD2) .LT. UROUND) GO TO 10
TD = TD1 - TD2
TR = TD1/TD2
IF (TR .GT. 0.01D0 .AND. DABS(TR - 1.D0) .GT. 0.01D0)
> GO TO 20

C USE ARITHMETIC MEAN TEMPERATURE DIFFERENCE
10 Q1 = UA1*0.5D0*(TD1 + TD2)
FUN(NNL) = Q1 - F(I3)*(H(I4) - H(I3))
CALL MATRIX (NNL,IVART(I1), 0.5D0*UA1,CJAC,NROWJ)
CALL MATRIX (NNL,IVART(I2), 0.5D0*UA1,CJAC,NROWJ)
CALL MATRIX (NNL,IVART(I3),-0.5D0*UA1+F(I3)*DHT(I3),
> CJAC,NROWJ)
CALL MATRIX (NNL,IVART(I4),-0.5D0*UA1-F(I3)*DHT(I4),
> CJAC,NROWJ)
CALL MATRIX (NNL,IVARA(IUNIT),0.5D0*(TD1+TD2),
> CJAC,NROWJ)
GO TO 25

C USE LOG MEAN TEMPERATURE DIFFERENCE
20 TRLN = DLOG(TR)
Q1 = UA1*TD/TRLN
FUN(NNL) = Q1 - F(I3)*(H(I4) - H(I3))
CALL MATRIX (NNL,IVART(I1), Q1/TD-Q1/(TRLN*TD1),
> CJAC,NROWJ)
CALL MATRIX (NNL,IVART(I2),-Q1/TD+Q1/(TRLN*TD2),
> CJAC,NROWJ)
CALL MATRIX (NNL,IVART(I3), Q1/TD-Q1/(TRLN*TD2)+
> F(I3)*DHT(I3),CJAC,NROWJ)
CALL MATRIX (NNL,IVART(I4),-Q1/TD+Q1/(TRLN*TD1)-
> F(I3)*DHT(I4),CJAC,NROWJ)
CALL MATRIX (NNL,IVARA(IUNIT),TD/TRLN,CJAC,NROWJ)
25 CALL MATRIX (NNL,IVARC(I3), F(I3)*DHC(I3),CJAC,NROWJ)
CALL MATRIX (NNL,IVARC(I4),-F(I3)*DHC(I4),CJAC,NROWJ)
CALL MATRIX (NNL,IVARP(I3), F(I3)*DHP(I3),CJAC,NROWJ)
CALL MATRIX (NNL,IVARP(I4),-F(I3)*DHP(I4),CJAC,NROWJ)
CALL MATRIX (NNL,IVARF(I3),-H(I4) + H(I3),CJAC,NROWJ)

C Q(IUNIT) = Q1
C
RETURN
END

```

```

SUBROUTINE VBAL (I,FUN,CJAC)
C EQUILIBRIUM VAPOR PRESSURE RELATION
IMPLICIT REAL*8 (A-H,O-Z)
COMMON/AA/ C(50),F(50),P(50),T(50),Y(50),IVARC(50),
> IVARF(50),IVARP(50),IVART(50),IVARY(50),KFAZ(50),MSUB
COMMON/NN/ NROWA,NROWJ,NL,NNL,INIT
DIMENSION CJAC(NROWJ,1),FUN(1)
NNL = NNL + 1
C
C SET ICALC TO SIGNAL THAT EQUILIBRIUM CALCS. ARE TO BE DONE
ICALC = 1
C
GO TO (10,20),MSUB
C
C LI-BR/WATER
10 CALL EGB1 (P(I),C(I),T(I),Y(I),KFAZ(I),ICALC,TE,
> DFP,DFT,DFC,H,DHP,DHT,DFC,DHY)
GO TO 30
C
C ALTERNATIVE WORKING FLUID (NCT CCDED)
20 CALL EGB2 (P(I),C(I),T(I),Y(I),KFAZ(I),ICALC,TE,
> DFP,DFT,DFC,H,DHP,DHT,DFC,DHY)
C
30 FUN(NNL) = TE - T(I)
CALL MATRIX (NNL,IVARC(I),DFC,CJAC,NROWJ)
CALL MATRIX (NNL,IVARP(I),DFP,CJAC,NROWJ)
CALL MATRIX (NNL,IVART(I),DFT,CJAC,NROWJ)
RETURN
END

```

```
SUBROUTINE MATRIX (I,J,AIJ,A,NROWA)
IMPLICIT REAL*8 (A-H,O-Z)
DIMENSION A(NROWA,1)
IF (J .EQ. 0) RETURN
A(I,J) = A(I,J) + AIJ
RETURN
END
```

```

SUBROUTINE EQB1 (PP,XX,TT,YY,KFAZ,ICALC,TE,
> DFP,DFT,DFX,H,DHP,DHT,DHX,DHY)
C LI-BR/WATER SYSTEM
C ICALC = 1, EQUILIBRIUM TEMPERATURE CALCULATION
C ICALC = 2, ENTHALPY CALCULATION
    IMPLICIT REAL*8 (A-H,O-Z)
    COMMON/SS/ CSCALE,FSCALE,PSCALE,TSCALE,YSCALE
C CONSTANTS FOR EQUILIBRIUM TEMPERATURE OF LI-BR SOLUTION
C T = A1*TS + B1
C TS = -2.*E/(D + SQRT(D*D - 4.*E*(C-LOG10(P)))) - F
C A1 = A10 + A11*X + A12*X*X + A13*X*X*X
C B1 = B10 + B11*X + B12*X*X + B13*X*X*X
    DATA A10,A11,A12,A13/-2.00755D0,0.16976D0,-3.133362D-3,
> 1.97668D-5/
    DATA E10,B11,B12,B13/321.128D0,-19.322D0,0.374382D0,
> -2.0637D-3/
    DATA C,D,E,F/6.21147D0,-2886.373D0,-337269.46D0,
> 459.72D0/
C CONSTANTS FOR ENTHALPY OF LI-BR SOLUTION
C H = A2 + B2*T + C2*T*T
C A2 = A20 + A21*T + A22*T*T + A23*T*T*T + A24*T*T*T*T
C B2 = B20 + B21*T + B22*T*T + B23*T*T*T + B24*T*T*T*T
C C2 = C20 + C21*T + C22*T*T + C23*T*T*T + C24*T*T*T*T
    DATA A20,A21,A22,A23,A24/-1015.07D0,79.5387D0,
> -2.358016D0,0.03031583D0,-1.400261D-4/
    DATA B20,B21,B22,B23,B24/4.68108D0,-3.037766D-1,
> 8.44845D-3,-1.047721D-4,4.80097D-7/
    DATA C20,C21,C22,C23,C24/-4.9107D-3,3.83184D-4,
> -1.078963D-5,1.3152D-7,-5.897D-10/
C CONSTANTS FOR ENTHALPY OF SATURATED LIQUID WATER
C H = A3 + B3*T + C3*T*T + D3*T*T*T
    DATA A3,E3,C3,D3/-32.2008D0,1.00999D0,-1.06632D-4,
> 3.16450D-7/
C CONSTANTS FOR HEAT OF VAPORIZATION OF SAT. LIQUID WATER
C H = A4 + B4*T + C4*T*T + D4*T*T*T
    DATA A4,E4,C4,D4/1094.18D0,-0.585952D0,2.48223D-4,
> -1.12395D-6/
C CONSTANTS FOR CP OF WATER VAPOR
C CP = B5 + C5*T + D5*T*T + E5*P + F5*P*P
    DATA B5,C5,D5,E5,F5/0.432655D0,1.56667D-4,-2.41935D-7,
> 1.12575D-3,-1.12182D-6/
C
C          I/P          O/P
C KFAZ = 1      SCLUTION          P AND X          T
C KFAZ = 2      WATER VAPOR        P                  T
C KFAZ = 3      PURE WATER        P                  T
C KFAZ = 4      SATURATED LIQUID-VAPOR WATER MIXTURE
C
C UNSCALE P,X,T,Y FOR CALCS. INTERNAL TO THIS ROUTINE
P = PP/PSCALE
X = XX/CSCALE
T = TT/TSCALE
Y = YY/YSCALE
C
C      GO TO (10,20,30,40),KFAZ
C
C LI-BR SOLUTCN
10 GO TO (11,12),ICALC
C EQUILIBRIUM TEMPERATURE CALCULATION
11 G = D + DSQRT(D*D - 4.D0*E*(C - DLOG10(P)))
    TS = -2.D0*E/G - F
    A1 = A10 + A11*X + A12*X*X + A13*X*X*X
    B1 = B10 + B11*X + B12*X*X + B13*X*X*X
    TE = A1*TS + B1
C
    DA = A11 + 2.D0*A12*X + 3.D0*A13*X*X
    DB = B11 + 2.D0*B12*X + 3.D0*B13*X*X
    DFP = 4.D0*A1*E*E/(DLOG(10.D0)*P*G*G*(G-D))

```

```

DFT = -1.00
DFX = DA*TS + DB
GO TO 100
C ENTHALPY CALCULATION
12 A2 = A20 + A21*X + A22*X*X + A23*X*X*X + A24*X*X*X*X
B2 = B20 + B21*X + B22*X*X + B23*X*X*X + B24*X*X*X*X
C2 = C20 + C21*X + C22*X*X + C23*X*X*X + C24*X*X*X*X
DA = A21 + 2.00*A22*X + 3.00*A23*X*X + 4.00*A24*X*X*X*X
DB = B21 + 2.00*B22*X + 3.00*B23*X*X + 4.00*B24*X*X*X*X
DC = C21 + 2.00*C22*X + 3.00*C23*X*X + 4.00*C24*X*X*X*X
H = A2 + B2*T + C2*T*T
DHP = 0.00
DHT = B2 + 2.00*C2*T
DHX = DA + DB*T + DC*T*T
DHY = 0.00
GO TO 200
C WATER VAPOR
20 G = D + DSQRT(D*D - 4.00*E*(C - DLCG10(P)))
TS = -2.00*E/G - F
DTS = 4.00*E/E/(DLOG(10.00)*P*G*G*(G-D))
GO TO (21,22),ICALC
C EQUILIBRIUM TEMPERATURE CALCULATION
21 TE = TS
DFP = DTS
DFT = -1.00
DFX = 0.00
GO TO 100
C ENTHALPY CALCULATION
22 CP = B5 + C5*T + D5*T*T + E5*P + F5*P*P
CPS = B5 + C5*TS + D5*TS*TS + E5*P + F5*P*P
HG = A3 + A4 + (B3+B4)*TS + (C3+C4)*TS*TS
> + (D3+D4)*TS*TS*TS
DHG = (B3+B4 + 2.00*(C3+C4)*TS + 3.00*(D3+D4)*TS*TS)*DTS
HI = (B5+E5*P+F5*P*P)*(T-TS) + C5/2.00*(T*T-TS*TS)
> + D5/3.00*(T*T-TS*TS)
DHI = (E5+2.00*F5*P)*(T-TS) - CPS*DTS
H = HG + HI
DHP = DHG + DHI
DHT = CP
DHX = 0.00
DHY = 0.00
GO TO 200
C LIQUID WATER
30 GO TO (20,32),ICALC
C ENTHALPY CALCULATION
32 H = A3 + B3*T + C3*T*T + D3*T*T*T
DHP = 0.00
DHT = B3 + 2.00*C3*T + 3.00*D3*T*T
DHX = 0.00
DHY = 0.00
GO TO 200
C SATURATED LIQUID-VAPOR WATER MIXTURE
40 GO TO (20,42),ICALC
C ENTHALPY CALCULATION
42 HL = A3 + B3*T + C3*T*T + D3*T*T*T
HG = HL + A4 + B4*T + C4*T*T + D4*T*T*T
H = (1.00 - Y)*HL + Y*HG
DHL = B3 + 2.00*C3*T + 3.00*D3*T*T
DHG = DHL + B4 + 2.00*C4*T + 3.00*D4*T*T
DHP = 0.00
DHT = (1.00 - Y)*DHL + Y*DHG
DHX = 0.00
DHY = -HL + HG
GO TO 200
C

```

```
C SCALE TE,DFP,DFX (DFT ALREADY OK)
100 TE = TE*TSCALE
      DFP = DFP*TSCALE/PSCALE
      DFX = DFX*TSCALE/CSCALE
      RETURN
C
C SCALE H,DHP,DHX,DHY (DHT ALREADY OK)
200 H = H*TSCALE
      DHP = DHP*TSCALE/PSCALE
      DHX = DHX*TSCALE/CSCALE
      DHY = DHY*TSCALE/YSCALE
      RETURN
C
END
```

```
SUBROUTINE EQB2 (P,X,T,W,KFAZ,ICALC,TE,  
> DFP,DFT,DFX,H,DHP,DHT,DHX,DHY)  
C THIS IS A DUMMY ROUTINE  
IMPLICIT REAL*8 (A-H,O-Z)  
RETURN  
END
```

## INTERNAL DISTRIBUTION

1. M. R. Ally
2. B. Beard
3. J. Braunstein
4. R. S. Carlsmith
5. K. W. Childs
- 6-7. R. L. Cox
8. J. S. Crowell
9. R. C. Devault
10. W. Fulkerson
11. S. I. Kaplan
12. M. A. Kuliasha
13. R. P. Leinius/G. E. Whitesides/  
Enrichment Technology Library
14. F. C. Maienschein
15. S. H. McConathy
16. M. R. Patterson
17. A. S. Quist
18. H. Perez-Blanco
- 19-20. Central Research Library
21. Document Reference Section - Y-12
- 22-23. Enrichment Technology Library
- 24-25. Laboratory Records
26. Laboratory Records, RC
27. ORNL Patent Section

## EXTERNAL DISTRIBUTION

28. W. J. Biermann, 45 Foxcroft Drive, Fayetteville, NY 13066
29. J. M. Calm, Electric Power Research Institute, P. O. Box 10412, Palo Alto, CA 94303
30. J. G. Carbonell (Energy Div. Advisory Committee), Assoc. Professor of Computer Science, Carnegie-Mellon University, Pittsburgh, PA 15213
31. G. Douglas Carver, Tennessee Valley Authority, 703 Power Building, Chattanooga, TN 37402
32. R. N. Chappell, U. S. Department of Energy, Idaho Operations Office, 785 DOE Place, Idaho Falls, ID 83402
33. D. C. Erickson, Energy Concepts Co., 627 Ridgely Ave., Annapolis, MD 21401
34. R. J. Fiskum, U. S. Department of Energy, CE-132 FORSTL, GF-217, 1000 Independence Avenue, SW, Washington, DC 20585.
35. Joel Gilbert, Dames & Moore, Inc., 7101 Wisconsin Ave., Suite 700, Bethesda, MD 20814
36. S. Malcolm Gillis, (Energy Div. Advisory Committee), Duke University, Public Policy and Economics, 4875 Duke Station, Durham, NC 27706
37. G. Grossman, Technion-Israel Institute of Technology, Faculty of Mechanical Engineering, Haifa, Israel
38. W. T. Hanna, Battelle Columbus Laboratories, 505 King Avenue, Columbus, OH 43201
39. F. C. Hayes, Trane Company, 3600 Pammel Creek Road, LaCrosse, WI 54601
40. Fritz R. Kalhammer, (Energy Div. Advisory Committee), Vice President, Electric Power Institute, P. O. Box 10412, Palo Alto, CA 94303
41. Alan Karp, Electric Power Research Institute, Energy Management and Utilization Division, P. O. Box 10412, Palo Alto, CA 94303

## EXTERNAL DISTRIBUTION (cont.)

42. R. E. Kasperson, (Energy Div. Advisory Committee), Professor of Government and Geography, Clark University, Graduate School of Geography, Worcester, MA 01610
43. M. Lessen, (Energy Div. Advisory Committee), Consulting Engineer, 12 Country Club Drive, Rochester, NY 14618
44. R. A. Macriss, Institute of Gas Technology, 3424 South State Street, Chicago, IL 60616
45. L. A. McNeely, 7310 Steinmeier Drive, Indianapolis, IN 46250
46. D. K. Miller, Borg-Warner Air Conditioning, Inc., P. O. Box 1592, York, PA 17405
47. J. I. Mills, EG&G-Idaho, Inc., P. O. Box 1625-WCB, Idaho Falls, ID 83415
48. B. A. Phillips, Phillips Engineering Company, 721 Pleasant Street, St. Joseph, MI 49085
49. R. Radermacher, University of Maryland, Department of Mechanical Engineering, College Park, MD 20742
50. W. J. Rebello, PAR Enterprises, 11928 Appling Valley Road, Fairfax, VA 22030
51. R. C. Reimann, Carrier Corp., P. O. Box 4808, Syracuse, NY 13221
52. S. L. Richlen, U. S. Department of Energy, Office of Industrial Programs, CE-141 FORSTL, 1000 Independence Avenue, SW, Washington, DC 20585
53. U. Rockenfeller, Rocky Research Company, 1453 Rawhide Road, Boulder City, NV 89005
54. J. N. Rogers, Division 8324, Sandia Laboratories, Livermore, CA 94550
55. J. D. Ryan, U. S. Department of Energy, CE-132 FORSTL, GF-217, 1000 Independence Avenue, SW, Washington, DC 20585.
56. D. S. Severson, Gas Research Institute, 8800 West Bryn Mawr Avenue, Chicago, IL 60631
57. Sam V. Shelton, Georgia Institute of Technology, Department of Mechanical Engineering, Atlanta, GA 30332
58. J. J. Tuzson, Gas Research Institute, 8800 West Bryn Mawr Avenue, Chicago, IL 60631
- 59-72. Energy Conservation Distribution, Oak Ridge National Laboratory, P. O. Box Y, Building 9102-2, Room 218, Oak Ridge, TN 37831
73. U. S. Department of Energy, Mathematics and Geosciences, Washington, DC 20545
74. U. S. Department of Energy, Office of the Assistant Manager for Energy, Oak Ridge Operations, Oak Ridge, TN 37830
- 75-104. U. S. Department of Energy, Office of Scientific and Technical Information, P. O. Box 62, Oak Ridge, TN 37831