THE LEVEL-ONE MODULES LIBRARY FOR DSNP:
Dynamic Simulator for
Nuclear Power-plants

ANL-CT-77-22
Rev. 02

Printed on June 10, 1979

NOTICE

This report was prepared as an account of work sponsored by the United States Government. Neither the United States nor the United States Department of Energy, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness or usefulness of any information, apparatus, product or process disclosed, or represents that its use would not infringe privately owned rights.

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

~~DISSEMINATION OF THIS DOCUMENT IS LIMITED~~
29
ARGONNE NATIONAL LABORATORY, ARGONNE, ILLINOIS

Operated under Contract W-31-109-Eng-38 for the
U. S. DEPARTMENT OF ENERGY

TECHNICAL MEMORANDUM

ANL-CT-77-22
Rev. 02
Mathematics and
Computers (UC-32)

THE LEVEL-ONE MODULES LIBRARY FOR DSNP:
Dynamic Simulator for Nuclear Power-plants

by

David Saphier

Components Technology Division

ARGONNE NATIONAL LABORATORY

September, 1978

DISTRIBUTION LIST.

<u>NAME</u>	<u>LOCATION</u>	<u>COPIES</u>		
		<u>M</u>	<u>C</u>	<u>L</u>
A. R. BRUNSVOLD	Argonne National Laboratory, (CT)	1		
B. CLYMER	Electronic Associates, Inc.	1	1	1
P. R. HUEBOTTER	Argonne National Laboratory, (CT)	1	1	1
J. J. KAGANOVE	Argonne National Laboratory, (AMD)	1		
T. W. KIRSCH	Argonne National Laboratory, (CT)	1	1	1
W. C. LIPINSKI	Argonne National Laboratory, (CT)	1		
A. B. LONG	Electric Power Research Institute	1	1	1
J. T. MADELL	Argonne National Laboratory, (CT)	1		
J. MECH	Argonne National Laboratory, (ENG)	1		
D. MOHR	Argonne National Laboratory, (EBR2)	1		
T. P. MULCAHEY	Argonne National Laboratory, (CT)	1		1
G. F. POPPER	Argonne National Laboratory, (CT)	1	1	1
R. PRIEMER	The U. of I. Circle Campus	1		
G. N. REDDY	Argonne National Laboratory, (EES)	1		
C. K. SANATHANAN	The U. of I. Circle Campus	1		
D. SAPHIER	Argonne National Laboratory, (CT)	1	1	1
N. F. SATHER	Argonne National Laboratory, (EES)	1		
A. W. SCHAEFER	Argonne National Laboratory, (CT)	1		
M. R. SIMS	Argonne National Laboratory, (CT)	5	5	5
J. H. TESSIER	Argonne National Laboratory, (RAS)	1		
D. K. WARINNER	Argonne National Laboratory, (RAS)	1	1	1
R. S. ZENO	Argonne National Laboratory, (CT)	1	1	1

M - DSNP Users Manual. ANL-CT-77-20

C - System - Precompiler Manual. ANL-CT-77-21

L - Library Manual - Description of Component Modules. ANL-CT-77-22

Printed on June 10, 1979

PREFACE

The Dynamic Simulator for Nuclear Power-plants (DSNP) is a system of programs and data sets by which a nuclear power plant or part thereof can be simulated at different levels of sophistication. The acronym DSNP is used interchangeably for the DSNP language, for the DSNP precompiler, for the DSNP libraries, and for the DSNP document generator.

The DSNP language is a set of simple block oriented statements, which together with the appropriate data, comprise a simulation of a nuclear power plant. The majority of the DSNP statements will result in the inclusion of a simulated physical module into the program. FORTRAN statements can be inserted with no restrictions among DSNP statements.

The DSNP precompiler is a FORTRAN program that will interpret the DSNP statements, rearrange the appropriate data sets, search the libraries for the requested modules, and produce an executable FORTRAN program - the plant simulation.

The DSNP libraries include models of most of the components found in nuclear power plants. There are three libraries of modules each having a different level of sophistication. A level-four library is also available to include user supplied modules.

The DSNP document generator produces all of the necessary documents for the above programs and libraries. The document describing the DSNP language is directed to the general user, one whose task is to use the available system to simulate a nuclear power plant. This document is actually a "Users Manual". The document describing the DSNP library is directed to the user who wishes to understand more profoundly the structure of the various modules and to optionally change, replace, or add modules to the library. The DSNP document that describes the precompiler is intended for the DSNP system programmer and gives the details of the subroutines used in the translation process.

The entire DSNP system can be obtained from the National Energy Software Center Library at the Argonne National Laboratory, 9700 South Cass Avenue, Argonne, Illinois, 60439 by requesting Package 784: DSNP, Dynamic Simulation Nuclear Power Plants.

This document describes the level-one DSNP library and was produced by the DSNP document generator. Dimensions of physical variables, wherever applicable, are all given consistently in SI metric units.

TABLE OF CONTENTS

INTRODUCTION	1
ACAMS1(LOOP)	3
BLCK1	6
BOILER1(LCGF)	8
CAVIT1(ZPCG,ZTAV,P1,P2,.....Pn,LCOP)	18
CNTRL1	21
CNVRG1(V0,V1,VNAM,VER,IT,ITMX,MOD,IPR,LCNV)	24
CONDU1(CO,ZMAT,TEMP,RO)	27
CORP1(LOOP,ZMATC)	29
CPSP1(CP,ZMAT,TEMP,PRES)	34
CPUTM1	36
DENST1(RO,ZMAT,TEMP,PRES)	38
DERIV1(LOOP)	40
DRUMS1(LOCF)	42
EUMP1(KFL,ZBLCK)	48
EULER1(LOOP)	51
FDEK1	53
FDWTR1(LOOP)	56
FLCW01(KF,ZPI,ZPX,KFITHX,LOOP)	59
FUNCT1(X,Y,KK,FX,FY,N,KLST)	63
GAMAP1(LOCF)	66
GAUSS1(N,M,A,D)	69
HPIPE1(KPH,ZW,ZTAV,PFAV,X,VALV)	71
HPUMP1(KZ,ZFIC,ZPFIN,ZTPIN,KITHX,LOOP)	75
IBXMS1(LOOP)	79
JUNC01(ZPJUN,ZF1,ZP2,.....ZPn,ZPJX,KFITHX,LCOP)	87
IAGFN1(X,Y,XF,YF,CL,K)	90
LOZZ1(X,Y,XX,YY,N,N1)	93

TABLE OF CONTENTS (Contd.)

LPLEN1 (LOOP)	95
MACRC1	98
MAINF1	102
MILNE1 (LOOP)	104
MOTOR1 (KP1, ZTORP, LOOP)	109
MXPLE1 (LCCP, F1in, T1in, ..., F1ex, T1ex, ..., ZVOIPL)	111
NFUTP1 (LOOP)	114
NFUTR1 (LOOP)	116
NEUTS1	120
NOINT1 (LOOP)	124
PIPRO1 (IP, FIN, FFX, VPLD, PF, ET)	126
PENUP1 (D, F, CP, CK, FM, ZRE, ZPR, ZPE)	131
RUNGC1 (LCCP)	133
RUNGF1 (LOOP)	136
SAFTY1	138
SINPS1 (LCCP)	141
SOCON1 (TEC, COS)	144
SODCP1 (TEC, CPS)	146
SODEN1 (TEC, RCC)	148
SODMU1 (TEC, SMU)	150
SOHXP1 (U, ZRE, ZFF, FI, RX, D, CK, P, PD)	152
SOHXS1 (U, ZRE, ZFR, FI, RX, D, CK, P, PD)	155
SSCON1 (TEM, CON)	157
SSDCP1 (TEM, CP)	159
SSDEN1 (TEC, RCC)	161
STCON1 (TEMP, CC, RO)	163
STDCP1 (TEC, CPS, P)	165
STDEN1 (TEC, FCC, F)	167

TABLE OF CONTENTS (Contd.)

STDMU1(VS,TEMP,FC)	169
STEPR1(ZZ,ZZP)	171
STHXP1(U,ZPE,ZPR,FI,RX,D,CK,P,PD)	174
STHXS1(U,ZRE,ZPR,FI,RX,D,CK,P,PD)	177
STIFF1(LOOP)	179
TEFNN1	184
TIMER1	186
TPHUL1(X,P,R)	190
TPOWR1	192
TPRIN1	194
TRAPZ1(LOOP)	196
UCCRC1(ZPE,PD,CK,FI,RX,D,U)	199
UCCRI1(CKF,CKC,FI,RX,U)	202
UPIEX1(RI,RX,DH,CK,ZNU)	204
UPLEN1(LOOP)	206
VISCO1(VS,ZMAT,TEMP,PO)	209
WACON1(TEMP,CC,FC)	211
WAECP1(TFC,CPS,P)	213
WADEN1(TEC,ROC,F)	215
WADNU1(VS,TEMP,FC)	217
WAHXP1(U,ZRE,ZPR,FI,RX,D,CK,P,PD)	219
WAHXS1(U,ZRE,ZPR,FI,RX,D,CK,P,PD)	221
WASAP1(VAR1,VAR2,QUC)	223
WASAT1(VAR1,VAR2,QUC)	225
WASHP1(VI1,VI2,VO1,VO2,Z,Z1,Z2,Z11,Z12,Z22,L)	227
WASHV1(VI1,VI2,VO1,VO2,Z,Z1,Z2,Z11,Z12,Z22,L)	231
WASIN1(VI1,VI2,TCID,VOLD,L)	235
WASPR1(IGO,VI1,VJ2,VO1,VO2,Z,Z1,Z2,Z11,Z12,Z22,L,IRET))	237

TABLE OF CONTENTS (Contd.)

WASPS1(VI1,VI2,VO1,VO2,Z,Z1,Z2,Z11,Z12,Z22,L)	241
WASPT1(VI1,VI2,VO1,VO2,Z,Z1,Z2,Z11,Z12,Z22,L)	245
WASPU1(VI1,VI2,VO1,VO2,Z,Z1,Z2,Z11,Z12,Z22,L)	248
WASPV1(VI1,VI2,VO1,VO2,Z,Z1,Z2,Z11,Z12,Z22,L)	252
WASR21(TCENT,V,F,DFDT,DFDV,DFDTT,DFDTV,DFDVV)	256
WASTE1(PSF,TCENT)	259
WASUV1(VI1,VI2,VO1,VO2,Z,Z1,Z2,Z11,Z12,Z22,L)	261
WBHXP1(ZRE,ZRP,P,CK,D,RI,FX,X,VF,VG,U)	265
WBHXS1(ZRE,ZRP,P,CK,D,RI,FX,X,VF,VG,U)	267
WSATF1(VI1,VI2,VO1,VO2,Z,Z1,Z2,Z11,Z12,Z22,L)	269
APPENDIX A: LIST OF SYMBOLS IN THE SYSTEM COMMON BLOCKS	272
APPENDIX B: LISTING OF THE SYSTEM COMMON BLOCKS	286

INTRODUCTION.

The level-one DSNP library is the basis of the DSNP system. The data set containing this library, unlike the level-two or level-three libraries, must always be present when a simulation program is submitted. The library contains four types of elements. The first types of elements are the control modules. These modules are used to generate the main program. `TIMER1`, `DEFIV1`, the I/O, and other control subroutines comprise the basic framework of the DSNP simulation program. These modules are processed by the precompiler and inserted into every DSNP FORTRAN simulation program. The final form, however, is dependent upon the particular program being generated. These modules have no specific format but most of them comprise a list of statements in FORTRAN, DSNP, or any arbitrary text requested by the DSNP system. They are stored in the library in the same sequence that they are requested by the precompiler in order to save search time.

The second types of elements are the basic system macros. These macros define the DSNP translation rules and therefore define the structure of the DSNP simulation language. The basic macros are stored at the end of the level-one library data set in a single module recognized as `**MACRO1`.

The third types and largest set of elements are the modules simulating the various components of a nuclear power plant. A component module can include a complete mathematical model of a particular component but, in many cases, several modules are requested to describe a power plant component. The nuclear reactor, for example, has a kinetic module, a decay-heat module, a power distribution module, a hydrodynamic module, a feedback module, and upper and lower plenum modules. The level-one library is comprised mostly of simple lumped parameter models of the various power plant components.

A component module can be in one of four forms. It can be a subroutine (which it is in the majority of cases), a function, a macro, or an arbitrary collection of statements. In certain instances, the module can be composed of several of the above forms. The modules are written in both DSNP and FORTRAN, and are subject to the translation process.

The fourth types of elements are the auxiliary functions requested in the simulation process that usually describe material properties or perform mathematical operations such as integration. These modules are either in the form of a subroutine or a function. The component modules and the auxiliary modules are arranged in alphabetic order.

Each module in the library has a six character identification name on its first record. The identifier is located in columns two to seven following the asterisk `**` in column one. The first five characters of the identifier are a symbolic name of the module. The sixth character is a number identifying the

library. For the level-one library, this number is one(1). Optionally, the number of records present in the module can be placed as a four digit integer in columns 9 to 12. If this number is absent, the module must be terminated by a FORTRAN FND statement beginning in column seven. Additional details on the structure of the various types of modules can be found in the users manual "The Simulation Language of DSNP", ANL-CT-77-20.

The description of the modules in this document follows a fixed paragraph pattern. The first paragraph briefly describes the purpose of the module. If the module is a function or a subroutine, then the second paragraph will list and describe the parameters. The third paragraph will list and describe the common variables and the fourth paragraph lists and describes all of the local variables used in the module. Paragraph five contains a list of subroutines, functions and common blocks required by the module. The mathematical equations and the algorithms used for the solution are given in paragraph six. The equations are written using FORTRAN notation and the variables used are those listed previously in paragraphs two, three and four. The module organization and the solution pattern are described in paragraph seven. Any iteration schemes or specific solution methods are described here. The eighth paragraph describes typical applications of the module and gives restrictions if any. Recommendations to appropriate integration schemes to be used with the module are sometimes given in this section. A list of references, if provided, would be found as the ninth paragraph.

The modules in this document are arranged alphabetically. Some of the modules are split into submodules so that additional system generated statements can be inserted between two submodules. Submodules are not described separately in this document.

```

*****
* ADAMS1 * (LOOP)
*****

```

(1) PURPOSE.

This subroutine integrates a vector of first order differential equations using the second order Adams algorithm with fixed or variable time steps.

(2) PARAMETERS.

LOOP An indicator of the loop on which this integration method will be used.

(3) COMMON VARIABLES.

<u>NAMES</u>	<u>COMBLK</u>	<u>DESCRIPTION</u>
DELT	TIMER	Current integration time step, in s.
DTMIN	TIMER	Minimum time step allowed, in s.
DYDT(I)	DERIV	Vector of the derivatives to be integrated, integrator input.
DYDT1(I)	DERIV	Intermediate or past derivative vector.
DYDT2(I)	DERIV	Intermediate or past derivative vector.
ERRMES(I)	TIMER	A vector of words to store simulation error messages.
IDIV	TIMER	If set to one, indicates one or more time step halving by an integration routine.
IDUB	TIMER	If set to one, indicates that the integration requests doubling of the time step.
ITERM	TIMER	If less than one, indicates that a simulation execution time error has been encountered.
LETE	TIMER	If one, use fixed time step. If two, use variable time step.

NCDER	DERIV	Number of integrations to be performed in the simulation.
TIME	TIMER	Current simulation time, in s.
YYYY(I)	DERIV	Vector of the integrated variables, integrator output.
YYYY1(I)	DERIV	Intermediate or past integration results.

(4) LOCAL VARIABLES.

DELT1	The previous integration time step interval.
EFMESL	The text "IN ADAMS".
ERR	The ratio of the computed to allowed integration errors.
JENTRY	The first entry indicator and is initially set to zero. It will be reset to one after the first entry.
KDB	Counts the number of integrations in which the error ratio ERR was less than 0.005.
KDU	Counts the total number of integrations performed in a loop.

(5) OTHER SUBROUTINES OR MODULES REQUIRED.

Subroutines: DERIV1, TRAPZ1.
Common Blocks: DERIV5, TIMER5, CONST5.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

The Adams second order integration algorithm used in this module allows fixed or variable time step integration as given by,

$$\begin{aligned}
 \text{YYYY}(\text{I}) = & \text{YYYY1}(\text{I}) + \text{DELT} * (\text{DYDT1}(\text{I}) - 0.5 * (\text{DYDT2}(\text{I}) - \\
 & \text{DYDT1}(\text{I})) * \text{DELT} / \text{DELT1}) \quad , \quad (1)
 \end{aligned}$$

and the integration error is estimated by,

$$\text{EYYYY3}(\text{I}) = \text{DABS}((\text{DYDT}(\text{I}) - \text{DYDT1}(\text{I})) / \text{DELT})$$

$$-(DYDT1(I) - DYDT2(I)) / DELT1) * DELT / 12 \quad . \quad (2)$$

(7) ORGANIZATION OF MODULE.

The Adams integration algorithm requires information from the previous time step, that is DYDT1 and YYYY1, which is not available in the first time step. The TRAPZ1 algorithm is therefore used to start the integration over the first time interval.

In each subsequent step, the derivatives and the function values from the last time step are stored prior to using Eq. (1). The following storage operations are performed.

$$\begin{aligned} YYYY1(J) &= YYYY(I) \\ DYDT2(I) &= DYDT1(I) \\ DYDT1(I) &= DYDT(I) \end{aligned} \quad (3)$$

The integration is performed only on those elements of the integration vector for which IYYYY(I)=LOOP. Next, the derivatives at TIME+DELT are calculated by calling DERIV1. If variable time step integration is requested, the integration error is evaluated by using Eq. (2). The ratio of calculated and allowed error, ERR, is then evaluated from,

$$ERR = EYYYY3(I) / (DABS(YYYY(I) * EYYYY1(I)) + EYYYY2(I)) \quad . \quad (4)$$

If ERR is greater than one for any of the integrations, the time step is halved and the integration is repeated setting the JDIV indicator to one. If ERR is less than 0.005 for all of the integrations, the doubling time request indicator JDUB is set to one.

If at any time DELT is reduced below DTMIN, an error message is produced and the simulation is terminated.

(8) APPLICABILITY AND RESTRICTIONS.

The Adams integration algorithm should be used in loops where slow to medium rate transients are expected. If it is used with fast transients, very small time steps can be expected. The integration algorithm can be used with fixed time steps where LETM equals one, in which case the error calculation part will be skipped.

* BLOCK1 *

(1) PURPOSE.

This is the BLOCKDATA subroutine that initializes the simulation program. There are no executable statements in this subroutine.

(2) PARAMETERS.

None.

(3) COMMON VARIABLES.

All of the common variables present in the simulation are included in the BLOCKDATA subroutine.

(4) LOCAL VARIABLES.

None.

(5) OTHER SUBROUTINES OR MODULES REQUIRED.

Common Blocks: ALL.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

None.

(7) ORGANIZATION OF MODULE.

The first five records of this module comprise the heading statements to be included in the subroutine. Next, all of the common blocks for the simulation are extracted by the system from the libraries and the users program. Then, the data statements that are produced by the system and that set all of the variables to zero are included and all of the data statements from the various default data sets, if present, are inserted in the same order as the user calls

them. Finally, the data statements produced by the users "DATA(BLOCK)=..." statements are properly translated and included as simple FORTRAN DATA statements. The last segment of the module includes the data for the CONST5 block.

(8) APPLICABILITY AND RESTRICTIONS.

This subroutine is always included in the simulation. The size depends upon the number of common blocks present in the system, the number and size of default data sets included, and the number of data statements.

```
*****
* BOILER* (ICOP)
*****
```

(1) PURPOSE.

This macro produces a subroutine BOILER# that solves the thermodynamics of a boiler and can be a part of any simulated steam generator.

(2) PARAMETERS.

ICOP Indicator to the loop from which this subroutine is called.

(3) COMMON VARIABLES.

<u>NAMES</u>	<u>COMBLK</u>	<u>DESCRIPTION</u>
JTIM	TIMER	Time step counter.
TEBPE#	BOIL#	Boiler primary fluid boiling region temperature, in C.
TEBPI#	BOIL#	Boiler primary side fluid inlet temperature, in C.
TEBPN#	BOIL#	Boiler primary fluid subcooled region temperature, in C.
TEBPX#	BOIL#	Boiler primary side exit temperature, in C.
TEBWB#	BOIL#	Boiler wall boiling region temperature, in C.
TEBWN#	BOIL#	Boiler wall nonboiling region temperature, in C.
ZEGIP#	BOIL#	Power transmitted into the boiler primary side, in W.
ZEGXB#	BOIL#	Power leaving the boiler secondary side, in W.
ZFBP#	BOIL#	Boiler primary side flow, in kg/s.
ZFD#	BOIL#	Boiler secondary or downcomer flow, in kg/s.

ZFR#	BCIL#	Boiler secondary riser flow, in kg/s.
ZHBI#	BCIL#	Secondary coolant inlet enthalpy, in J/kg.
ZHBSX#	BCIL#	Boiler exit enthalpy, in J/kg.
ZFBP#	BOIL#	Boiler primary side pressure, in Pa.
ZPBS#	BOIL#	Boiler secondary side pressure, in Pa.
ZREPX#	BOIL#	Boiler primary coolant exit density, in kg/m ³ .
ZRBSX#	BCIL#	Boiler steam water mixture exit density, in kg/m ³ .
ZTEBS#	BOIL#	Boiler secondary side saturation temperature, in C.
ZTED#	BOIL#	Boiler water inlet temperature, in C.
ZYNB#	BOIL#	Boiler subcooled part length, in m.
ZXQX#	BOIL#	Boiler exit steam quality.

(4) ICCAI VARIABLES.

ARIP	Primary side heat transfer area per unit length, in m.
ARIS	Secondary side heat transfer area per unit length, in m.
AFER	$ZXQA * (BERO - ZZZ1) / (ZXQA * (BERO - ZZZ1) + ZZZ1) ** 2$.
AFBP	Primary side flow cross sectional area, in m ² .
ARBS	Secondary (water) side flow cross sectional area, in m ² .
BBB	Conductivity of the steam-water boiling mixture, in W/m.
BFNB	Boiler subcooled region water conductivity, in W/m-C.
BCP	Average primary fluid conductivity, in W/m-C.
BERO	Steam water specific volume ratio, VBSS/VBSW.
BNAS	Change in boiling part steam-water mass, in kg/s.

CBB	Average specific heat of the steam-water mixture, in J/kg-C.
CENB	Subcooled region water specific heat, in J/kg-C.
CFP	Average primary fluid specific heat, in J/kg-C.
DZXNB	Time derivative of ZXNB#, in m/s.
DZXQA	Time derivative of ZXQA#, in m/s.
EPSA	Absolute error allowed in static calculations.
EFBR	Relative error allowed in static calculations.
FP	Flow in a primary subcooled channel, in kg/s.
FR	Flow in the upper part of the boiling region, in kg/s.
FS	Flow in a secondary channel, in kg/s.
F,F1 F2,F11 F12,F22	Derivatives of the Helmholtz function. Return parameters from the steam-water subroutines.
HPSS	Enthalpy of saturated steam, in J/kg.
HPSW	Enthalpy of saturated water, in J/kg.
ITI	Inner iteration counter.
ITIMX	Maximum number of allowed inner iterations.
ITO	External iterations counter.
ITOMX	Maximum number of allowed external iterations.
KBLC	Number of channels in the boiler.
KI1	Logical indicator of first entry to subroutine.
KZ	Parameter indicating water (100), or steam (200) properties. Used with the steam-water property functions.
PBS1	Secondary side pressure at the previous property evaluations, in Pa.
QBW	Weight of the boiler pipe per unit length, in kg/m.
PCBA	Average steam-water mixture density, in kg/m ³ .
RCBNA	Subcooled water density, in kg/m ³ .

ROBP	Average primary fluid density, in kg/m^3 .
RCBSS	Specific density of saturated steam, in kg/m^3 .
RCBSW	Specific density of saturated water, in kg/m^3 .
SEF	Viscosity of steam-water mixture, in Pa-s.
SENB	Subcooled region water viscosity, in Pa-s.
SEP	Average primary fluid viscosity, in Pa-s.
TEBNA	Secondary average subcooled water temperature, in C.
TEBNA1	TEBNA at previous evaluation, in C.
TEBPA	Primary coolant average temperature, or temperature at boiler water level, in C.
TEBPA1	Previous TEBPA, in C.
TEBPX1	Previous value of TEBPX#, in C.
TERS1	Value of ZTERS# at previous evaluation, in C.
UEBS	Boiling part wall to steam heat transfer per unit length, in W/m-C .
UBNS	Subcooled region wall to water heat transfer per unit length, in W/m-C .
UBP	Primary coolant to wall heat transfer per unit length, in W/m-C .
VEAV	Average steam-water specific volume, in m^3/kg .
VBSS	Specific volume of saturated steam, in m^3/kg .
VESW	Specific volume of saturated water, in m^3/kg .
VF	Volume of primary channel per unit length, in m^2 .
VS	Volume of secondary channel per unit length, in m^2 .
XDP	Primary hydraulic diameter, in m.
XDS	Secondary hydraulic diameter, in m.
ZEW	Boiler wall conductivity, in W/m-C .
ZCPW	Boiler wall specific heat, in J/kg-C .
ZHBA	Average steam-water mixture enthalpy, in J/kg-C .

ZLATH Latent heat of evaporation, in J/kg.
 ZMATH Primary fluid material identifier.
 ZMATS Secondary fluid material identifier.
 ZPDI Pitch to diameter ratio.
 ZPFB Prandtl number in secondary side boiling region.
 ZPRN Prandtl number in secondary side subcooled region.
 ZPEP Prandtl number of primary fluid.
 ZPEB Reynolds number in secondary side boiling region.
 ZPEN Reynolds number in secondary side subcooled region.
 ZPEP Reynolds number of primary fluid.
 ZROBW Boiler wall material density, in kg/m³.
 ZXB Boiler overall length, in m.
 ZXBB Boiling part length, in m.
 ZXNB1 Previous ZXNB#, in m.
 ZXQA Average steam quality in boiling channel.
 ZXQA1 ZXQA from previous iteration.
 ZXQX1 Previous value of ZXQX#.
 ZXRBI Boiler channel inner diameter, in m.
 ZXPBX Boiler channel external diameter, in m.

(5) OTHER SUBROUTINES OR MODULES REQUIRED.

Subroutines: CONDU1, CPSPE1, DENST1, RENUP1, STHXS1,
 VISCO1, WAHXS1, WASAT1, WASPT1, #HXP1.
 Common Blocks: BOIL#5, CONST5, DEPIV5, TIMER5.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

The boiler is subdivided into two axial regions; the boiling and the non-boiling regions. The boundary between these two regions is not fixed but is a time dependent

variable obtained from the heat balance equations. In the radial direction, there are three regions, namely the primary fluid, sodium or water, the secondary coolant, water or steam, and the boiler piping wall. All of the regions are lumped into a single node. The boiler therefore has six nodes for which the state variables are evaluated. For the boiler primary side boiling region, the steady state heat balance is given by,

$$CBP*FP*(TFBPI*-TEBPA)=UBP*ZXBB*(TEBPN*-TEBWB*) \quad , \quad (1)$$

where TEBPN* is given by,

$$TEBPN*=(TFBPA+TEBPI*)/2 \quad . \quad (2)$$

Inserting Eq. (2) into Eq. (1) and rearranging terms, the equation for TEBPA is obtained.

$$TEBPA=(TEBPI*[(CBP*FP-UBP*ZXBB)/2]+TEBWB*UBP*ZXBB)/[(CBP*FP+UBP*ZXBB)/2] \quad (3)$$

For the boiler primary side subcooled region, a similar heat balance equation is used.

$$CBP*FP*(TFBPA-TEBPX*)=UBP*ZXNB*(TEBPN*-TEBWN*) \quad , \quad (4)$$

where TEBPN is a linear average given by,

$$TEBPN=(TEBPA+TEBPX*)/2 \quad . \quad (5)$$

Inserting Eq. (5) into Eq. (4) and rearranging the terms, the equation for TEBPX* is obtained.

$$TEBPX*=(TEBPA*(CBP*FP-UBP*ZXNB)/2+TEBWN*UBP*ZXNB)/[(CBP*FP+UBP*ZXNB)/2] \quad (6)$$

The wall steady state heat balance equation is given by,

$$UBP*(TEBPN*-TEBWB*)=UBBS*(TEBWB*-ZTEBS*) \quad , \quad (7)$$

from which the equation for the initial value of TEBWB* is obtained by rearranging the terms.

$$TEBWB*=(UBP*TEBPN*+UBBS*ZTEBS*)/(UBP+UBBS) \quad (8)$$

For the second wall node, an expression identical to Eq. (7) is used and after rearranging the terms, the second wall node temperature is given by,

$$TEBWN*=(UBP*TEBPN*+UBNS*TEBNA)/(UBP+UBNS) \quad . \quad (9)$$

On the boiler secondary or water side, the two parameters of importance are the boundary location between the boiling and nonboiling region, and the exit steam quality. These are computed as follows. The heat balance for the

subcooled region is,

$$FS*CBNR*(ZTEBS*-ZTED*)=UBNS*ZXNB*(TEBWN*-TEBNA) \quad , \quad (10)$$

from which the boundary location ZXNB is obtained.

$$ZXNB*=FS*CBNR*(ZTEBS*-ZTED*)/UBNS*(TEBWN*-TEBNA) \quad (11)$$

The boiling part length is then obtained from,

$$ZXBB=ZXB-ZXNB \quad . \quad (12)$$

For the two phase boiling region, it is assumed that all of the heat transferred into this node is converted into steam. The pressure is assumed to be constant throughout the boiler and so are the saturation properties. The heat balance equation is therefore,

$$FS*ZXQX**ZLATH=URSS*ZXBB*(TEBWB*-ZTEBS*) \quad , \quad (13)$$

from which the steady state steam exit quality is obtained.

$$ZXQX*=URSS*ZXBB*(TEBWB*-ZTEBS*)/(FS*ZLATH) \quad (14)$$

The dynamic boiler equations are obtained from the dynamic energy balance. For the primary side nonboiling region, the equation is,

$$TEBPP*=INTEGRAL[CBP*FP*(TEBPI*-TEBPA)-UBP*ZXBB*(TEBPN*-TEBWB*)]/(CBP*ROBP*ZXBB*VP) \quad , \quad (15)$$

and by assuming a linear temperature change,

$$TEBPA=2*TEBPP*-TEBPI* \quad . \quad (16)$$

For the primary side nonboiling region, the temperature is,

$$TEBPN*=INTEGRAL[CBP*FP*(TEBPA-TEBPI*)-UBP*ZXNB*(TEBPN*-TEBWN*)]/(CBP*ROBP*ZXNB**VP) \quad , \quad (17)$$

and again assuming a linear temperature change within the region,

$$TEBPX*=2*TEBPN*-TEBPA \quad . \quad (18)$$

For the boiler walls, the temperatures are calculated from the two similar energy balance Eqs. (19) and (20).

$$TEBWB*=INTEGRAL[UBP*(TEBPE*-TEBWB*)-UBBS*(TEBWB*-ZTEBS*)]/(ZROBW*ZCPW) \quad (19)$$

$$TEBWN*=INTEGRAL[UBP*(TEBPN*-TEBWN*)-UBNS*(TEBWN*-TEBNA)]/(ZROBW*ZCPW) \quad (20)$$

On the boiler secondary side, the variables integrated are the length of the subcooled region and the steam exit quality. It is assumed that saturation temperatures prevail throughout the boiling region and that the steam quality is pressure dependent only. The pressure is obtained from the momentum balance equations solved in the hydrodynamic part. The time derivative of the subcooled boiler length is given by,

$$DZXNB = (UBNS * ZXNB * (TEBWN - TEBNA) - FR * CBNB * (ZTEPS - ZTED)) / (TEBNA * ROBNA * CBNB) \quad (21)$$

and by integrating Eq. (21), the nonboiling length is obtained.

$$ZXNB = \text{INTEGRAL}[DZXNB] \quad (22)$$

By subtracting the nonboiling length from the boiler total length, the boiling part length is obtained.

$$ZXBB = ZXB - ZXNB \quad (23)$$

The energy transfer into the boiling region will change the quantity of water evaporated and the steam exit quality will vary. For the dynamic energy balance, one can write,

$$d/dt[ZLATH * BMAS * ZXQA] = UBBS * ZXBB * (TEBWB - ZTEBS) - FR * ZLATH * ZXQX \quad (24)$$

Now by expanding the derivative on the left hand side of the above equation, yields,

$$d/dt[ZLATH * BMAS * ZXQA] = ZLATH * (BMAS * dZXQA/dt + ZXQA * dBMAS/dt) \quad (25)$$

$$dBMAS/dt = d/dt[ZXBB * ROBA * VS] = VS * (ZXBB * dROBA/dt - ROBA * DZXNB) \quad (26)$$

$$dROBA/dt = d/dt[1 / ((ZXQA * (VESS - VBSW) + VBSW)] = -dZXQA/dt * ABER / ZXQA \quad (27)$$

where,

$$ABER = ZXQA * (BERO - 1) / (ZXQA * (BERO - 1) + 1) \quad (28)$$

and,

$$BERO = VESS / VBSW \quad (29)$$

Inserting Eqs. (27), (28) and (29) into Eq. (25), the two differential equations,

$$dBMAS/dt = -VS * (ROBA * DZXNB + ZXBB * DZXQA * ABER / ZXQA) \quad (30)$$

and,

$$\frac{d}{dt}[ZIATH*BMAS*ZXQA]=dZXQA/dt*(ROBA-ABER)*ZXBB*VS \\ *ZIATH-ZLATH*ZXQA*ROBA*VS*DZXNB) \quad , \quad (31)$$

are obtained from which the derivative for the average steam quality is finally calculated.

$$DZXQA=(UBES*ZXBB*(TEBWB-ZTEBS)-FR*ZLATH*ZXQA*ZIATH \\ *ZXQA*ROBA*VS*DZXNB)/((ROBA-ABER)*ZXBB*VS*ZLATH) \quad (32)$$

The riser flow FR may differ somewhat from the downcomer flow during fast transients and is given by,

$$FR=FS-dBMAS/dt \quad . \quad (33)$$

(7) ORGANIZATION OF MODULE.

This module has three sections. An initialization section, an initial conditions(IC) computation section and a dynamic section. The initialization section is executed upon first entry only. Here the basic parameters of the boiler are computed from the data supplied. In the IC section, an iterative process is used to solve Eqs. (1) to (14). An inner iteration loop will converge on the boiler subcooled length ZXNB# and an outer iteration loop will converge on the boiler exit quality of the two phase mixture.

(8) APPLICABILITY AND RESTRICTIONS.

This boiler module is a two region, lumped parameter model with a variable boundary between the two regions. The module is a macro that, when used by the system, will produce a subroutine, or several subroutines, simulating the thermodynamic equations of a counterflow boiler. The lower or nonboiling node is the preheater section in which the water is heated up to the saturation temperature by the primary coolant. In the upper node, boiling takes place under saturated conditions with no superheating and a mixture of saturated water and steam leaves the boiler.

This component can be made part of any steam generator simulated by DSNP. To use this module, the following definition should be included in the program declarations segment.

• DFBOILA1(KELC,ZMATP,ZMATS,ZPDI,ZXRBI,ZXRBX,ZXB,
FPSA,FPSR,ITIMX,ITOMX,ZROBW,ARBP,ZBW,ZCPW);

where a is an alphanumeric single character name assigned

by the user to the a particular boiler being simulated. The above definition will create a subroutine simulating the boiler. To use this module in the simulation, the following statement should be included.

- BOILA1;

This statement has to be included in both the static and dynamic part of the simulation. Additional data might have to be supplied to this module by using the "DATA(BOILA)=..." statement, depending upon how the module is included in the simulation.

```

*****
* CAVIT1 *   (ZPCG,ZTAV,P1,P2,.....Pn,LOOP)
*****

```

(1) PURPCSE.

This module contains macros and DSNP statements producing a subroutine for each hydraulic cavity defined by DFHCVTii. The subroutine calculates the fluid level in the cavity and the pressure at each of the pipe inlets or outlets P1,P2, to Pn.

(2) PARAMETERS.

LCOP Indicator of the loop from which this
 subroutine is called.

P1,P2...PN Pressure at the pipe inlet or outlet, in Pa.

ZPCG Cover gas pressure in the cavity, in Pa.

ZTAV Average temperature of the fluid in the
 cavity, in C.

(3) COMMON VARIABLES.

<u>NAMES</u>	<u>COMBLK</u>	<u>DESCRIPTION</u>
JTIM	TIMER	Time step counter.
ZDXH#	HCA##	Level change due to flow variations, in m.
ZFLOW(I)	HFLOW	Magnitude of the flow in the Ith flow path, in kg/s.
ZCHPIP(I)	HPIPE	Mass of fluid in the pipe, in kg.
ZXHC#	HCA##	Fluid level in a cavity, in m.

(4) LOCAL VARIABLES.

KFNT First entry logical indicator.

ZARCAV Cross sectional area of the cavity, in m².

ZFLCI Sum of all of the inlet flows, in kg/s.

ZFLCX Sum of all of the exit flows, in kg/s.
 ZQHCV Same as ZXHC#, in m.
 ZCHLOP Total weight of the fluid in the flow loop of
 which this cavity is a part, in kg.
 ZROCV Average density of the fluid in the cavity, in
 kg/m³.
 ZTAV1 Average temperature of the fluid used in the
 last density evaluation, in C.
 ZVCAV Volume of the cavity, in m³.
 ZXHCV0 Initial fluid level, in m.
 ZXHIE(I) Pipe inlet or outlet elevation relative to the
 bottom of the cavity, in m. The dimension I is
 set by the precompiler.

(5) OTHER SUBROUTINES OR MODULES REQUIRED.

Subroutines: SODEN1 or WADEN1.
 Common Blocks: CONST5, DERJVS, HCA##5, HFLOW5, HPIPE5,
 TIMER5.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

The fluid level in the cavity changes either due to a
 difference between the total inlet and outlet flow, or due
 to a change in the fluid density somewhere along the flow
 path connected to the cavity. Equations (1) and (2)
 compute these changes.

$$ZDXH\# = \text{INTEGRAL} [(ZFLCI - ZFLCX) / (ZARCV * ZROCV)] \quad (1)$$

$$ZXHC\# = (ZQHIOF - \text{SUM} [ZQHPIP(I)]) / (ZARCAV * ZROCV) + ZDXH\# \quad (2)$$

The hydrostatic pressure at the pipe entrance or exit is
 given by,

$$PI = ZPCG + (ZXHCV - ZXHIE(I)) * ZROCV * 9.81 \quad (3)$$

(7) ORGANIZATION OF MODULE.

Upon entering the subroutine, the temperature ZTAV1 at
 which the fluid density was last evaluated is compared to

ZTAV. If the difference is greater than 1 C., the fluid density is recalculated.

Upon first entry only, the initial fluid inventory in the loop is computed by summing the weight of the fluid of each pipe connected to the cavity loop.

During the IC calculations, only Eq. (2) is evaluated. During the dynamic computations, the total inflow and outflow is first summed and then the integration of Eq. (1) is performed. Finally, the hydrostatic pressure at the entrance to each pipe is computed.

(8) APPLICABILITY AND RESTRICTIONS.

The HCVT# subroutine is used to simulate a hydraulic network when a plenum in a core or heat exchanger exists in which the level of the fluid can change, and the cover gas pressure is given as a parameter of the subroutine. A subroutine is generated for each cavity defined by the DSNP statement,

- DECAVTii(HO,ZPCG,ZTAV,ACV,INFLO(Wi,Hi,Pi,...),EXPLOW
(Wj,Hj,Pj,...),PIPE5(K1,K2,...KN),V,MAT): ,

where ii is the two digit identifying number for the cavity, HO is the numerical value for ZXHCVO, and ACV is the numerical value for ZARCAV. Wi, Hi, and Pi are the inlet flow rate, entry elevation value, and hydrostatic pressure for the ith entry. Wj, Hj, and Pj are the same variables for the jth outgoing flow. There is no limit to the number of flows that can enter or leave a cavity. PIPES(K1,K2,...KN) indicate the pipe numbers that comprise the flow paths passing through this cavity. V is the numerical value for the cavity volume ZVCAV and MAT is the fluid material indicator. It can be SO for sodium, WA for water, or ST for steam. To use this module in the simulation, the following statement should be included.

- CAVTii;

```

*****
* CNTRL1 *
*****

```

(1) PURPOSE.

This subroutine calculates the reactivity generated by the control system and reactivity perturbations inserted externally.

(2) PARAMETERS.

None.

(3) COMMON VARIABLES.

<u>NAMES</u>	<u>COMBIK</u>	<u>DESCRIPTION</u>
EXREC(I)	CNTRL	External reactivity inserted to initiate a transient, in absolute units.
IEXR	CNTRL	Indicates which type of external reactivity inserted. 1-step, 2-ramp, 3-function.
JTIM	TIMER	Time step counter.
KDEGFR	CNTRL	Interpolation level to be used with external reactivity function.
KREFUN	CNTRL	Number of values in the given reactivity function.
RRCN	CNTRL	Reactivity generated by the control system.
TCNTRL	TIMER	Time at which the change in external control reactivity is activated.
TIME	TIMER	Current simulation time, in s.

(4) LOCAL VARIABLES.

EXRAMP External ramp reactivity coefficient.

JRC Points to the external reactivity value to be used

from the EXPEC vector.

KLIST If the external reactivity is given as an arbitrary function, KLAST points to the last function value in EXPEC used in the interpolation process.

RKRAMP Reactivity contributed by the ramp function in the previous ramp interval.

TIME1 The time at which the previous reactivity evaluation took place, in s.

TPAMP The time at which the slope of the ramp reactivity changes, in s.

(5) OTHER SUBROUTINES OR MODULES REQUIRED.

Subroutines: FUNCT1.

Common Blocks: CONST5, CNTRL5, TIMEF5.

Functions: RAMP.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

For a step change in reactivity, IEXR=1, and for $TIEXR(JRC) \leq TIME < TIEXR(JRCH)$,

$$RKCEN = RKCEN + EXPEC(JRC) \quad . \quad (1)$$

For a ramp change in reactivity, IEXR=2, and for $TIEXR(JRC) \leq TIME < TIEXR(JRCH)$,

$$RKCEN = RKRAMP + EXPEC(JRC) * RAMP(TIEXR(JRC)) \quad . \quad (2)$$

For an arbitrary external reactivity function, IEXR=3, a second order functional interpolation is performed.

$$RKCEN = FUNCT1(TIEXR, EXPEC, TIME) \quad (3)$$

(7) ORGANIZATION OF MODULE.

During the IC calculations, the external reactivity is zero. For a dynamic entry and in order to save computations, a new external evaluation is made only if the time of the present entry is different from the time of the previous entry. Then, according to the value of IEXR, one of the above three equations is evaluated.

(R) APPLICABILITY AND RESTRICTIONS.

At present, only three external modes of reactivity can be inserted into the core in order to initiate a transient. These are a step, a ramp, and an arbitrary function. Any control algorithm, if required, can be inserted by writing the proper equations and providing the COMIN variables that will drive the control algorithm.

In order to initiate a transient, the following values must be supplied as input data by the user.

- a) For a step perturbation: IEXF=1, EXREC(I) and TIEXR(I).
- b) For a ramp perturbation: IEXF=2, EXREC(I) and TIEXR(I).
- c) For an arbitrary reactivity function: IEXR=3, KREFUN, EXREC(I) and TIEXR(I).

To use this module in the simulation, the following statement should be included.

• CNTRL1;

```

*****
* CNVRG1 *   (V0,V1,VNAM,VER,IT,ITMX,MOD,IPR,LCNV)
*****

```

(1) PURPOSE.

This subroutine performs a convergence check on the variable V0.

(2) PARAMETERS.

IPR Alphanumeric printing indicator. If the character P is entered, printing of each iteration will result.

IT Counter of the number of iterations.

ITMX Maximum number of iterations allowed to converge the V0 variable.

LCNV Subroutine output. A zero indicates that the process has converged; a one indicates no convergence; a two indicates the maximum number of iterations were exceeded.

MCD The mode of the convergence testing criteria. A indicates absolute and R indicates relative.

VER Maximum error allowed in the convergence process.

VNAM Alphanumeric name of the variable on which the convergence check is made.

V0 Variable for which the convergence is tested.

V1 Value of V0 from the previous iteration.

(3) COMMON VARIABLES.

<u>NAMES</u>	<u>COMBLK</u>	<u>DESCRIPTION</u>
IC6	TIMER	Standard FORTRAN output unit (usually 6).

(4) LOCAL VARIABLES.

EPS Absolute or relative discrepancy between V0 and V1.

(5) OTHER SUBROUTINES OR MODULES REQUIRED.

None.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

The convergence criterion can be applied either to the variable absolute error or to the variable relative error. In the first case, the absolute value of the difference between the present and the previous values are computed by,

$$EPS = DABS(V0 - V1) \quad . \quad (1)$$

In the second case, the absolute relative value of the difference is computed by,

$$FPS = DABS((V0 - V1)/V0) \quad . \quad (2)$$

(7) ORGANIZATION OF MODULE.

Upon entering the subroutine, a check is first made to determine whether the iteration status should be printed. Then, a check is made to find out whether the maximum number of iterations was exceeded. If MOD is A, then the absolute error is computed by Eq. (1). If MOD is R, then the relative error is computed from Eq. (2). This error is then compared to the maximum error allowed for convergence, VER. If the computed error is less, then LCNV is set equal to zero and if the computed error is greater, then LCNV is set equal to one.

(8) APPLICABILITY AND RESTRICTIONS.

The convergence subroutine is usually invoked by a macro call. For the absolute convergence criteria use,

• CONVERGA(V0,VER,LBL,ITMX,P or N); ,

and for relative convergence criteria use,

• CONVERGR(V0,VER,LBL,ITMX,P or N); ,

where LBL is the label to which the control is to be returned if convergence is not achieved. P or N is for Printing or No printing of the iteration process. If the variable did not converge within the predetermined error limit, the simulation will be terminated. These macro calls can only be used from the main body of the DSNP

program. Alternatively, the FORTRAN CALL statement can be used as,

```
CALL CNVFG1(VC,V1,'VNAME',VER,IT,ITMX,'MOD','IPR',LCNV) ,
```

for convergence testing in any iterative calculation.

```

*****
* CONDU1 *   (CC,ZMAT,TEMP,RO)
*****

```

(1) PURPOSE.

This subroutine calculates the conductivity of the material ZMAT by calling the appropriate material property subroutine.

(2) PARAMETERS.

CC calculated conductivity, in W/m-C.

RO Material density, in kg/m³.

TEMP Temperature at which the conductivity is to be evaluated, in C.

ZMAT Alphanumeric indicator to the material for which the conductivity is to be evaluated.

(3) COMMON VARIABLES.

<u>NAMES</u>	<u>COMBLK</u>	<u>DESCRIPTION</u>
IO6	TIMER	Standard FORTAN output unit (usually 6).
ZZZSO	CONST	"SO" indicating sodium properties.
ZZZSS	CONST	"SS" indicating stainless steel properties.
ZZZST	CONST	"ST" indicating steam properties.
ZZZWA	CONST	"WA" indicating water properties.

(4) LOCAL VARIABLES.

V Specific volume at TEMP, in m³/kg.

(5) OTHER SUBROUTINES OR MODULES REQUIRED.

Subroutines: SOCON1, SSCON1, STCON1, WACON1.
Common Blocks: CONST5, TIMER5.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

The conductivity of the requested material is evaluated by calling the appropriate material property subroutine.

(7) ORGANIZATION OF MODULE.

The content of ZMAT is tested for the indicator of the material for which the conductivity is to be evaluated. If ZMAT=SO, the conductivity of sodium is computed by calling SOCON1. If ZMAT=SS, the conductivity of stainless steel is computed by calling SSCON1. If ZMAT=WA or ST, the conductivity of water or steam is computed by calling WACON1 or STCON1. If ZMAT does not contain any of the above names, an error message is printed.

(8) APPLICABILITY AND RESTRICTIONS.

At present, the conductivity of only four materials, as shown above, can be computed. The number can, however, be easily expanded by adding additional material property subroutines. The RO parameter is required only for water or steam property calculations. To use this module in the simulation, the following statement should be included.

CALL CONDU1(CO,ZMAT,TEMP,RO)

```

*****
* CORTP1 *   (LCOP,ZMATC)
*****

```

(1) PURPOSE.

This subroutine calculates the lumped parameter core and blanket thermodynamics.

(2) PARAMETERS.

LCOP Indicator to the loop from which this subroutine is called.

ZMATC Alphanumeric indicator to the coolant flowing through the core; WA for water, SO for sodium, and ST for steam.

(3) COMMON VARIABLES.

<u>NAMES</u>	<u>COMBIK</u>	<u>DESCRIPTION</u>
CC	CORTP	Primary coolant specific heat, in J/kg-C.
CF	CORTP	Core fuel specific heat, in J/kg-C.
CL	CORTP	Fuel cladding specific heat, in J/kg-C.
GA	GAMAR	Fraction of the decay heat liberated in the coolant.
JTIM	TIMER	Time step counter.
QMF	CORTP	Total weight of fuel in the core, in kg.
QML	CORTP	Total weight of cladding in the core, in kg.
ROCA	CORTP	Coolant average density in the core, in kg/m ³ .
TECA	CORTP	Core coolant average temperature, in C.
TECA0	CORTP	Initial TECA, in C.

TECI	CORTP	Core coolant inlet temperature, in C.
TECX	CORTP	Core coolant exit temperature, in C.
TEFA	CORTP	Core fuel average temperature, in C.
TEFAO	CORTP	Initial TEFA, in C.
TELA	CORTP	Core cladding average temperature, in C.
UCA	CORTP	Total heat transfer between the fuel and the cladding, in W/C.
ULA	CORTP	Total heat transfer between the fuel cladding and the core coolant, in W/C.
VC	CORTP	Volume of coolant in the core, in m ³ .
WGT	GAMAR	Power produced by the decay of fission products, in W.
ZEGXC	CORTP	Power leaving the core through the coolant, in W.
ZFLCOR	CORTP	Flow through the core, in kg/s.
ZHCX	CORTP	Core exit enthalpy, in J/kg.
ZFCORA	CORTP	Average pressure in core, in Pa.
ZFOCX	CORTP	Core exit coolant density, in kg/m ³ .
ZTECEL	CORTP	Core coolant boiling temperature, in C.
ZVOLFC	CORTP	Core coolant void volume fraction.

(4) LOCAL VARIABLES.

CFM	CF*QMF	fuel heat capacity, in J/C.
CIM	CI*QML	cladding heat capacity, in J/C.
CVF	CC*VC*FOCA	coolant heat capacity, in J/C.
DTECX		Time derivative of the core exit temperature, in C/s.
DTELA		Time derivative of the cladding temperature TELA, in C/s.

DZHCX Derivative of ZHCX.

F,F1 Derivatives of the Helmholtz function. Return
F2,F11 parameters from steam-water subroutines.
F12,F22

HSS Enthalpy of saturated steam, in J/kg.

HSW Enthalpy of saturated water, in J/kg.

KZ Constant used with the WASP functions.

TECA1 Value of TECA at the previous integration time
step, in C.

TK Temperature, in K.

VCX Coolant specific volume, in m³/kg.

VSS Saturated steam specific volume, in m³/kg.

VSW Saturated water specific volume, in m³/kg.

ZIATH Coolant latent heat of evaporation, in J/kg.

ZQUAL Quality of the two phase flow leaving the core if
boiling takes place.

(5) OTHER SUBROUTINES OR MODULES REQUIRED.

Subroutines: CPSPE1, DENST1, SODFN1, WASPT1.
Common Blocks: CONST5, COPTP5, GAMAP5, TPOWR5.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

This is a simple lumped parameter model to calculate the core average fuel, cladding and coolant temperatures. The fuel temperature is given by,

$$TEFA = \text{INTEGRAL}[WQT + WGT*(1-GA) - UCA*(TEFA - TELA)]/CFM] \quad . (1)$$

The cladding temperature is given by,

$$TELA = \text{INTEGRAL}[UCA*(TEFA - TELA) - ULA*(TELA - TECA)]/CLM] \quad . (2)$$

The coolant core exit temperature is given by,

$$TECX = \text{INTEGRAL}[WGT*GA + ULA*(TELA - TECA) - GC*CC* \\ (TECX - TECI)]/CVR] \quad . (3)$$

If the coolant is water, the saturation temperature at pressure ZPCRA is computed by using the steam-water saturation property function WASAT1. The integral in Eq. (3) is limited between zero and the water saturation temperature ZTECBL. The water exit enthalpy is then obtained by,

$$ZHCX = \text{INTEGRAL}[(WGT*GA + UIA*(TEIA - TECA) - ZFLCOR* \\ (ZHCX - TECI*CC)) / (ROCA*VC)] \quad . \quad (4)$$

By knowing the saturation pressure, temperature, and the exit enthalpy, the quality of the steam in the two phase flow can be computed, if boiling takes place, by,

$$ZQUAI = (ZHCX - HSW) / (ZLATH * ZZZ2) \quad , \quad (5)$$

and the steam volume fraction is computed from,

$$ZVCIFC = ZQUAL * VSS / (ZQUAI * VSS + (1 - ZQUAL) * VSW) \quad . \quad (6)$$

(7) ORGANIZATION OF MODULE.

This module has a static initial condition part and a dynamic part. In the module STATIC section, the fuel cladding and core coolant exit temperatures are calculated from a single node heat balance equation. The initial conditions are computed from the coolant inlet temperature TCIN and the power produced in the core WOT+WGT. The equations used are,

$$TECX = TECI + (WOT + WGT) / (GC * CC) \quad , \quad (7)$$

$$TECA = (TECI + TECX) / 2 \quad , \quad (8)$$

$$TEIA = (TECA + (WOT + WGT * (1 - GA)) / UIA \quad , \quad (9)$$

$$TEFA = TEIA + (WOT + WGT * (1 - GA)) / UCA \quad . \quad (10)$$

If the coolant in the core is water, the steam-water property subroutine WASPT1 is called to calculate the coolant enthalpy at the exit of the core.

(8) APPLICABILITY AND RESTRICTIONS.

This module represents a simple lumped parameter approximation of the core thermodynamics. The fuel, the cladding, and the coolant are all represented by a single node. It should only be used in simple models where the details of the temperature distribution in the core are not necessary. If the module is used with the NEUTR1 kinetic module, the STIFF1 integration method is

recommended. If, however, the prompt jump approximation NFUTP1 is used, or the power is represented by an arbitrary function, any convenient integration method can be used. To use this module in the simulation, the following statement should be included.

- CORTE1;

```

*****
* CPSPE1 *   (CF,ZMAT,TEMP,PRES)
*****

```

(1) PURPOSE.

This subroutine calculates the specific heat of the material ZMAT by calling the appropriate material property subroutine.

(2) PARAMETERS.

CF Calculated specific heat, in J/kg-C.

PRES Pressure at which the specific heat of the material is evaluated, in Pa. Applicable only to water properties.

TEMP Temperature at which the specific heat is to be evaluated, in C.

ZMAT Alphanumeric indicator of the material for which the specific heat is to be evaluated.

(3) COMMON VARIABLES.

<u>NAMES</u>	<u>COMBLK</u>	<u>DESCRIPTION</u>
IC6	TIMER	Standard FORTRAN output unit (usually 6).
ZZZSO	CONST	"SO" indicating sodium properties.
ZZZSS	CONST	"SS" indicating stainless steel properties.
ZZZST	CONST	"ST" indicating steam properties.
ZZZWA	CONST	"WA" indicating water properties.

(4) LOCAL VARIABLES.

V Specific volume at TEMP, in m³/kg.

(5) OTHER SUBROUTINES OF MODULES REQUIRED.

Subroutines: SODCP1, SSDCP1, WASPT1.
Common Blocks: CONST5, TIMER5.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

The specific heat of the requested material is evaluated by calling the appropriate material property subroutine.

(7) ORGANIZATION OF MODULE.

The content of ZMAT is tested for the indicator of the material for which the specific heat is to be evaluated. If ZMAT=SO, the specific heat of sodium is computed by calling SODCP1. If ZMAT=SS, the specific heat of stainless steel is computed by calling SSDCP1. If ZMAT=WA or ST, the specific heat of water or steam is computed by calling WASPT1. If ZMAT does not contain any of the above names, an error message is printed.

(8) APPLICABILITY AND RESTRICTIONS.

At present, the specific heat of only four materials, as shown above, can be computed. The number can, however, be easily expanded by adding additional material property subroutines. The PRES parameter is required only for water or steam property calculations. To use this module in the simulation, the following statement should be included.

CALL CPSPE1(CP,ZMAT,TEMP,PRES)

```

*****
* CPUTM1 *
*****

```

(1) PURPOSE.

This module checks for the remaining CPU time. If less than TCPU seconds are left, the module will set ITERM equal to one and the simulation will be orderly terminated.

(2) PARAMETERS.

None.

(3) COMMON VARIABLES.

<u>NAMES</u>	<u>COMBLK</u>	<u>DESCRIPTION</u>
ITERM	TIMER	If less than one, indicates that a simulation execution time error has been encountered.
TCPU	TIMER	CPU time required for orderly termination of the simulation and initially set to five seconds.

(4) LOCAL VARIABLES.

None.

(5) OTHER SUBROUTINES OR MODULES REQUIRED.

Functions: TIMEF.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

None.

(7) ORGANIZATION OF MODULE.

Upon entering this module, the remaining CPU time for the program execution is computed by using the system function TLEFT. If this time is less than five seconds, ITERM is set to one. This will result in an orderly termination of the simulation instead of an operator issued cancel command because of an insufficient time request.

(8) APPLICABILITY AND RESTRICTIONS.

The TLEFT function used in this subroutine is installation dependant. It calculates the remaining time in hundreths of seconds. If a different system is used, this function should be replaced by an equivalent one.

```
*****
* DENST1 * (FC,ZMAT,TEMP,PRES)
*****
```

(1) PURPOSE.

This subroutine calculates the density of the material ZMAT by calling the appropriate material property subroutine.

(2) PARAMETERS.

PRES Pressure at which the density is to be evaluated, in Pa. Applicable only to water or steam.

FC Calculated density, in kg/m³.

TEMP Temperature at which the density is to be evaluated, in C.

ZMAT Alpharumeric indicator to the material for which the density is to be evaluated.

(3) COMMON VARIABLES.

<u>NAMES</u>	<u>COMPLX DESCRIPTION</u>
IO6	TIMER Standard FORTRAN output unit (usually 6).
ZZZSO	CONST "SO" indicating sodium properties.
ZZZSS	CONST "SS" indicating stainless steel properties.
ZZZST	CONST "ST" indicating steam properties.
ZZZWA	CONST "WA" indicating water properties.

(4) LOCAL VARIABLES.

F,F1 Parameters of the WASPT1 subroutine.
 F2,F11 Not used here.
 F12,F22

K Indicator for WASPT. If 100, evaluate water properties. If 200, evaluate steam properties.

TOUT Output temperature from WASPT1, in C.
V Specific volume at TEMP and PRES, in m³/kg.

(5) OTHER SUBROUTINES OR MODULES REQUIRED.

Subroutines: SODEN1, SSDEN1, WADEN1.
Common Blocks: CONST5, TIMEF5.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

The density of the requested material is evaluated by calling the appropriate material property subroutine.

(7) ORGANIZATION OF MODULE.

The content of ZMAT is tested for the indicator of the material for which the density is to be evaluated. If ZMAT=SO, the density of sodium is computed by calling SODEN1. If ZMAT=SS, the density of the stainless steel is computed by calling SSDEN1. If ZMAT=WA or ST, the density of water or steam is computed by calling WADEN1. If ZMAT does not contain any of the above names, an error message is printed.

(8) APPLICABILITY AND RESTRICTIONS.

At present, the density of only four materials, as shown above, can be computed. The number can, however, be easily expanded by adding additional material property subroutines. The PRES parameter is required only for water or steam property calculations. To use this module in the simulation, the following statement should be included.

CALL DENST1(RO,ZMAT,TEMP,PRES)

* DERIV1 * (LCCF)

(1) PURPOSE.

This subroutine coordinates the calculation of derivatives for the centralized integration subroutines.

(2) PARAMETERS.

ICOP Indicator of the loop for which the derivatives are to be calculated.

(3) COMMON VARIABLES.

All of the common variables belonging to the DSNP modules in which integration was requested will be inserted into this module. See Appendix A for a complete list.

(4) LOCAL VARIABLES.

I Loop indexes.

KZ Loop indexes produced by the precompiler.

(5) OTHER SUBROUTINES OR MODULES REQUIRED.

All of the subroutines and functions which require integration will be called from DERIV1.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

In this module, the integrated variables are updated at each integration time step and iteration. The principal function of DERIV1 is to use the various dynamic modules according to the loop to which they belong to produce the vector of system derivatives. It includes the actual translation of the various SIMULATE LOOPS created by the user in the DSNP program.

(7) ORGANIZATION OF MODULE.

Only the heading of this control module is stored in the library. The rest of the subroutine is produced by the precompiler.

The first part of the subroutine initializes all of the integration vectors prior to the calculation of the initial conditions. The initialization results in all of the vectors being set to zero and the error vectors, EYYYY1 and EYYYY2, to the default value of 0.005.

The initial conditions(IC) part of the subroutine is entered after the IC calculation has converged. The integration vector YYYY is now given the proper values according to the IC values of the appropriate variables. The integration error limit is set to the values requested by each of the integrators overriding the default values previously set.

The dynamic section of the subroutine has as many parts as the number of LOOPS defined by the user. Each part is accessed from the appropriate integration subroutine according to the parameter LOCP. In each dynamic part, the integrated variables are first updated to the latest integration result by making them equal to the appropriate YYYY(I). Next, the program section, as defined by each SIMULATE ICOP, is executed and the derivatives for the next time step are computed.

(8) APPLICABILITY AND RESTRICTIONS.

Any DSNP statement is allowed in the SIMULATE segment and will be properly translated and most probably inserted into DERIV1. There are no limitations on the number of statements or simulation loops. The practical limit is 98 loops, however, because of the number of LOOP labels available.

```

*****
* DRUMS1 *   (ICCP)
*****

```

(1) PURPOSE.

This macro produces a subroutine DFOM#1 that solves the thermodynamics of a steam drum and can be a part of any simulated steam generator.

(2) PARAMETERS.

LCOP Indicator to the loop from which this subroutine is called.

(3) COMMON VARIABLES.

<u>NAMES</u>	<u>COMMON</u>	<u>DESCRIPTION</u>
ZFDRI#	DRUM#	Steam drum steam-water mixture inlet flow, in kg/s.
ZFDRX#	DFUM#	Steam drum exit flow to the downcomer, in kg/s.
ZFDW#	DRUM#	Feedwater inlet flow, in kg/s.
ZFSTX#	DFUM#	Steam exit flow, in kg/s.
ZHDRI#	DRUM#	Steam drum inlet enthalpy, in J/kg.
ZHDRX#	DRUM#	Water average and exit enthalpy in the steam drum, in J/kg.
ZHFDW#	DFUM#	Feedwater inlet enthalpy, in J/kg.
ZHSTX#	DRUM#	Steam drum steam exit enthalpy, in J/kg.
ZFDR#	DFUM#	Pressure in the steam drum, in Pa.
ZODS#	DRUM#	Mass of steam in the steam drum, in kg.
ZODW#	DRUM#	Mass of water in the steam drum, in kg.
ZRODS#	DRUM#	Steam drum steam density, in kg/m ³ .

ZFODW#	DRUM#	Steam drum water density, in kg/m ³ .
ZTDRX#	DRUM#	Steam drum water exit temperature, in C.
ZTEDR#	DRUM#	Steam drum steam exit temperature, in C.
ZYDL#	DRUM#	Initial water level in the steam drum, in m.

(4) ICCAI VARIABLES.

FFL	Flashing or condensing rate, in kg/s.
FIDS	Incoming steam flow from the riser, in kg/s.
FIDW	Incoming water flow from the riser, in kg/s.
HDSS	Saturated steam enthalpy at the drum pressure, in J/kg.
HDSW	Saturated water enthalpy at the drum pressure, in J/kg.
P1	Steam drum pressure, same as ZPDR#, but resulting from a mass balance, in Pa.
QDS	Mass of the steam in the drum, in kg.
QDW	Mass of the water in the drum, in kg.
QUALIN	Quality of the steam in the two phase inlet steam-water mixture.
RCDS	Steam density, in kg/m ³ .
RCDW	Water density, in kg/m ³ .
TFFL	Steam flashing delay time, in s.
TEDR	Saturation temperature at the steam drum pressure, in C.
VDRM	Volume of the steam drum, in m ³ .
VDRS	Volume of the steam in the drum, in m ³ .
VDRW	Volume of the water in the drum, in m ³ .
VDS	Specific volume of the steam, in m ³ /kg.
VDSS	Specific volume of the saturated steam, in m ³ /kg.

VDSW Specific volume of the saturated water, in m^3/kg .

VDS1 Specific volume of the steam at the end of the iteration cycle, in m^3/kg .

VDW Specific volume of the water, in m^3/kg .

(5) CITER SUBROUTINES OR MODULES REQUIRED.

Subroutines: WASAT1, WASHP1, WASHV1, WASPT1.
Common Blocks: POIL#5, CONST5, DERIV5, DRUM#5, TIMER5.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

It is assumed that no heat transfer takes place in the steam drum between the water or steam and the wall. Initially or at equilibrium, all of the state variables are functions of the prevailing pressure.

The incoming two phase flow is separated into a steam flow given by,

$$FIDS = \text{QUALIN} * ZFDRI \quad , \quad (1)$$

and a water flow given by,

$$FIDW = (1 - \text{QUALIN}) * ZFDRI \quad . \quad (2)$$

The rest of the inlet and exit flows at steady state are set according to the above values.

$$ZFDW\# = FIDS \quad (3)$$

$$ZFSTX\# = FIDS \quad (4)$$

$$ZFDRX\# = FIDW \quad (5)$$

The enthalpy of the steam leaving the steam drum at equilibrium is for the saturated steam condition,

$$ZHSTX\# = HDSS \quad , \quad (6)$$

and the enthalpy of the water returned to the boiler inlet through the downcomer is given by,

$$ZHLRX\# = (ZFDW\# * ZHFDW\# + FIDW * HDSW) / ZFDRX\# \quad . \quad (7)$$

It is also assumed that at steady state no condensation or flashing takes place, so that FFL is equal to zero.

The dynamic mass balance is given by two equations; one

for the liquid phase,

$$dQDW/dt = FIDW + ZFDW\# - FFL - ZFDRX\# \quad , \quad (8)$$

and one for the vapor phase,

$$dQDS/dt = FIDS + FFL - ZFSTX\# \quad . \quad (9)$$

The dynamic energy balance equation gives the change in steam and water enthalpy,

$$\frac{dZHSTX\#}{dt} = \frac{(FIDS + FFL) * HDSS - (ZFSTX\# + dQDS/dt) * ZHSTX\#}{QDS} \quad . \quad (10)$$

For the water average enthalpy, the following equation is used.

$$\frac{dZHDPX\#}{dt} = \frac{(FIDW * HDSW - (ZFDRX\# + dQDW/dt) * ZHDPX\# - FFL * HDSS)}{QDW} \quad (11)$$

(7) ORGANIZATION OF MODULE.

Initial Conditions Section.

Upon entering the module, the saturation pressure ZPDR# corresponding to the drum pressure ZPDR# is computed by,

$$CALL \text{ WASAT1}(ZPDR\#, ZTEDR\#, \dots) \quad (12)$$

Next, the enthalpy of the saturated steam and saturated water are computed by,

$$CALL \text{ WASPT1}(ZPDR\#, ZTEDR\#, ZTEDR\#, VDSW, F, F1, F2, \dots) \quad (13)$$

$$HDSW = F - (ZTEDR\# + 273) * F1 - VDSW * F2 \quad (14)$$

$$CALL \text{ WASPT}(ZPDR\#, ZTEDR\#, ZTEDR\#, VDSW, F, F1, \dots) \quad (15)$$

$$HDSS = F - (ZTEDR\# + 273) * F1 - VDSS * F2 \quad . \quad (16)$$

The latent heat of evaporation is computed from,

$$ZLATH = HDSS - HDSW \quad . \quad (17)$$

Then, the inlet steam quality is computed by,

$$QUALIN = (ZHDPX\# - HDSW) / ZLATH \quad , \quad (18)$$

from which the separated steam and water inlet flows are obtained using Eqs. (1) and (2). The initial value of the various flows entering and leaving the steam drum are computed from Eqs. (3) to (5), and the initial enthalpy of

the steam and water outlet are computed by Eqs. (6) and (7).

To calculate the initial water and steam quantities in the steam drum, property function WASHP1 is used. This subroutine provides all of the thermodynamic properties once the enthalpy and pressure are known.

$$\text{CALL WASHP1}(ZHDX\#,ZPDF\#,ZTDRX\#,VDW,...) \quad (19)$$

The volumes of water and steam in the steam drum are then obtained from the initial water quantity ZQDW# which is given as input by,

$$VDFW=ZQDW\#*VDW \text{ and,} \quad (20)$$

$$VDFS=VDFM-VDFW \quad . \quad (21)$$

And finally, the initial mass of the steam is computed from,

$$ZQDS\#=VDFS/VDS \quad . \quad (22)$$

Dynamic Section.

In the dynamic part of the subroutine, the differential Eqs. (8), (9), (10) and (11) are integrated. It is assumed that some flashing of the water takes place when the steam enthalpy in the steam drum is greater than HDSS. This process is assumed to be linear and is described by,

$$FFL=(ZHSTX\#-HDSS)/TDFL \quad . \quad (23)$$

Again as in the static part, the volumes of water and steam in the steam drum are now computed by,

$$\text{CALL WASHP1}(ZHDX\#,ZPDF\#,ZTDRX\#,VDW,...) \quad , \quad (24)$$

$$VDFW=ZQDW\#*VDW \quad , \quad (25)$$

$$VDFS=VDFM-VDFW \quad . \quad (26)$$

The specific volume of the steam is then obtained from,

$$VDS1=VDFS/ZQDS\# \quad , \quad (27)$$

from which the pressure in the steam drum is computed by,

$$\text{CALL WASHV1}(ZHSTX\#,VDS1,ZTEDR\#01,VDS,F,...) \quad , \quad (28)$$

$$ZPDF\#01=-F2 \quad . \quad (29)$$

Upon exit from this subroutine, the exit steam and water density is computed and an energy balance is performed.

(8) APPLICABILITY AND RESTRICTIONS.

The steam drum has only two nodes; one contains all of the water and the other contains all of the steam in the module. The steam drum can be made part of any level-one recirculating steam generator model and should be appropriately connected to a boiler, to a feedwater supply unit, and to either a superheater or directly to the turbine. The user can include several steam drums into a system simulation by using the following definition.

• DFDRUMa1(VDFM,Lexp,TDFL,PAR1,PAR2,PAR3): ,

where a stands for a single character name given to the steam drum, and Lexp is the algebraic expression used to calculate the water level in the steam drum and can make use of any COMMON or local variables, or any of PAR1, PAR2, PAR3, given in the definition. To use the module in the simulation, the following statement should be included.

• DRUMa1;

The macro for this module can be destroyed when it is not needed any more.

```
*****
* DUMP11 *   (KFL,ZBLOCK)
*****
```

(1) PURPOSE.

This subroutine will output a common block of variables in DSNP and can be used subsequently as an input data set. The second entry to the subroutine is DUMPF. At this entry, data set KFL1 will be copied into data set KFL2.

(2) PARAMETERS.

KFL Data set into which the common block ZBLOCK is to be dumped.

KFL1 Data set to be copied into data set KFL2.

KFL2 Data set into which KFL is copied.

ZBLOCK Name of the common block to be written in DSNP data format. If ZBLOCK='ALL', all of the common blocks are dumped.

(3) COMMON VARIABLES.

All of the common blocks are inserted into this module. See Appendix A for the list of variables.

(4) LOCAL VARIABLES.

KALL 'ALL'

KFL1 If the output data set is KFL=6, then KFL1=6. Otherwise KFL1=ISTORD is the number of the temporary output data set.

KFZ Counter of the number of lines printed on a page.

KFZ1 Counter of the number of pages printed.

KICA Total number of the named common blocks present in DSNP.

KRBLCK Contains the name of the data block to be dumped and is an integer equivalent to

ZZBLCK.

KZ Set to NSP if KFL=IO6. Otherwise set to NPR.

NFR '.'

NSP ' '

ZABLK(KICA) A vector containing the names of DSNP common blocks.

ZADAT(10) A vector used to input or output 80 byte records.

ZALL 'ALL'

ZZBLCK Set to ZBLOCK an equivalent to KKBICK.

(5) OTHER SUBROUTINES OR MODULES REQUIRED.

Common Blocks: ALL. An appropriate statement that inserts all of the common blocks is produced by the precompiler.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

This module is used to generate the DUMPB1 subroutine in the FCRTAN simulation program. To find out which block is to be dumped, the parameter ZBLOCK is compared to all of the block names in the ZABLK vector. A computed GOTO is then used to transfer control to the appropriate output statement.

(7) ORGANIZATION OF MODULE.

The output produced by this subroutine is dependent upon the output data set. If the output is directed to the standard output or where KFL is equal to six, each record is printed on a new line. Hence, the first character output in the record is set to a blank. If, however, a data set is being generated, the output is first sent to a temporary data set ISTORD. This data set is then copied to KFL and a period is placed in the first column of each record. If the standard output data set is used, a title will also be printed containing the information on the present simulation time.

The DUMPF entry causes a data set KFI1 to be copied into KFL2. At this entry, the copying process is also dependent

upon the output data set. If the output is data set six, then it will be printed with a subtitle on each page and 60 lines per page. This entry is used as a result of the "IISTD KFL1 TO KFL2" DSNP statement.

(8) APPLICABILITY AND RESTRICTIONS.

The DUMPF1 subroutine is mainly used to create data sets at a given time in the simulation process. These data sets can then be used by subsequent simulations. It is also used for debugging purposes, since it is possible to print the values of any common block at any location in the program with a simple "DUMP A TO B" DSNP statement.

The DUMPF entry is used to copy one data set into another. Usually the output data set will be six and the listing of the data set is for documentation purposes. For example, "IISTD IFORT TO IO6" will produce a listing of the FORTRAN simulation program. Note that KFL1 and KFL2 will be rewound before the copying operation, except when the output data set KFL2 is equal to IO6.

To use this module in the simulation, the following statement should be included to output a single block of data.

- DUMP block⁵ TO if1;

To output all of the simulation data blocks, the following statement should be included.

- DUMP ALL TO if1;

To copy a data set into another data set the following statement should be included.

- DUMPF if1 TO if2;

if1, if1 and if2 are the FORTRAN characters assigned to the appropriate data sets.

```

*****
* EULER1 * (LCOP)
*****

```

(1) PURPOSE.

This subroutine performs rectangular Euler integration.

(2) PARAMETERS.

LCOP Indicator of the loop on which this integration method will be used.

(3) COMMON VARIABLES.

<u>NAMES</u>	<u>COMMON</u>	<u>DESCRIPTION</u>
DELT	TIMER	Current integration time step, in s.
DYDT(I)	DERIV	Vector of the derivatives to be integrated, integrator input.
IYYYY(J)	DEPIV	Integration indicator. If not equal to LOOP, no integration will be performed.
NCDER	DEPIV	Number of integrations to be performed in the simulation.
TIME	TIMER	Current simulation time, in s.
YYYY(I)	DEPIV	Vector of the integrated variables, integrator output.

(4) LOCAL VARIABLES.

Ncne.

(5) OTHER SUBROUTINES OR MODULES REQUIRED.

Subroutines: DERIV1.
Common Blocks: DERIV5, TIMER5.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

This module performs a simple rectangular integration on the derivative vector DYDT according to Eq. (1).

$$YYYY(I) = YYYY(I) + DELT * DYDT(I) \quad (1)$$

The results of the integration are stored in the YYYY vector.

(7) ORGANIZATION OF MODULE.

Upon entering the subroutine, a check is made to see if there are any integrations in this simulation. If NODEP is equal to zero, the integration process is skipped and the DERIV1 subroutine is called before returning. There are NCDEP variables to be integrated but this subroutine will integrate only those derivatives for which IYYYY(I) equals the loop number in which this integration subroutine is used.

The last step in this module is to call the DERIV1 subroutine generated by the DSNP system. This subroutine controls the sequence of all the necessary computations leading to the evaluation of the derivatives for the integrated loop.

(8) APPLICABILITY AND RESTRICTIONS.

This is a very simple integration method. It can be used either with constant or variable time steps. It is the responsibility of the user, however, to control the error accumulation and the changes, if any, in the integration time step magnitude.

```

*****
* FDBEK1 *
*****

```

(1) PURPOSE.

This subroutine calculates the feedback reactivity effects.

(2) PARAMETERS.

None.

(3) COMMON VARIABLES.

<u>NAMES</u>	<u>COMBLK</u>	<u>DESCRIPTION</u>
JTIM	TIMER	Time step counter.
RKD	FDBEK	Doppler reactivity feedback.
RKDC	FDBEK	Coefficient of doppler reactivity.
RKE	FDBEK	Core expansion reactivity feedback.
RKEC	FDBEK	Coefficient of core expansion reactivity.
RKFB	FDBEK	Total feedback reactivity.
RKS	FDBEK	Sodium density reactivity feedback.
RKSC	FDBEK	Coefficient of sodium expansion reactivity.
RKV	FDBEK	Core void reactivity.
RKVC	FDBEK	Core void reactivity coefficient.
TECA	CORTP	Core coolant average temperature, in C.
TECA0	CORTP	Initial TECA, in C.
TEFA	CORTP	Core fuel average temperature, in C.
TEFA0	CORTP	Initial TEFA, in C.
ZVOLFC	CORTP	Core coolant void volume fraction.

(4) LCCAL VARIABLES.

None.

(5) OTHER SUBROUTINES OR MODULES REQUIRED.

Common Blocks: CONST5, COFTP5, FDBEK5, NEUTR5, TIMER5.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

This module uses average core temperatures to calculate the reactivity feedback. The Doppler reactivity is calculated by,

$$RKD = RKDC * DLOG(TEFA/TEFA0) \quad . \quad (1)$$

The feedback reactivity effect due to core expansion is calculated by,

$$RKE = RKEC * (TEFA - TEFA0) \quad . \quad (2)$$

The feedback reactivity due to sodium expansion or "the sodium effect" is calculated by,

$$RKS = RKSC * (TECA - TECA0) \quad . \quad (3)$$

The feedback reactivity due to voiding or partial voiding of the core caused by boiling is calculated by,

$$RKV = RKVC * ZVOLFC \quad . \quad (4)$$

The appropriate reactivity coefficients, RKDC, RKEC, and RKSC for the core must be given as input data. The total feedback reactivity is then calculated by summing Eqs. (1), (2), (3), and (4) as,

$$RKFB = PKS + RKD + RKE + RKV \quad . \quad (5)$$

(7) ORGANIZATION OF MODULE.

Initially, the total feedback reactivity RKFB is assumed to be zero since the reactor is at steady state. Upon first entering the subroutine (JTIM=0), this variable is set to zero. On any subsequent entry, the contribution of each reactivity effect will be calculated by using Eqs. (1) to (5).

(8) APPLICABILITY AND RESTRICTIONS.

This module is a simplified model for calculating the reactivity feedback effects given the core average temperatures. It is usually used with the lumped parameter core model. The appropriate reactivity coefficients must be precalculated by other methods and given as input data. To use this module in the simulation, the following statement should be included.

- FDBEK1;

```

*****
* FDWTR1 *   (LCOP)
*****

```

(1) PURPOSE.

This module is a macro that produces one or more subroutines FDWTR1 that will adjust a flow so as to maintain a constant liquid level in a steam drum using a proportional plus integral controller.

(2) PARAMETERS.

LCOP An indicator of the loop from which this module is called.

(3) COMMON VARIABLES.

<u>NAMES</u>	<u>COMMON</u>	<u>DESCRIPTION</u>
ZFDCN#	FDWT#	The feedwater controller integration part.
ZFDW#	DRUM#	Feedwater inlet flow, in kg/s.
ZXDL#	DRUM#	Initial water level in the steam drum, in m.

(4) LOCAL VARIABLES.

CKP Controller proportional gain constant.

EPS Controller error function XDL-ZXDL#.

FDW Initial feedwater flow required by the steam drum, in kg/s.

KF Indicates the first entry and initially is set to .FAISF. .

TAU Controller integration time constant.

XDL Initial water level in the steam drum, in m.

(5) OTHER SUBROUTINES OR MODULES REQUIRED.

Common Blocks: CONST5, DERIV5, DRUM#5, FDWT#5, TIMER5.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

A proportional plus integral controller is used to maintain a constant water level in the steam drum. The deviation from the constant level is computed by,

$$EPS = XDI - ZXDia \quad , \quad (1)$$

and the feedwater flow to the steam drum is computed from,

$$ZFDWa = FDW * (1 + CKP * EPS + INTEGRAL[EPS / TAU]) \quad . \quad (2)$$

(7) ORGANIZATION OF MODULE.

The initialization of the module takes place upon terminating the IC calculations. This includes setting the water level reference value XDI and the initial flow FDW to the values calculated in the IC program section. Each subsequent entry to the subroutine will skip the initialization section. Subsequent entries to the subroutine will calculate the error in water level and produce a corrected flow if such an error EPS is different than zero.

(8) APPLICABILITY AND RESTRICTIONS.

This module should be located outside and after the initial conditions iteration loops since it only requires one initial adjustment and has no effect on the other values during the IC calculations. To include this module in the simulation, the following definition should be made in the program declarations segment.

• DFDWTa1 (CKP,TAU,BL,BU) ;

"a" is a single character name given to the module and must be the same as the name given to the steam drum to which this module is connected. BL and BU are the lower and upper limits of the fluid level and are usually the bottom and the top of the steam drum. To use this module in the simulation, the following statement should be included.

- FDWTR1;

The macro for this module can be destroyed when no longer needed.

```

*****
* FLOW01 * (KF,ZPI,ZPX,KFITMX,LOOP)
*****

```

(1) PURPOSE.

This module contains macros, DSNP statements, subroutines, and functions that will calculate the flow through a path defined by DFLOWii. When translated by the precompiler, this module produces the,

SUBROUTINE HFIOW1(KF,ZPI,ZPX,KFITMX,ICOP) that calculates the flow in the KFth path;

FUNCTION DFIPF(KF,LOCP) that calculates the total pressure drop in a path; and

SUBROUTINE FLINT(KF,ZPI,ZPX,KFITMX,LOOP) that calculates the flow derivatives for a defined path.

(2) PARAMETERS.

KF Indicates the path number as defined in FLOW.

KFITMX Maximum number of static iterations allowed in the flow IC calculations.

ICOP Indicates the loop from which this module is called.

ZPI Pressure at the inlet to the flow path, in Pa.

ZPX Pressure at the exit of the flow path, in Pa.

(3) COMMON VARIABLES.

<u>NAMES</u>	<u>COMMON</u>	<u>DESCRIPTION</u>
JTIM	TIMER	Time step counter.
ZDPFL(I)	HFLOW	Total pressure drop across a flow path, in Pa.
ZFLOW(I)	HFLOW	Magnitude of the flow in the Ith flow path, in kg/s.
ZINERF(I)	HFLOW	Equivalent inertance used in Eq. (1), in FLINT1, in 1/m.

ZINERT(I) HPIPF Pipe inertance, in 1/m.

(4) LOCAL VARIABLES.

DFIDT Flow time derivative, in kg/sec².

EPS Error in the static iterations.

KFIT Iteration counter.

KFLI First entry logical indicator to HFLOW1.

KJUNK If greater than zero, indicates transfer to JJUNii entry.

(5) OTHER SUBROUTINES OR MODULES REQUIRED.

Functions: DPHPI1, HPUMP1.

Common Blocks: CONST5, HFLOW5, HPIPE5, HPUMP5, TIMEF5.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

The flow total inertance is computed by summing up all of the contributions from the pipe segments belonging to the Kfth path cr,

$$ZINERT(KF) = \text{SUM}(ZINERT(I)) \quad . \quad (1)$$

The actual flow values calculated during the static IC part and the flow derivatives calculated during the dynamic part are obtained by calling the FLINT1 subroutine. The FLINT1 subroutine calculates the time derivative of the flow by using the following equation.

$$DFIDT = (ZPI - ZPX + ZDPFL(KF)) / ZINERT(KF) \quad (2)$$

For the static calculations, the error term is estimated to be,

$$FPS = DFIDT / ZFLOW(KF) \quad . \quad (3)$$

If this term is greater than ZZZER, the flow is changed according to,

$$ZFLOW(KF) = ZFLOW(KF) * (1 + EPS) \quad , \quad (4)$$

and the pressure drop ZDPFL(KF) is recalculated using the DELPF function. During the dynamic calculations, the flow is obtained by integrating DFIDT.

$$ZFLOW(KF) = INTEGRAL[DFLDT] \quad (5)$$

The pressure drop for a particular flow path or flow defined by DFLWII(WI) is calculated in DELPF. This is done by summing the static and frictional pressure drops for all of the pipe segments composing a flow path and adding the head produced by a pump, if present, in the flow path. The summation is done by,

$$DELPF1 = \text{SUM:DPHP1}(I, WI, TAV) + \text{SUM:HPUMP1}(I, WI, TPIN, KITMX, LOOP) \quad (6)$$

where DPHP1 and HPUMP1 are functions calculating the pressure drop in a pipe segment and the head generated in a pump, I indicates the Ith pipe or pump, WI is the flow rate, and TAV and TPIN are the average and inlet temperatures respectively. For details on the calculation of pressure drops and pressure heads, see the sections on HFIPE1 and HPUMP1.

(7) ORGANIZATION OF MODULE.

Upon first entry to HFLOW1, the inertance of all the hydraulic pipe segments are calculated by calling DPHP1 and then the flow inertances are evaluated by Eq. (1). If a hydraulic flow junction is present in the simulation, the equivalent junction inertance ZINJUN will be evaluated.

Upon entering the FLINT1 subroutine, the iteration counter KFIT is reset and the pressure drop in the flow path is evaluated by the function DELPF. The static iterations are continued using Eqs. (2) to (4) until EPS is less than ZZZEP or until KFIT is greater than KFITMX. If convergence is not achieved, a message to this effect is printed. During the dynamic calculations where JTIM is greater than zero, the iteration part is skipped and Eq. (5) is calculated. The number of statements in the DELPF1 function is dependent upon the number of flow definitions (see below). Three statements are generated by the precompiler macros for each flow definition. First, the defined flow WI is assigned the flow value from the flow vector ZFLCW(ii). Then a statement equivalent to Eq. (6) is produced and a GOTO 99 statement returns control to the calling subroutine. A computed GOTO statement uses the KF parameter to determine which set of statements will be executed.

(8) APPLICABILITY AND RESTRICTIONS.

The FLOW01 module is used to solve the hydraulic flow equations for single passage loops. In cases where multiple flow exists forming a flow junction, a JJUNii entry is generated to cope with the particular form of the junction. This subroutine is produced when a flow path is defined by,

• DFIOWii(WI)=(WO,PI,PX,KFITMX,PIPES(J1,I2,...),
PUMPS(J1,J2,...));

where ii is a two digit number assigned sequentially to the flow definitions, WO is the numerical value of the initial flow estimate, PI and PX are the inlet and outlet pressures of the defined flow path, and KFITMX is the maximum number of static iterations used to calculate the IC flow rate. The numbers J1,I2,... after PIPES indicate the pipe path through which the flow passes and the numbers J1,J2,... indicate the pump(s) present in the defined flow path. There is no limit on the number of pipes or pumps that can be present in a flow path.

The initial flow estimate WO should be as close to the steady state flow rate as possible or otherwise many static iterations will be required. In large systems, this convergence process may be very slow. To use a flow path calculation in the simulation, the following statement should be included.

• FLOWii;

If a flow path starts or terminates at a junction (see JUNC01), the above statement is not needed since the junction statement will take care of including the flow path calculations into the simulation.

```

*****
* FUNCT1 *   (X,V,KK,FX,FY,N,KLIST)
*****

```

(1) PURPOSE.

This module generates an arbitrary function using a KKth degree Lagrangian interpolating polynomial.

(2) PARAMETERS.

FX(I) A vector of abscissas of the arbitrary function. It must be given in increasing order and must have at least KK+1 consecutive numbers.

FY(I) A vector containing the values of the arbitrary function corresponding to the abscissas given in FX.

KK Order of the interpolation requested in this evaluation. ($0 < KK < 10$)

KLIST Location of the lowest function value used in the last interpolation or the search will start from FX(KLIST). Upon exit from the function, this variable is set to IX.

N Number of function values supplied in vectors FX and FY. ($N > KK$)

X Independent variable for which the value of the function is sought.

V Interpolated value of the arbitrary function for the arbitrary variable X. This is the output value of this module.

(3) COMMON VARIABLES.

None.

(4) LOCAL VARIABLES.

CI(I) K+1 coefficients of the interpolating polynomial can be either supplied to or calculated by the interpolating subroutine.

IX First location of the K+1 elements from the FX and

FY vectors used in the generation of the interpolating polynomial. ($0 < IX < N$)

K Degree of interpolation actually used in calling the Lagrange interpolating subroutine.

NN Last location for which a function is available. Usually $NN=N$, except when the function is not fully loaded.

(5) OTHER SUBROUTINES OR MODULES REQUIRED.

Subroutines: IAGPN1.
Common Blocks: CONST5.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

This function generator uses the Lagrange interpolating polynomial to compute the value of the function Y at X. The detailed algorithm used is described in module IAGPN1.

(7) ORGANIZATION OF MODULE.

The arbitrary function generator first performs a test to check whether the requested level of interpolation is within the predefined limits of zero less than K less than ten. If K is less than one, the value is set to one. If K is greater than nine, the value is set to nine. The range of the independent variable X is then tested. If X is less than $FX(1)$, a warning is issued. The interpolating level K is set to two and a first degree extrapolation using $FX(1)$ and $FX(2)$ will be calculated. If X is greater than $FX(N)$, a warning is issued. Sometimes the function is not fully loaded as when the last locations are empty. Then a search will be performed, if necessary, to find the last meaningful value of $F(X)$. The interpolating level K is set to two and a first degree extrapolation using $X(N-1)$ and $X(N)$ will be calculated.

The location J of the first value where $FX(I)$ is greater than X is then found. The search will start at the KLST value of X and after IX is found, KLST is reset to the last IX value. The location of the first IX value used by the interpolating subroutine is calculated from,

$$IX = J - K/2 - 1 \quad . \quad (1)$$

Equation (1) places the interpolation values of $FX(I)$ symmetrically around X. If IX is less than one, the

interpolation level is reduced to K plus IX and IX is set to one. If IX plus K is greater than N, then the interpolation level is reduced to K equal to N-IX. Finally, the value of the arbitrary function Y at X is calculated by the Lagrangian interpolation using the subroutine IAGRN1.

(8) APPLICABILITY AND RESTRICTIONS.

This module is an arbitrary function generator. It will calculate a value of any function Y at X if appropriate abscissas FX and ordinates FY are supplied for any number of points N greater than one. Any spacing between the abscissas is acceptable but the values must be stored in vector FX in increasing order. To use this module in the simulation, the following statement should be included.

```
CALL FUNCT1(X,Y,KK,FX,FY,N,KLST)
```

The function value Y at X is evaluated by using a KKth order Lagrangian interpolating polynomial symmetrically spaced around X. The degree of the polynomial is limited to KK less than ten. For additional details on arbitrary function generation in DSNP, see paragraph 2.4.2 in the DSNP users manual, ANI-CT-77-20.

```

*****
* GAMAR1 * (LCOP)
*****

```

(1) PURPOSE.

This subroutine calculates the power generated by radioactive decay.

(2) PARAMETERS.

LCOP Indicator to the loop from which this subroutine is called.

(3) COMMON VARIABLES.

<u>NAMES</u>	<u>COMBIX</u>	<u>DESCRIPTION</u>
BGM(I)	GAMAR	Fraction of the fission product produced by the Ith group.
GAM(I)	GAMAR	Disintegration constant of the Ith fission product.
IGAM	GAMAR	Number of radioactive FP decay groups assumed in the model.
JTIM	TIMER	Time step counter.
PGMA(I)	GAMAR	Fraction of the decay heat produced by the Ith FP group.
PNJ	NEUTR	Normalized neutronic power at the Jth time step.
WGN	GAMAR	Total normalized radiation power produced by the fission products.
WGP	GAMAR	Power produced by the decay of fission products, in W.
WGPO	GAMAR	Initial WGP, in W.
WGT	GAMAR	Power produced by the decay of fission products, in W.
WGTO	GAMAR	Initial WGT, in W.

(4) LOCAL VARIABLES.

None.

(5) OTHER SUBROUTINES OR MODULES REQUIRED.

Common Blocks: CONST5, DERIV5, GAMAR5, NEUTR5, TIMER5, TPOWR5.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

It is assumed that the heat produced by the fission products (FP) is due to the decay and gamma radiation of up to five FP elements. The rate by which the FP are produced in the core is directly proportional to the reactor power level. A lumped parameter model is assumed and the exponential decay equations given below are used.

$$PGMA(I) = \text{INTGR}[BGM(I) * PNJ - PGMA(I)] * GAM(I) \quad . \quad (1)$$

The total decay radiation level is given by,

$$WGN = \text{SUM}[PGMA] \quad . \quad (2)$$

To obtain the actual power produced by the FP decay, the following expressions are used.

$$WGP = WGN * WGP0 \quad (3)$$

$$WGT = WGN * WGT0 \quad (4)$$

(7) ORGANIZATION OF MODULE.

This subroutine calculates the normalized power level WGN which is set to one upon first entry to the subroutine. During the IC calculations, the initial power WGP0 in MW is provided as input from which WGT0 is then obtained. WGT0 and WGP0 are subsequently used to transform the normalized decay heat WGN to the appropriate power in watts by using Eqs. (3) and (4).

(8) APPLICABILITY AND RESTRICTIONS.

This module is a simple lumped parameter model to calculate the power contributed by the radioactive decay of FP accumulated in the reactor fuel. The steady state contribution WGP0 should be determined by the degree of

burn-up of the particular core and supplied by the user as input. The default value is WGPO equal to zero or zero burn-up.

To use this module in the simulation, the following statement should be included.

- GAMAR1;

```

*****
* GAUSS1 *   (N,M,A,D)
*****

```

(1) PURPOSE.

This subroutine solves a system of N coupled algebraic equations using the Gauss-Jordan elimination method. Optionally the transverse matrix and the determinant of the coefficient matrix can be computed.

(2) PARAMETERS.

A The coefficient matrix (N,M). The solution is always in $A(I,N+1)$, $I=1\dots N$.

D The determinant of the coefficient matrix.

M For ordinary solution, set M equal to $N+1$. To get the transverse matrix, set M equal to $2N+1$.

N The number of equations to be solved.

(3) COMMON VARIABLES.

None.

(4) LOCAL VARIABLES.

I Row index.

J Column index.

K Pivot element index.

(5) OTHER SUBROUTINES OR MODULES REQUIRED.

None.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

The Gauss-Jordan elimination algorithm is given by,

NORMALIZATION: $A(I,K) = A(K,J) / A(K,K) ; J=M,M-1,\dots,K$ (1)

REDUCTION: $A(I,J) = A(I,J) - A(I,K) * A(K,J) ;$
for $I=1\dots N, J=M,M-1,\dots,K$ (2)

and K in Eqs. (1) and (2) goes from $1\dots N$.

(7) ORGANIZATION OF MODULE.

Prior to calculating the above algorithm, the pivot element is tested. If it is less than ZZZER, an error message is printed. Next, the K th row is normalized by Eq. (1). Finally, the the row element above and below the pivot is eliminated by Eq. (2).

(8) APPLICABILITY AND RESTRICTIONS.

The efficiency and accuracy of the method can be maintained to about fifty equations. The subroutine will not solve the equations if any of the pivot elements becomes less than ZZZER. If necessary, the coefficient matrix should be rearranged so that there will not be zero elements on the diagonal part.

It is possible to obtain the inverse matrix by this subroutine. If desired, set M equal to $2N+1$ rather than M equal to $N+1$ and set the new matrix $A' = (A|U|I)$ where I is the unit matrix and U the free constant in any of the equations. Upon terminating the solution, I will be replaced by the inverse of A . To use this module in the simulation, the following statement should be included.

CALL GAUSS1(N,M,A)

REFERENCES.

- [1] B. Carnham, H. A. Luther, J. O. Wilkes, "Applied Numerical Methods", John Wiley, (1969).

```

*****
* HPIEE1 *   (KPH,ZW,ZTAV,PRAV,X,VALV)
*****

```

(1) PURPOSE.

This module contains macros and DSNP statements producing a function that calculates the pressure drop in a hydraulic pipe segment defined by DEFHPIPEii.

(2) PARAMETERS.

KPH The pipe path as defined by "ii" in the DEFHPIPEii definition. ii must be two digits.

PRAV The average or entry pressure for the pipe segment, in Pa.

VALV The pipe resistance expressed in equivalent length, in m.

X The steam quality for two phase flow. Set X less than or equal to zero for single phase flow.

ZTAV The average temperature of the fluid in the pipe, in C.

ZW The flow rate through the pipe, in kg/s.

(3) COMMON VARIABLES.

NAMES	COMBLK	DESCRIPTION
ZDPH(I)	HPIPE	Frictional and static pressure drop across a pipe, in Pa.
ZFALF(I)	HPIPE	A constant in the pipe friction factor calculation. Use 1 for sodium, 2 for water, or 3 for steam.
ZFRK(I)	HPIPE	A constant in the pipe friction factor calculation. Use 1 for sodium, 2 for water, or 3 for steam.
ZINEFT(I)	HPIPE	Pipe inertance, in 1/m.
ZQHPIP(I)	HPIPE	Mass of fluid in the pipe, in kg.

(4) LOCAL VARIABLES.

DPHPI1 The pressure drop across the pipe, in Pa.

KEHMAX The total number of hydraulic pipe segment defined in the simulation.

KPIP The initial DPHPI1 function entry indicator.

KMAT(I) A material indicator for the fluid flowing in the pipe. SO, WA, or ST are used for sodium, water, or steam respectively.

ZAFL(I) The Ith pipe flow cross sectional area, in m².

ZHIXG(I) The Ith pipe elevation difference multiplied by the gravity constant, in (m/s)²

ZHPRES(I) The friction factor constant for the Ith pipe, in 1/m².

ZMAT(K) Same as KMAT but stored in a double precision word.

ZMUHP(I) The Ith pipe fluid dynamic viscosity, in Pa-s.

ZFHP(I) The Ith pipe fluid average density, in kg/m³.

ZTHA(I) The Ith pipe fluid average temperature, in C.

ZWSIGN The sign of ZW multiplied by one.

ZXDE(I) The Ith pipe hydraulic diameter, in m.

ZXHI(I) The Ith pipe inlet elevation, in m.

ZXHX(I) The Ith pipe exit elevation, in m.

ZXLE(I) Pipe equivalent length, in m.

ZXLP(I) The Ith pipe length, in m.

(5) OTHER SUBROUTINES OR MODULES REQUIRED.

Subroutines: DENST1, VISO1.
Common Blocks: CONST5, HPIPE5, TIMER5.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

This function created by DSNP will calculate, in addition to the static and friction pressure drop across the pipe,

the pipe inertance ZINERT and the weight of the fluid ZQHPIP in the pipe. The first is then used in the momentum balance equations to calculate the flow through a path. The second is used to calculate the changes in the fluid level in a cavity to which the appropriate pipe is connected. Upon first entry to the pipe function, all of the pipe inertance and elevation constants are computed by,

$$ZINERT(K) = ZXLP(K) / ZAFI(K) \quad . \quad (1)$$

$$ZHIXG(K) = 9.81 * (ZXHI(K) - ZXHX(K)) \quad . \quad (2)$$

Then the friction coefficient for a particular pipe and material is calculated from,

$$ZHPRES(K) = 2 * ZFRK(KMAT(K)) * ZXLE(K) / (ZAFI(K) * (2 - ZFAIF(KMAT(K))) * ZXDE(K) * (1 + ZFAIF(KMAT(K)))) \quad . \quad (3)$$

Next, the material properties of the fluid in the pipe are evaluated but only if a change in the temperature or pressure are detected. The DENST1 subroutine and VISCO1 subroutines are used to obtain the fluid density and dynamic viscosity. The fluid weight in the pipe is then calculated from,

$$ZQHPIP(KPH) = ZPHF(KPH) * ZXLP(KPH) * ZAFI(KPH) \quad . \quad (4)$$

The pressure drop across the pipe is obtained from,

$$ZDPH(KPH) = ZHIXG(KPH) * ZRHP(KPH) - ZHPRES(KPH) * ZMUHP(KPH) / ZRHP(KPH) * ZWSIGN * DABS(ZW) * (1 - ZFAIF(KMAT(KPH))) \quad . \quad (5)$$

(7) ORGANIZATION OF MODULE.

Upon first entry to the function, the pipe constants for all of the system pipes are calculated using Eqs. (1) to (3). For each subsequent entry, this part is skipped and comparison between the previous and most recent fluid average temperatures is made. If the difference is greater than 1 C., the fluid density and dynamic viscosity are reevaluated using the appropriate material property functions. Finally, Eqs. (4) and (5) are evaluated producing the pressure drop (or gain) in the pipe segment.

(8) APPLICABILITY AND RESTRICTIONS.

A hydraulic pipe segment is defined by the following DSNP statement.

• DEFHPIPIi(7W,TIN,TEX,ZXHI,ZXHX,ZXLP,ZXLE,ZXDE,ZAFI,

ZMAT,PRAV,COMBL);

ii is a two digit number assigned to pipe segment definitions in a sequential order starting with 01. TIN and TEX are the pipe section inlet and outlet temperatures. PRAV is either the inlet or average pressure of the pipe segment and COMBI is the appropriate COMMON data block in which TIN, TEX and PRAV are located. If several hydraulic pipe segments are used in a model, then a particular COMBI should be used only once in all of the pipe and pump definitions. An "*" should be placed in those definitions where no COMBI is desired. If more COMMON blocks are required than pipe definitions used, a dummy definition has to be made. All of the other variables in the definition are given above. Numerical values for ZXHI, ZXHX, ZXLP, ZXLF, ZXDE and ZEFL must be provided. This pipe module will not at the present time compute two-phase pressure drop.

```

*****
* HPUMP1 *   (K7,ZFLC,ZEPIN,ZTPIN,KITMX,LCOP)
*****

```

(1) PURPOSE.

This module contains macros and DSNP statements producing a function that calculates the pressure head produced by a pump defined by DFPUMPii.

(2) PARAMETERS.

KITMX The maximum number of static iterations allowed while evaluating the pump speed.

K7 The identification number of the pump under evaluation.

LCOP The indication number to the loop from which this module is called.

ZFIO The flow through the pump, in kg/s.

ZEPIN The pump inlet pressure, in Pa.

ZTPIN The fluid inlet temperature, in C.

(3) COMMON VARIABLES.

<u>NAMES</u>	<u>COMBIK</u>	<u>DESCRIPTION</u>
JTIM	TIMER	Time step counter.
ZFLOP(I)	HPUMP	Rated flow through the pump, in kg/s.
ZHRPU(I)	HPUMP	Rated head generated by the pump at rated flow, in m.
ZHRZF(I)	HPUMP	Rated head produced by the pump at zero flow and rated speed, in m.
ZIPUMP(I)	HPUMP	Pump inertia, in kg/s.
ZFUMAT(I)	HPUMP	Material indicator of the fluid flowing through the pump.
ZPU1(I)	HPUMP	Constant in the pump efficiency calculation.

ZPU2(I)	HPUMP	Constant in the pump efficiency calculation.
ZREV(I)	HPUMP	Speed of the pump, in radians/s.
ZREVR(I)	HPUMP	Rated speed of the pump, in radians/s.
ZROPU(I)	HPUMP	Fluid inlet density to the pump, in kg/m ³ .
ZTEPIN(I)	HPUMP	Pump fluid inlet temperature, in C.
ZTORP(I)	HPUMP	Pump torque, in N-m.
ZTORPR(I)	HPUMP	Rated pump torque, N-m.

(4) LOCAL VARIABLES.

EPS The error in computing the pump speed during static iterations.

ETA The pump efficiency.

KITE Static iteration counter.

ZAEPS The absolute value of EPS.

ZFLON The ratio of flows ZFLO/ZFLOR.

ZFP The ratio of flows ZFLON/ZFEVN.

ZFEVN The ratio of pump speeds ZREV/ZREVR.

ZTORD The torque delivered to the pump by the motor, in N-m.

(5) OTHER SUBROUTINES OF MODULES REQUIRED.

Subroutines: MOTOR1, SCDFN1, WADEN1.
 Common Blocks: CONST5, DERIV5, HPUMP5, TIMER5 and other common blocks containing data required by this function.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

The contribution of a pump to the hydraulic loop is a positive head generated by the revolving rotor. The approximate relation for the generated head for the KZth

pump is given by,

$$\begin{aligned} \text{HPUMP1} &= \text{ZHRZF}(KZ) * \text{ZFEVN}^{**2} + (\text{ZHRPU}(KZ) - \text{ZHRZF}(KZ)) \\ &\quad * \text{ZFLO}^{**1.8} \end{aligned} \quad (1)$$

The head HPUMP1 is calculated in meters and is later converted to Pascals by,

$$\text{HPUMP1} = \text{HPUMP1} * 9.81 * \text{ZROPU}(KZ) \quad (2)$$

The pump efficiency under the prevailing flow conditions is given approximately by,

$$\text{ETA} = \text{ZPU1}(KZ) * \text{ZFR} - \text{ZPU2} * \text{ZFF}^{**2} \quad (3)$$

The pump torque is given by,

$$\text{ZTCFP}(KZ) = \text{HPUMP1} * 9.81 * \text{ZFLO} / (\text{ZREV}(KZ) * \text{ETA}) \quad (4)$$

The relative error term used during the static iterations is given by the difference between the torque delivered by the motor and the required pump torque.

$$\text{EPS} = (\text{ZTCRD} - \text{ZTORP}(KZ)) / \text{ZTORD} \quad (5)$$

For the dynamic integration process, the pump speed is obtained by,

$$\text{ZREV}(KZ) = \text{INTEGRAL}((\text{ZTCRD} - \text{ZTORP}(KZ)) / \text{ZIPUMP}(KZ)) \quad (6)$$

(7) ORGANIZATION OF MODULE.

Upon entering the function, the normalized flow rate and pump speed are computed as,

$$\text{ZFICN} = \text{ZFLO} / \text{ZFICR}(KZ) \quad (7)$$

$$\text{ZFEVN} = \text{ZREV}(KZ) / \text{ZREVR}(KZ) \quad (8)$$

The ratio of the normalized variables is given by,

$$\text{ZFR} = \text{ZFICN} / \text{ZFEVN} \quad (9)$$

These normalized variables are then used in Eqs. (1) to (6). The torque delivered to the pump ZTORD is computed by the MOTCR1 subroutine.

Equations (1) to (4) are solved during each entry. During the static calculations, the pump speed can be changed so as to reduce FPS below ZZZER (0.0001). If, however, the motor algorithm delivers a torque equal to the demanded pump torque, no iterations will be performed and the pump

speed will remain constant as defined by the user.

(8) APPLICABILITY AND RESTRICTIONS.

This is a generalized pump function and can be defined by,

- DFPUMFii (ZHFPU,ZHRZF,ZREVR,ZPPIN,ZTORPR,ZFLO,ZFLOR,
ZIPUMP,ZPU1,ZPU2,ZREV,ZTEPIN,ZPUMAT,KITMX,COMBLOCK);

The characteristics of the iith pump are defined by the macro parameters in the above definition. The COMBLOCK is the name of the data block containing the fluid temperature and should be inserted into the definition only if not previously inserted in a DFPUMP or DFHPIP definition. Otherwise a dummy parameter "*" should be inserted.

If the designed pump head ZHRP" for the reference operating conditions does not match the required head in the hydraulic loop, many iterations will be necessary to achieve convergence. The user should match the defining parameters to the manufacturers data. Once a pump is defined, it can be used in the definition of any flow path to which it belongs.

```

*****
* IHXMS1 * (I.CCF)
*****

```

(1) PURPOSE.

This macro produces a subroutine SIHX#1 to solve the thermodynamic equations of the intermediate heat exchanger (IHx) with any number of axial nodes and three radial nodes.

(2) PARAMETERS.

IHX Number of IHX axial segments to be simulated.

NAME A single character that will be inserted into the subroutine name SIHX#1 to replace # so as to produce a unique name for the module.

NIW Number of parallel channels in the IHX.

ZEDI Pitch to diameter ratio in the IHX.

(3) COMMON VARIABLES.

<u>NAMES</u>	<u>COMMON</u>	<u>DESCRIPTION</u>
AF*P	IHXM#	Total IHX primary cross sectional flow area (shell side), in m.
P*W	IHXM#	Thermal conductivity of the IHX tube, in W/m-C.
C*W	IHXM#	Specific heat of the IHX tube metal, in J/kg-C.
E*PSHA	IHXM#	Maximum absolute error allowed in the IHX temperature calculation.
E*PSHR	IHXM#	Maximum relative error allowed in the temperature calculation of IHX.
F*HI	IHXM#	Secondary IHX flow, in kg/s.
F*PI	IHXM#	Primary IHX flow, in kg/s.
ITI*MX	IHXM#	Maximum number of inner iterations in the IHX initial temperature calculation.

ITO*MX	IHXM#	Maximum number of external iterations in the IHX initial temperature calculation.
JTIM	TIMFR	Time step counter.
R*W	IHXM#	IHX tube metal density, in kg/m^3 .
TE#H(I)	IHXM#	IHX secondary side temperature distribution, in C.
TE#HI	IHXM#	IHX secondary coolant inlet temperature, in C.
TE#HX	IHXM#	IHX secondary coolant exit temperature, in C.
TF#P(I)	IHXM#	Primary IHX coolant temperature, in C.
TE#PI	IHXM#	IHX primary sodium inlet temperature, in C.
TF#PX	IHXM#	IHX primary exit temperature, in C.
TE#W(I)	IHXM#	IHX tube wall temperature distribution, in C.
X*RH	IHXM#	IHX tube inner radius, in m.
X*RP	IHXM#	IHX tube external radius, in m.
X*WL	IHXM#	IHX tube length, in m.
ZEGII*	IHXM#	Power transmitted into the heat exchanger primary, in W.
ZEGIX*	IHXM#	Power leaving the heat exchanger secondary side, in W.
ZPH#A	IHXM#	Heat exchanger secondary side average pressure, in Pa.
ZPP#B	IHXM#	Primary side average pressure, in Pa. Important only when water is the working fluid.
ZROHX*	IHXM#	Heat exchanger secondary exit density, in kg/m^3 .
ZROPX*	IHXM#	Heat exchanger primary side fluid exit density, in kg/m^3 .
Z#PDI	IHM#	Not used.

(4) LOCAL VARIABLES.

ARSH Secondary node heat transfer area, in m^2 .

ARIP Primary node heat transfer area, in m .

BR Secondary node coolant conductivity, in $W/m-C$.

BP Primary node coolant conductivity, in $W/m-C$.

CH Secondary coolant specific heat, in $J/kg-C$.

CP Primary coolant specific heat, in $J/kg-C$.

DTEH(I) Time derivative of TEH(I).

DTEPH Nodal temperature increment for initial distribution estimate.

DTEP(I) Time derivative of TEP(I).

DTEW(I) Time derivative of TEW(I).

EPESH Sum of the absolute and relative error ratios channel average or exit temperature.

EPSP Sum of the absolute and relative error ratios in the primary channel.

FH Average flow per secondary channel, in kg/s .

FBM Secondary flow in channel divided by secondary node mass, in $1/s$.

FP Average flow per primary channel, in kg/s .

FPM Primary flow in channel divided by primary node mass, in $1/s$.

ITI Internal iterations counter.

ITO External iteration counter.

OW Weight of tube in one node, in kg .

RH Secondary coolant average density, in kg/m^3 .

RP#X Secondary side exit density, in kg/m^3 .

SH Secondary coolant average dynamic viscosity, in $kg/s-m$.

SP Primary coolant average dynamic viscosity, in $kg/s-m$.

TEHA Secondary coolant average temperature, or average node temperature, in C.

TEHA1 Last TPFA at which the material properties were evaluated, in C.

TEHH Secondary coolant node inlet temperature, in C.

TEPA Primary coolant average temperature or average node temperature, in C.

TEPA1 Last TEPA at which material properties were evaluated, in C.

TEPP Primary coolant node inlet temperature, in C.

TH Secondary heat transfer coefficient, in W/m^2-C . Also total heat transfer per node, in W/C .

UHM Secondary node heat transfer divided by node heat capacity, in $1/s$.

UH1 Coefficient $(FH*CH-UH/2)/(FH*CH+UH/2)$.

UH2 Coefficient $UH/(FH*CH+UH/2)$.

UP Primary heat transfer coefficient, in $J/C-m^2-s$. Also total heat transfer per node, in $J/C-s$.

UPM Primary node heat transfer divided by node heat capacity, in $1/s$.

UP1 Coefficient $(FP*CP-UP)/2$.

UP2 Coefficient $UP/(FP*CP+UP/2)$.

UWH Secondary node heat transfer divided by node tube heat capacity, in $1/s$.

UWP Primary node heat transfer divided by node tube heat capacity, in $1/s$.

UW1 Coefficient $0.5*UP/(UP+UH)$.

UW2 Coefficient $0.5*UH/(UP+UH)$.

VH Volume of secondary channel node, in m^3 .

VP Volume of primary channel node, in m^3 .

XDH Secondary channel (inside tube) hydraulic diameter, in m.

XDP Primary channel (outside tube) hydraulic diameter, in m.

The heat balance equations solved for each of the above nodes are,

$$CP*FP*VP(T)*DTEP(I) = FP*CP*(TEP(I-1) - TEP(I)) - UP*(TEPA - TEW(J)) \quad , \quad (1)$$

$$CW*QW*DTFW(I) = UP*(TEPA - TEW(I)) - UH*(TEW(I) - TEHA) \quad , \quad (2)$$

$$CH*RH*VH(I)*DTEH(I) = FH*CH*(TEH(I+1) - TEH(I)) - UH*(TEHA - TEW(I)) \quad . \quad (3)$$

The heat transfer correlations are given in the appropriate `UIHXSP1` and `UIHXS1` modules. The average primary and secondary coolant temperature is given by,

$$TEPA = (TEP(I-1) + TEP(I)) / 2 \text{ and } TEHA = (TEH(I) + TEH(I+1)) / 2 \quad . (4)$$

To calculate the initial temperature distribution, the derivatives in Eqs. (1) to (3) are set to zero and an implicit iterative technique is used.

(7) ORGANIZATION OF MODULE.

Upon first entry to the subroutine, several constants required by the model are computed.

The weight of the tube material in each node is given by,

$$OW = 3.14 * (HFP**2 - X*RH**2) * X*WL*R*W / IHX \quad . \quad (5)$$

The primary channel hydraulic diameter is given by,

$$XDP = (4*AR*F/NXW) / (2*3.14*X*RP) \quad . \quad (6)$$

The secondary channel hydraulic diameter is given by,

$$XDW=2*X*RH \quad . \quad (7)$$

The primary node volume is given by,

$$VP=AR*P*X*WL/(NXW*IHX) \quad . \quad (8)$$

The secondary node volume is given by,

$$VH=3.14*XAPH**2*XAWL/IHX \quad . \quad (9)$$

The primary node heat transfer area is given by,

$$ARIP=2*3.14*X*RP*X*WL/IHX \quad . \quad (10)$$

The secondary node heat transfer area is given by,

$$ARIH=2*3.14*X*RH*X*WL/IHX \quad . \quad (11)$$

The nodal temperature increment for the first initial distribution guess is given by,

$$DTEPH=(TE*FI-TE*HI-100)/IHX \quad . \quad (12)$$

The temperatures for each of the nodes are then estimated by,

$$TE*P(I)=TE*P(I-1)-DTEPH \quad , \quad (13)$$

$$TE*H(I)=TE*H(I-1)-DTEPH \quad , \quad (14)$$

and

$$TE*W(I)=(TE*P(I)+TE*H(I))/2 \quad . \quad (15)$$

For each entry, the average thermodynamic properties are calculated for the primary side using the primary average temperature,

$$TEPA=(TE*PI+TE*PX)/2 \quad , \quad (16)$$

and for the secondary side, using the secondary average temperature,

$$TEHA=(TE*HI+TE*HX)/2 \quad , \quad (17)$$

by calling the appropriate material property subroutines.

The average flow rate in the primary and secondary channels is obtained from,

$$FP=FP*I/NXW, \text{ and } FH=FH*I/NXW \quad . \quad (18)$$

The heat transfer coefficient UP for the primary side and

UH for the secondary side are then evaluated using the special subroutines UIHXP1 and UIHXS1 respectively.

For the static calculations, the coefficients of the static equations are then evaluated and the static distribution is calculated from a modified form of Eq. (1). The convergence criteria requires that in two consecutive iterations the combined relative and absolute errors in the primary and secondary exit temperatures must be less than one. This can be called the inner iteration. In addition, an external iteration is performed that requires the convergence of primary and secondary average temperatures. The thermodynamic properties are reevaluated for external iterations only.

If static convergence is achieved, the coolant exit density of the primary and secondary sides is calculated. If convergence in the initial static distribution is not achieved, a warning message is printed.

During the dynamic calculations when JTIM is greater than zero, the static part of the module is skipped. The coefficients of the dynamic equations are first computed and then Eqs. (1) to (3) are solved.

(8) APPLICABILITY AND RESTRICTIONS.

An IHXM#1 subroutine is produced by the system for each heat exchanger in the simulation defined by,

• IHXM#1(INX,NXW,ZPD1,MAT1,MAT2);

where MAT1 and MAT2 are the primary and secondary fluids used in the heat exchanger and can be either SO for sodium or WA for water. The # symbol in the above definition can be any character or number identifying the particular heat exchanger. Therefore up to 36 different heat exchangers can be defined in the system. This module can produce only counterflow, tube and shell, heat exchangers in which sodium, water, or steam can flow and no change of phase takes place. To use this module once it is defined, the following statement is included.

• IHXM#1;

```

*****
* JUNC01 * (ZPJUN,ZP1,ZP2,...ZPn,ZPJX,KFITMX,LOOP)
*****

```

(1) PURPOSE.

This module contains macros and DSNP statements producing an entry to subroutine HFLOW1 for each hydraulic junction defined by DFJUNCI(PJ). The statements produced by this module calculate the flow rate of paths entering or leaving the iith junction and the pressure at the junction.

(2) PARAMETERS.

KFITMX Maximum number of static iterations in computing the flows.

LOOP Indicator of the loop to which the particular junction belongs.

ZPJUN Pressure at the junction, in Pa.

ZPJX Pressure at the exit flow section connected to the junction, in Pa.

ZP1,ZP2,.. Pressures at the entrance to the flow path connected to the junction, in Pa.

(3) COMMON VARIABLES.

None.

(4) LOCAL VARIABLES.

EPS Error in computing junction exit flow.

KFIT Iterations counter.

ZFLOW2 Value of the flow leaving the junction as computed in the previous iteration, in kg/s.

ZINJUN(I) Equivalent inertance used for computing the pressure at the junction, in 1/m

(5) CITER SUBROUTINES OF MODULES REQUIRED.

Subroutines: FLOW01.

Common Blocks: Same as in FLOW01.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

The pressure at the Jth junction of the hydraulic network is given by,

$$ZPJUN = ZINJUN(J) * ((ZPJX - ZDPFL(ITEK)) / ZINERF(IEX) + SUM: (ZPin + ZDPFL(Iin)) / ZINERF(Iin)) : \quad , \quad (1)$$

where Iex and Iin are for the exit and inlet flows, and ZINJUN is given by,

$$ZINJUN(J) = 1 / (1 / ZINERF(Iex) + SUM: (1 / ZINERF(Iin))) : \quad , \quad (2)$$

and is computed upon the first entry into the module. The summations in Eqs. (1) and (2) are the over all incoming flows. With the pressure at the junction known, or estimated during static iterations, the flow in the incoming branches is computed by calling FLINT1 as in HFLOW1. The exit flow is obtained by summing all of the incoming flows by,

$$ZFLOW(Iex) = SUM: ZFLOW(Iin) : \quad . \quad (3)$$

(7) ORGANIZATION OF MODULE.

Upon entering JJUN, a check is made for first entry and, if required, the equivalent inertance ZINJUN of all of the junctions in the system are calculated using Eq. (2). The pressure drop or gain in each of the flow paths is then calculated by DELPF. One statement per flow path is produced by the precompiler. Next, Eq.(1) is solved and the flows, both static and dynamic, are calculated in FLINT1. For the static calculations, the convergence criteria requires that the relative difference between two consecutive iterations of the exit flow be less than 0.0001. KFITMX static iterations are allowed before control is returned to the main program.

(8) APPLICABILITY AND RESTRICTIONS.

An entry to the HFLOW1 module is produced for each junction of the system hydraulic network defined by,

* DFJUNCii(JPJUN) = (INFLO(I1,I2,...),EXPLO(Iex)

INPR(ZP1,ZP2,...),EXPR(7PJX),KFITMX) .

I1, I2, ... and Iex are the appropriate flow numbers as defined in DFLOW. The number of paths entering a junction is unlimited but only one path can leave the junction. In all of the hydraulic calculations, the flow is assumed to be incompressible. The number of junctions in the simulation has to be defined prior to the definition of a junction by,

- DFNJ(N); .

To use this module in the simulation, the following statement should be included.

- JUNCii;

```
*****
* LAGRNI *   (X,Y,XF,YF,CL,K)
*****
```

(1) PURPOSE.

This module calculates the Lagrange interpolation polynomial.

(2) PARAMETERS.

CL(I) Lagrange coefficients.

K Degree of the polynomial to be evaluated. K ranges from one to nine. If K is negative, the Lagrange coefficients evaluated at the previous step or supplied as parameters should be used.

X Independent variable for which the value of the interpolated function is sought.

XF(I) Abscissas of K+1 points of the function to be interpolated. The values must be given in increasing order.

Y Value of the interpolated polynomial function at X. This is the output value of this module.

YF(I) Ordinates of the function to be interpolated given at K+1 points of XF(I).

(3) COMMON VARIABLES.

None.

(4) LOCAL VARIABLES.

C Intermediate product of the Lagrange coefficient denominator $XF(I) - XF(J)$.

XL(I) Intermediate differences of the independent variable X or, $X - XF(I)$.

XIC Product of the K+1 differences XL(I).

(5) OTHER SUBROUTINES OR MODULES REQUIRED.

None.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

The value of the interpolating polynomial of degree K at the independent variable X is calculated from the well known Lagrangian interpolating polynomial [1] given by,

$$Y = \sum_{I=1}^{I=K+1} XIC * CL(I) / XL(I) \quad , \quad (1)$$

where $XL(I)$ are the differences of the independent variable given by,

$$XL(I) = X - XL(I) \quad , \quad (2)$$

and XIC is the product of differences given by,

$$XIC = \prod_{I=1}^{I=K+1} XL(I) \quad . \quad (3)$$

The Lagrange coefficients $CL(I)$ are given by,

$$CL(I) = YF(I) / \prod_{\substack{J=1 \\ J \neq I}}^{J=K+1} (XF(I) - XF(J)) \quad . \quad (4)$$

(7) ORGANIZATION OF MODULE.

Upon entering the subroutine, differences are calculated by using Eq. (2). If the relative difference is less than $10E-6$, or the independent variable X is "almost" equal to one of the given abscissas, the interpolating polynomial is not evaluated and the output value Y is set to the value of the function YF(I) at this point.

Next, the sign of K is tested. If it is positive, the Lagrange coefficients $CL(I)$ are calculated from Eq. (4). If K is negative, previously calculated or supplied values are used. Finally, the value of the interpolated function Y at X is calculated from the Kth degree Lagrange polynomial using Eq. (1).

(8) APPLICABILITY AND RESTRICTIONS.

The use of the Lagrangian interpolation scheme is limited to one less than or equal to K and less than ten. To use this module in the simulation, the following statement should be included.

CALL LAGRNI(X,Y,XF,YF,CI,X)

REFERENCES.

- [1] F. Henrici, "Elements of Numerical Analysis", John Wiley, (1963)

```

*****
* LOAZZ1 *   (X,Y,XX,YY,N,N1)
*****

```

(1) PURPOSE.

This subroutine loads the variables X and Y into vectors XX and YY. The loading location is determined by the numerical value of X so that the vector XX has the numbers stored in an increasing sequence.

(2) PARAMETERS.

N Number of storage locations in XX or YY.

N1 Pointer to the next empty location in XX. If the vector is fully loaded, then N1=N.

X Variable to be loaded in XX.

XX Storage vector of N locations.

Y Variable to be loaded in YY.

YY Storage vector of N locations.

(3) COMMON VARIABLES.

None.

(4) LOCAL VARIABLES.

K Counter of the shifting operations performed when loading a number X into the previously loaded vector.

(5) OTHER SUBROUTINES OR MODULES REQUIRED.

None.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

Loading a new X,Y pair into the vectors XX and YY respectively, can be either into the next empty location N1 if the new X is greater than the previously inserted values, or among the previously inserted values, if the new X is less than XX(N1-1). In the second case, X is compared to the XX(I) value and inserted in place of the first value for which XX(I) is greater than X. A shifting operation of all of the values between I and N1 then takes place. If N1 equals N, the last value of the vectors XX and YY will be lost.

(7) ORGANIZATION OF MODULE.

Upon entering the module for the first time for a particular function storage, the contents will be tested and, if empty or partially empty, the first free location pointer N1 will be set up. If any two consecutive XX(I) are equal, the locations between I and N are considered empty and N1 will equal I.

The value of X is then compared to the values in XX. X will replace the first value greater than or equal to X. If X replaces a larger value, all of the vector values XX(I) and YY(I) will be shifted one location or to XX(I+1) and YY(I+1) respectively. If N1 is less than N, the next free location pointer N1 is increased by one.

(8) APPLICABILITY AND RESTRICTIONS.

When a function is defined, the number of locations designated is fixed. If it is desired to load or change a function during the simulation, the values overflowing the function storage capacity will be lost. The numbers lost will always be those with the largest independent variable XX(I). To use this module in the simulation, the following statement should be included.

CALL LCAZZ1(X,Y,XX,YY,N,N1)

```

*****
* LPLEN1 *   (LCOP)
*****

```

(1) PURPOSE.

This module simulates the thermodynamics of the core inlet lower plenum.

(2) PARAMETERS.

LOOP Indicator to the loop from which this subroutine is called.

(3) COMMON VARIABLES.

<u>NAMES</u>	<u>COMMON</u>	<u>DESCRIPTION</u>
CC	COFTP	Primary coolant specific heat, in J/kg-C.
CM	LPLEN	Specific heat of the metal in the lower plenum, in J/kg-C.
OIPF	LPLEN	Weight of metal in the lower plenum, in kg.
ROCM	LPLEN	Coolant average density in the lower plenum, in kg/m ³ .
TEMA	LPLEN	Average coolant temperature in the lower plenum, in C.
TEMFA	LPLEN	Average metal temperature in the lower plenum, in C.
TEMI	LPLEN	Lower plenum inlet temperature, in C.
TEMX	LPLEN	Lower plenum exit temperature, in C.
UMA	LPLEN	Total heat transfer from metal to coolant in the lower plenum, in W/C.
VMC	LPLEN	Volume of coolant in the lower plenum, in m ³ .

(4) LOCAL VARIABLES.

CMC Total lower plenum heat capacity given by Eq. (4),
in J/C.

CMF Total lower plenum metal node heat capacity given by
Eq. (3), in J/C.

(5) OTHER SUBROUTINES OR MODULES REQUIRED.

Subroutines: SODEN1.

Common Blocks: CORTP5, DERIV5, IPLEN5, TIMER5.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

This is a simplified lumped parameter model of the core inlet or lower plenum. It is assumed to have a metal node that includes all of the metal parts in the lower plenum and a coolant node that includes the total volume of the lower plenum coolant.

The metal and coolant node temperatures are calculated from the appropriate heat balance equations. The coolant average temperature $TEMA$ is given by,

$$TEMA = \frac{INTEGRAL[CC*GC*(TEMI - TEMA) + UMA*(TEMFA - TEMA)]}{CMC} \quad (1)$$

and the metal node average temperature is given by,

$$TEMFA = \frac{INTEGRAL[UMA*(TEMA - TEMFA)]}{CMF} \quad (2)$$

It is assumed that ideal mixing prevails in the lower plenum and that the exit temperature is equal to the coolant average temperature.

(7) ORGANIZATION OF MODULE.

Upon the first entry to the module (JTIM=0), the initial conditions are calculated. Since there is no heat source in the lower plenum, all of the temperatures are initially assumed to be equal to the module inlet temperature $TEMI$. This value should be supplied either as input data or by a communications command connecting $TEMI$ to another module where this temperature is calculated. The metal heat capacity CMF is also calculated in the initial condition section by,

$$CMF = CM * QLPF \quad (3)$$

since it is assumed to be constant.

The coolant density ρ_{OCM} is the first value calculated during a dynamic entry by calling $SODEN1$. It is used in calculating the total coolant heat capacity by,

$$CMC = VMC * CC * \rho_{OCM} \quad . \quad (4)$$

Next, the metal node and coolant node average temperatures are calculated by integrating Eqs. (1) and (2). A centralized integration method is used as defined by the user. Finally, the module exit temperature $TEMX$ is made equal to the coolant average temperature.

(8) APPLICABILITY AND RESTRICTIONS.

The lower plenum inlet temperature must be provided either as input data, if the primary coolant loop is not a closed loop, or by a communications module connecting $TFMT$ to another module where this temperature is calculated. It is also suggested that this module be connected to the core thermodynamic module by a pipe delay to account for the error introduced by the ideal mixing assumption. To use this module in the simulation, the following statement should be included.

- $LPLEN1$;

```

*****
* MACFO1 *
*****

```

(1) PURPOSE.

This module contains all of the basic system macros.

(2) LIST OF MACROS.

```

APPEND#TO#      Appends a module of statements in situ.

CALL#;          Detects a subroutine CALL.

CNCT#(##)       Defines a connection of 2 variables.

CNTRL1          Inserts the CNTRL1 module.

CCMINALI        Inserts all of the COMMON blocks.

COMIN#;         Inserts a COMMON blocks.

COMMON/##/##;   DSNP COMMON definition.

CCNVERG#(##,##,##,##,##)
                 Defines convergence criteria on a variable.

CORTP1          Inserts the CORTP1 module.

DATA(##)=##;    Data statement.

DCFILE(##)      Gets a data set.

DEBUG;##;_%;    Debugs the following DSNP block.

DEBUG ALL;##;_%;
                 Debugs the DSNP block and all the
                 subroutines referenced from the block.

DEFHPIP#(##,##,##,##,##,##,##,##,##,##,##)
                 Defines a hydraulic pipe.

DESTROY         Destroy the last vulnerable macro used.

DFBOIL#1(##,##,##,##,##,##,##,##,##,##,##,##,##,##)
                 Defines a boiler.

DFCAVT#(##,##,##,##,INFLO(##),EXFLO(##),PIPES(##),##,##)
                 Defines a cavity.

DFDRUM#(##,##,##,##,##,##,##,##)
                 Defines a steam drum.

```

DFJUNC#(##)=(INFIC(#),EXFLO(#),INPR(#),EXPR(#),#)
 Defines a hydraulic junction.

DFLOW#(##)=(#,#,#,#,PIPES(#),PUMPS(#))
 Defines a flow path.

DEMXPLEN#(##,#,#,#,INFLO(#),EXFLO(#),#,#)
 Defines a mixing plenum.

DFNJ(#) Defines number of junctions.

DFPIPE#(##,#,#,#,#,#,#,#)
 Defines a thermal pipe. (Delay line)

DFPUMP#(##,#,#,#,#,#,#,#,#,#,#,#,#,#,#,#)
 Defines a pump.

DIIST#TO# Copies a data set during simulation
 execution.

DUMP#TO# Dumps all of the COMMON blocks into a data
 set.

DYM0D Initializes a dynamic module.

DYNAMIC Defines the dynamic section of a module.

ENTRY#; Records the presence of an ENTRY to function
 or subroutine.

FDBEK1 Inserts the FDBEK1 module.

FNCT#(##,#) Defines a single variable function.

FUNCTION#; Records the presence of a function.

GAMAR1 Inserts the GAMAR1 module.

IFE(#)#ELSE#;_%;
 Extended logical IF.

INCLUDE#; Includes a module into the simulation.

INTGRL(##)=(#,#)
 Integrates a single variable.

INTGRL(##)=(#,#,ER=#,#)
 Integrates a single variable with defined
 error criteria.

IN1(##,(#)) Auxilliary to integration.

IN2(##,(#)) Auxilliary to integration.

IN3(##,(#),#)

Subintegration function.

HXM#1(##,##,##)

Defines a heat exchanger.

LAG#=#=# First order lag.

LAG#=#=# Nth order lag.

LIMI(##) #IYVYV(##) #;#;

Limits an integration.

LNTGRL(##) = (##,##,##,##)

Integrates a single variable from a vector of variables.

LOAD FUNC#(##) Loads a function.

LLEN1 Inserts the LLEN1 module.

NEUTP1 Inserts the NEUTP1 module.

NEUTR1 Inserts the NEUTR1 module.

NEUTS1 Inserts the NEUTS1 module.

NOTI(##) #IYVYV(##) #;#;

Threshold excluding integration.

PIST#TO# Copies a data set after compilation.

SAFTY1 Inserts the SAFTY1 module.

SCRAMF(##,##,##) Defines a SCFAM initiating parameter.

SOIG(##) = (##,##,##)

Second order lag.

SCRT(##,##,##,##) and SCRT(##,##,##)

Sorts and copies statements.

STATIC Defines static part in module.

SUBROUTINE#; Records the presence of subroutine.

TERMINAL Defines the terminal section of a module.

TPOWER1 Inserts the TPOWER1 module.

UPLEN1 Inserts the UPLEN1 module.

VECTOREQ(##,##,##) and VECTOREQ(##,##)

Equalizes vectors.

VNTGRL(##) = (##,##,##,##)

Integrates vectors.

VNTGRL (#) = (#, #, #, #, #, #, #, #)
Integrates vectors with given error
criteria.

Z01ZZZ(##-#) = #(#-#) and Z01ZZZ(##-#) = # and Z01ZZZ(##)
Auxilliary functions to VECTOREQ and other
macros.

* MAINP1 *

(1) PURPOSE.

This module is used by the precompiler to generate the DSNP main program.

(2) PARAMETERS.

None.

(3) COMMON VARIABLES.

All of the simulation common blocks are inserted into the main program.

(4) LOCAL VARIABLES.

All of the variables defined by the user as local variables in the static part of the simulation will be local variables in the main program.

(5) OTHER SUBROUTINES OR MODULES REQUIRED.

All of the modules that require initial conditions to be calculated will be referenced from the main program.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

All of the initial conditions that are calculated for the simulation are controlled by the main program. The dynamic part of the simulation is executed by the TIMER control subroutine and the DEPRV subroutine. The statements contained in the main program are those translated from the DSNP static segment.

(7) ORGANIZATION OF MODULE.

The module, as in the rest of the library, includes prefixed statements that will be sorted out later and inserted into different parts of the main program. The presimulation statements that can be found in the DSNP declaration program segment will be executed first. The DERIV routine is then called to clear all the vectors associated with the integration process. Next, the contents of all of the simulation data blocks are transferred to the output printer (I06). The static segment follows next and, if necessary, an appropriate iteration scheme is devised based upon the user CONVERG definitions.

Upon finishing the initial condition calculations, the subroutine TIMER is then called to control the dynamic part of the simulation. After each master time step TIMEM, control is returned to the main program. The TPRIN1 routine is then called and, if so requested by the user, results at this particular time will be printed. An indicator is tested for simulation termination. Every 25th time step the remaining CPU time is checked and, if less than five seconds remain,, the simulation is orderly terminated. Prior to returning control to TIMER1 for the next time step, all of the delay pipe functions are updated.

(8) APPLICABILITY AND RESTRICTIONS.

Nct applicable.

```

*****
* MILNE1 * (LCOP)
*****

```

(1) PURPOSE.

This module performs fifth order integration using the Milne predictor-corrector method.

(2) PARAMETERS.

LOOP Indicator of the loop on which this integration method is used.

(3) COMMON VARIABLES.

<u>NAMES</u>	<u>COMBLK</u>	<u>DESCRIPTION</u>
DELT	TIMER	Current integration time step, in s.
DYDT(I)	DERIV	Vector of the derivatives to be integrated, integrator input.
DYDT1(I)	DERIV	Intermediate or past derivative vector.
DYDT2(I)	DERIV	Intermediate or past derivative vector.
DYDT3(I)	DERIV	Intermediate or past derivative vector.
DYDT4(I)	DERIV	Intermediate or past derivative vector.
DYDT5(I)	DERIV	Intermediate or past derivative vector.
DYDTM1(I)	DERIV	Vector storing past derivatives.
DYDTM2(I)	DERIV	Vector storing past derivatives.
DYDTM3(I)	DERIV	Vector storing past derivatives.
EFMES(I)	TIMER	A vector of words to store simulation error messages.
EYYYY1(I)	DERIV	Maximum relative integration error allowed, I=1 to NODER.

EYYYY2(I)	DERIV	Maximum absolute integration error allowed, I=1 to NODER.
EYYYY3(I)	DERIV	Evaluated integration error for the Ith integrated variable, I=1 to NODER.
ICUB	TIMER	If set to one, indicates that the integration requests doubling of the time step.
IDIV	TIMER	If set to one, indicates one or more time step halving by an integration routine.
ITERM	TIMER	If less than one, indicates that a simulation execution time error has been encountered.
IYYYY(I)	DERIV	Integration indicator. If not equal to LOOP, no integration will be performed.
NCDER	DERIV	Number of integrations to be performed in the simulation.
YYYY(I)	DERIV	Vector of the integrated variables, integrator output.
YYYY1(I)	DERIV	Intermediate or past integration results.
YYYY2(I)	DERIV	Intermediate or past integration results.
YYYY3(I)	DERIV	Intermediate or past integration results.
YYYY4(I)	DERIV	Intermediate or past integration results.
YYYY5(I)	DERIV	Intermediate or past integration results.
YYYYM1(I)	DERIV	A vector storing past values of the Ith integrated variable.
YYYYM2(I)	DERIV	A vector storing past values of the Ith integrated variable.
YYYYM3(I)	DERIV	A vector storing past values of the Ith integrated variable.

(4) LOCAL VARIABLES.

DT3 The value of DELT/3.

DT192 The value of DELT/192.

DT364 The value of DELT*3/64.

ER The integration error.

ERMESI The text "IN MILNE".

IDB The number of local time steps in one simulation time step (DELT/DELTS).

ISTEP The internal time step counter.

TIMES The internal time set to TIME upon entering the subrcutine, in s.

YCOR The corrector value of the integration step.

(5) OTHER SUBROUTINES OR MODULES REQUIRED.

Subroutines: DERIV1, EULER1.

Common Blocks: CONST5, DERIV5, TIMER5.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

The Milne fifth order predictor-corrector integration algorithm for variable or fixed time steps is given by,

$$T(t+Dt) = 0.96116 * YC(t+Dt) + 0.3884 * YP(t+Dt) \quad , \quad (1)$$

where the predictor YP is given by,

$$YP(t+Dt) = Y(t-Dt) + Dt * [8 * dy(t) / dt - 5 * dy(t-Dt) / dt + 4 * dy(t-2Dt) / dt - dy(t-3Dt) / dt] / 3 \quad , \quad (2)$$

and the corrector YC is given by,

$$YC(t+Dt) = [Y(t) + 7 * Y(t-Dt)] + Dt * [65 * dYP(t+Dt) / dt + 243 * dy(t) / dt + 51 * dy(t-Dt) / dt + dy(t-2Dt) / dt] / 192 \quad . \quad (3)$$

The integration error is given by,

$$ER = |YC - YP| / (|ERR * YC| + ERA) \quad , \quad (4)$$

where ERR is the maximum relative allowed integration error and ERA is the maximum absolute allowed error.

The magnitude of the next integration time step Dt is determined by the following algorithms.

If $0.05 \leq ER \leq 1.0$, $Dt(new) = Dt$.

If $ER \leq 0.05$, $Dt(new) = 2 * Dt$.

If $ER \geq 1.0$, $Dt(new) = Dt / 2$.

If ER is greater than one in any integration step, the integration step is repeated and time is not advanced until ER is less than one. The simulation is terminated if the $Dt(new)$ falls below $DTMIN$.

(7) ORGANIZATION OF MODULE.

The internal time $TIMES$ and the internal integration steps counter $ISTEP$ are set upon entering the subroutine. The execution process is dependent on whether the new integration time step is equal to the previous time step and whether more than four integration steps took place already. Since the $MILNE1$ method requires the values of the integrated variables and their derivatives at four previous time steps, the integration is initialized by first performing four Euler integration steps with $DELT/4$. During each of the four initial steps, the "past" value vectors $YYY1$ to $YYYY5$, $DYDT1$ to $DYDT5$, $YYYYM1$ to $YYYYM3$, and $DYDTM1$ to $DYDTM3$ are updated. The vectors with the suffix five have the most recent values. The vectors with the suffixes $M1$ to $M3$ store the values at $TIME - 5 * DELT$ to $TIME - 7 * DELT$ that are used in case the integration time step is doubled.

If the time step size does not change from one integration step to another, the above equations are solved and the "past" vectors are updated following each successful integration step.

If the integration step is doubled, the same equations are still solved. The "past" vectors 4, 2, $M3$ and $M1$ are used instead of 4, 3, 2 and 1. If the time step is halved internally because the integration error criteria was not met, the "past" vectors are recalculated using specific interpolation schemes to fill in the half step gaps.

(8) APPLICABILITY AND RESTRICTIONS.

The $MILNE1$ integration algorithm is fast and precise for rapidly varying transients. It only requires two evaluations of the derivatives while the $RUNGE1$ method requires four evaluations for each time step. It is

necessary, however, to have 18 additional storage locations for each variable integrated, which may become prohibitive with large simulations.

```

*****
* MOTOR1 *   (KP1,ZTORD,LOOP)
*****

```

(1) PURPOSE.

This subroutine simulates a simple motor driving a pump in the simulator.

(2) PARAMETERS.

KP1 The pump number which the motor is driving.

LCOP Indicator of the loop from which this subroutine is called.

ZTORD The torque delivered to the pump, in N-m.

(3) COMMON VARIABLES.

<u>NAMES</u>	<u>COMBIK</u>	<u>DESCRIPTION</u>
JTIM	TIMER	Time step counter.
ZPTRIP	HPUMP	A motor torque multiplier, that if becomes smaller than one during the simulation, causes the pump to trip.
ZREV(I)	HPUMP	Speed of the pump, in radians/s.
ZREVR(I)	HPUMP	Rated speed of the pump, in radians/s.
ZTORPR(J)	HPUMP	Rated pump torque, N-m.

(4) LOCAL VARIABLES.

EFS The relative error in pump speed.

(5) OTHER SUBROUTINES OR MODULES REQUIRED.

Common Blocks: CONST5, HPUMP5, TIMER5.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

This simple model maintains the torque to keep a constant pump speed. The difference between the pump rated speed and the pump speed is calculated by,

$$EPS = (ZREFV(KP1) - ZREVR(KP1)) / ZREVR(KP1) \quad , \quad (1)$$

from which the torque to be delivered to the pump is calculated from,

$$ZTORD = ZTORPR(KP1) * (12 * EPS) \quad . \quad (2)$$

The maximum torque delivered can be three times the required rated torque for the KP1th pump, ZTORPR(KP1).

(7) ORGANIZATION OF MODULE.

When the subroutine is first entered, the pump trip coefficient is set to one, and the delivered torque is set to the pump requested torque. During any time step for which the trip function ZPTRIP(KP1) is equal to one, Eqs. (1) and (2) are solved. If a pump trip was requested for the KP1th pump, that is ZPTRIP(KP1) was set to less than one, the delivered torque ZTORD is set to ZPTRIP(KP1)*ZTORPR(KP1) and stays at that value for the rest of the simulation.

(8) APPLICABILITY AND RESTRICTIONS.

This simplified model of the pump motor provides the simulated pump with the necessary driving torque. This module is automatically included whenever a pump is included in the simulation. It will maintain a constant pump speed by supplying a torque of up to three times the rated pump torque. This model will be replaced in the future by an improved model considering motor inertia and motor speed control mechanisms.

```

*****
* MXPLE1 * (ICCF,F1in,T1in,...,F1ex,T1ex,...ZVOLPL)
*****

```

(1) PURPOSE.

This module contains macros and DSNP statements producing a subroutine that will simulate the thermodynamics of a mixing plenum. The plenum has a metal wall node and a fluid node. There can be any number of inlets and exits to the plenum.

(2) PARAMETERS.

F1ex, F2ex, ... Exit flows from the plenum, in kg/s.

F1in, F2in, ... Inlet flows to the plenum, in kg/s.

LOOP Indicator to the loop from which this subroutine is called.

T1ex, T2ex, ... Temperatures of outlet flows, in C.

T1in, T2in, ... Temperatures of inlet flows, in C.

ZVOLPL Mixing plenum volume, in m³.

This subroutine will have a variable number of parameters according to the definition used.

(3) COMMON VARIABLES.

JTIM TIMER Time step counter.

ZFPLE# MXPL# Pressure in a mixing plenum, in Pa.

ZRPLE# MXPL# Mixing plenum fluid density, in kg/m³.

ZTM#A MXPL# Mixing plenum average inlet temperature, in C.

ZTPF# MXPL# Mixing plenum fluid average temperature, in C.

ZTPM# MXPL# Mixing plenum metal average temperature, in C.

(4) LOCAL VARIABLES.

TAV Average fluid inlet temperature, in C.
ZCPF Average fluid specific heat, in J/kg-C.
ZCPM Average metal node heat capacity, in J/kg-C.
ZEXFLO Total exit flow, in kg/s.
ZINFLO Total inlet flow, in kg/s.
ZROF Average fluid density, in kg/m³.
ZUMET Heat transfer coefficient from metal to coolant
 node, in J/C.

(5) OTHER SUBROUTINES OR MODULES REQUIRED.

Subroutines: SODEN1, or STDEN1, or WADEN1, SODCP1, or
 STDCP1, or WADCP1, SSDCP1.
Common Blocks: CONST5, MXPL#5, TIMEF5.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

This is a simple lumped parameter thermodynamic model of a mixing plenum. All of the metallic material in the plenum is lumped into a single node and the coolant is lumped in a single fluid node. An arbitrary number of inlet and outlet flows can be defined. The subroutine solves the following heat balance equation.

$$ZTPM\# = \text{INTEGRAL} [(ZTPF\# - ZTPM\#) * ZUMET / (Q_{met} * ZCPM)] \quad , \quad (1)$$

where Q_{met} is the mass of the metal node in kg, and

$$ZTPF\# = \text{INTEGRAL} [(ZINFLO * ZCPF * ZTM\#A - ZEXFLO * ZCPF * ZTPF\# - ZUMET * (ZTPF\# - ZTPN\#)) / (ZVOLPL * ZRPLE\# * ZCPF)] \quad . \quad (2)$$

(7) ORGANIZATION OF MODULE.

Upon entering the subroutine, the total inlet and exit flow is computed by summing up the individual flows.

$$ZINFIC = \text{SUM:FIin} \quad \text{and} \quad ZEXFLO = \text{SUM:FIex} \quad (3)$$

Knowing the total flow, the average inlet temperature is computed by,

$$TAV = \text{SUM}:[F1in * T1in] / ZINFLO \quad . \quad (4)$$

For the static calculations, ZTPM# and ZTPF are set to TAV and control is returned to the calling program.

For the dynamic solution, the average inlet temperature is first calculated and compared to the previous value TAV1. If the difference is greater than 1 C., the material properties are reevaluated. Finally, the heat balance Eqs. (1) and (2) are solved.

(8) APPLICABILITY AND RESTRICTIONS.

This mixing plenum model can be used instead of UPLEN1 and LPLEN1 when more than one flow entry or exit is present. It can be used to simulate mixing plena at the entrances and exits of all the power plant components. A subroutine is produced by the system for each mixing plenum defined by,

```
• DFMXELENa(Mfl,Mme,ZVOLPL,Qmet,ZUMET,INFLO(F1in,T1in,...)
    EXFLO(F1ex,T1ex....),ZTMET,ZTEFLA); ,
```

where a is any recognizable character or number identifying the particular plenum. Mfl and Mme are material indicators for the fluid (SO, WA, or ST) and for the metal (SS). Qmet is the weight of the metal in the cavity. The heat transfer coefficient ZUMET can be a number, a variable or an expression to be evaluated in the subroutine. To use this module in the simulation, the following statement should be included.

```
• MXPLENa;
```

```

*****
* NEUTP1 * (ICCP)
*****

```

(1) PURPOSE.

This module solves the prompt jump approximation to the neutron kinetic equations.

(2) PARAMETERS.

ICOP Indicator to the loop from which this subroutine is called.

(3) COMMON VARIAPLES.

<u>NAMES</u>	<u>COMBLK</u>	<u>DESCRIPTION</u>
BETA(I)	NEUTR	Delayed neutron fraction, I=1...6.
CLJ(I)	NEUTR	Delayed precursor concentration at the Jth time step, I=1...6.
DCL(I)	NEUTR	Derivative of the delayed neutron precursors, I=1...6.
EI	NEUTR	Prompt neutron life-time, in s.
EIAMDA(I)	NEUTR	Delayed precursor decay constants, I=1...6.
PNO	NEUTR	Normalized initial neutronic power level.
PNJ	NEUTR	Normalized neutronic power at the Jth time step.
PEAK	NEUTR	Total core excess reactivity.
PKCN	CNTRI	Reactivity generated by the control system.
RKFB	FDBEK	Total feedback reactivity.
RKSF	SAFTY	Reactivity generated by the safety system.
SOURCE	NEUTR	Value of neutron source in core startup simulations.

SUMC NEUTR Sum of CLJ(I)*FLAMDA(I) for I=1...6.

(4) LOCAL VARIABLES.

BETAT Total delayed neutron fraction.

DPNJ Derivative of the normalized neutronic power.

(5) OTHER SUBROUTINES OR MODULES REQUIRED.

Common Blocks: CONST5, CNTRL5, DEPIV5, FDBEK5, NEUTR5,
SAFTY5, TIMER5.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

The same point kinetic equations as given in NEUTR1 are solved here. Instead of computing the time derivative of the normalized neutronic power PNJ, the prompt jump approximation is used where the DPNJ derivative is assumed to be zero. PNJ is then calculated directly from,

$$PNJ = SUMC * FL / (BETA - PEAK) \quad . \quad (1)$$

The delayed neutron equations are solved as in NEUTR1.

(7) ORGANIZATION OF MODULE.

The module is organized exactly as in NEUTR1.

(8) APPLICABILITY AND RESTRICTIONS.

For short neutron lifetimes, below approximately $10E-6$, and reactivity values of less than one dollar, the prompt jump approximation gives the same results as a complete solution. A significant reduction in the number of time steps required to achieve a solution is usually experienced. To use this module in the simulation, the following statement should be included.

• NEUTP1;

```

*****
* NEUTR1 *   (LCOP)
*****

```

(1) PURPCSE.

This module solves the neutron point kinetic equations.

(2) PARAMETEPS.

LCOP Indicator to the loop from which this subroutine is called.

(3) COMMON VARIABLES.

<u>NAMES</u>	<u>COMBLK</u>	<u>DESCRIPTION</u>
BETA(I)	NEUTR	Delayed neutron fraction, I=1...6.
CIJ(I)	NFUTR	Delayed precursor concentration at the Jth time step, I=1...6.
DCL(I)	NEUTR	Derivative of the delayed neutron precursors, I=1...6.
EL	NEUTR	Prompt neutron life-time, in s.
EIAMDA(I)	NFUTR	Delayed precursor decay constants, I=1...6.
PNO	NEUTP	Normalized initial neutronic power level.
PNJ	NEUTR	Normalized neutronic power at the Jth time step.
BEAK	NEUTR	Total core excess reactivity.
RKCN	CNTFL	Reactivity generated by the control system.
RKFB	FDBEK	Total feedback reactivity.
RKSF	SAFTY	Reactivity generated by the safety system.
SOURCE	NEUTR	Value of neutron source in core startup simulations.

SUMC NEUTR Sum of CLJ(I)*ELAMDA(I) for I=1...6.

(4) LOCAL VARIABLES.

BETAT Total delayed neutron fraction.

DPNJ Derivative of the normalized neutronic power.

(5) OTHER SUBROUTINES OR MODULES REQUIRED.

Common Blocks: CONST5, CNTRL5, DERIV5, FEDEK5, NEUTR5,
SAFTY5, TIMER5.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

The point kinetic equations with six groups of delayed neutrons are solved in this module. The module calculates the time derivative DPNJ of the normalized neutronic power by,

$$DPNJ = (RFAK - BETA) * PNJ / EL + \sum_{i=1}^{I=6} CLJ(I) * ELAMDA(I) \quad , \quad (1)$$

and the time derivative of the delayed neutron precursors by,

$$DCL(I) = BETA(I) * PNJ / EL - ELAMDA(I) * CLJ(I) \quad . \quad (2)$$

A centralized integration method defined by the user solves Eqs. (1) and (2) simultaneously with the other system equations.

(7) ORGANIZATION OF MODULE.

Upon first entry to the module, the initial conditions for PNJ and CLJ(I) are computed by,

$$PNJ = PNO \quad , \quad (3)$$

and,

$$CLJ(I) = BETA(I) * PNJ / (EL * ELAMDA(I)) \quad . \quad (4)$$

Subsequent entries will skip the initial condition calculations.

The input that drives this module is the total reactivity REAK obtained by summing up the reactivities from the different sources as,

$$REAK = R_{KCN} + R_{KSF} + R_{KFB} \quad . \quad (5)$$

Finally, the derivatives are computed by using Eqs. (1) and (2), and submitted to the appropriate integration routine.

(8) APPLICABILITY AND RESTRICTIONS.

This module should be used with caution. The kinetic equation have one very small time constant $EL=10E-5$ to $10E-7$ and several time constants of $1/ELAMDA=0.1$. Together with the other system equations, they form a set of stiff differential equations. Integrators with constant increments require very small time steps to achieve a stable solution. Integrators with variable time steps and error control will keep the time step automatically very small. Thus long solution times can be expected for the loop in which the NEUTF1 module is inserted.

Two stiff integration methods can be used. The STIFF1 method adapted from CSMP-III [1], and the GEAR02 method developed by Gear [2]. Both methods significantly improve the solution time for stiff equations.

Alternatively, the prompt-jump approximation module NEUTP1 can be used. In particular, if system transients are to be simulated for long periods of time, and no prompt critical condition occurs, the prompt-jump approximation is well justified.

In addition, a self integrating module NEUTS1 is also available. This module uses an efficient integration method developed by Kaganove [3]. If this module is used, it should be placed in a nonintegrated loop (see section on integration methods). To use this module in the simulation, the following statement should be included.

- NEUTP1;

REFERENCES.

- [1] IBM, "Continuous System Modeling Program III (CSMP-III).", SH19-7001-2, (1972)
- [2] C. W. Gear, "Numerical Initial Value Problems in Ordinary Differential Equations", Prentice Hall, (1971).

- [3] J. J. Kaganove, "Numerical Solution of the One-Group Space-Independent Reactor Kinetics Equations for Neutron Density Given the Excess Reactivity", Argonne National Laboratory, ANL-6132, (1960).

```

*****
* NEUTS1 *
*****

```

(1) PURPOSE.

This module solves the lumped parameter neutron kinetic equations using the integration scheme developed by Kaganove [1].

(2) PARAMETERS.

Ncne.

(3) COMMON VARIABLES.

<u>NAMES</u>	<u>COMBIK</u>	<u>DESCRIPTION</u>
BETA(I)	NEUTR	Delayed neutror fraction, I=1...6.
CLJ(I)	NEUTR	Delayed precursor concentration at the Jth time step, I=1...6.
DCI(I)	NEUTR	Derivative of the delayed neutron precursors, I=1...6.
EL	NEUTR	Prompt neutron life-time, in s.
ELAMDA(I)	NEUTR	Delayed precursor decay constants, I=1...6.
EPSN	NEUTR	Integration step error in NEUTS1.
EPSN1	NEUTR	Lower integration error limit in NEUTS1. Double time step if EPSN is less than EPSN1.
EPSN2	NEUTR	Upper integration error limit in NEUTS1. Halve time step if EPSN is larger than EPSN2.
ERRMES(I)	TIMER	A vector of words to store simulation error messages.
HDTMIN	TIMER	Half of the minimum time step size.
ITERM	TIMER	If less than one, indicates that a simulation execution time error has been encountered.

JTIM	TIMER	Time step counter.
ENO	NEUTR	Normalized initial neutronic power level.
PNJ	NEUTR	Normalized neutronic power at the Jth time step.
REAK	NEUTR	Total core excess reactivity.
RKCN	CNIFL	Reactivity generated by the control system.
RKFB	FDBEK	Total feedback reactivity.
RKSF	SAFTY	Reactivity generated by the safety system.
SOURCE	NEUTR	Value of neutron source in core startup simulations.
SUMC	NEUTR	Sum of $CLJ(I) * FLAMDA(I)$ for $I=1...6$.
TEPSN	TIMER	$EPSN / EPSN1$.

(4) LOCAL VARIABLES.

DELTN	The internal neutronic time step, in s.
DPN	The integration increment of the normalized neutronic power.
DPN1	The integration increment of the full step of the normalized neutronic power.
DPN2	The integration increment of two half steps of the normalized neutronic power.
DPNH	The integration increment of the first half step of the normalized neutronic power.
DREAK	The reactivity increment in the time interval DEIT.
DBK	The linearly interpolated reactivity increment in the time interval TAU.
ENTO (I)	$(1 - \text{DEXP}(-\text{ELAMDA}(I) * \text{TAU})) / \text{ELAMDA}$
ENT1 (I)	$(\text{TAU} - \text{ENTO}(I)) / (\text{ELAMDA}(I) * \text{TAU})$
ERMESL	The text "IN NEUTS".

ITER The number of iterations in the internal integration process.

K A counter of integration steps.

PNP The same as PNJ, normalized neutronic power.

RKA $(FAK+DEK/2)*TAU$.

RKB $(FAK/2+DRK/3)*TAU$.

SUMB The sum of $BETA(I)*ENT1(I)$.

SUMC The sum of $DCI(I)*ENT(I)$.

TAU The integration time step, in s. The same as DEITN except when halved in the iteration process.

TEPSN The value of $EPSN/EPSN1$.

TIMEN The internal time in this module or $TIME<TIMEN<TIME+DELT$.

(5) OTHER SUBROUTINES OR MODULES REQUIRED.

Common Blocks: CONST5, CNTRL5, FDBEK5, NEUTR5, SAFTY5, TIMERS5.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

The algorithm and method of solution used in this module are given in reference [1].

(7) ORGANIZATION OF MODULE.

Upon first entry to the subroutine, initialization takes place. The steady state delayed neutron fraction is computed, either based upon the normalized neutronic initial power PNO, or the neutron source SOURCE, if a startup is simulated. During each entry while the dynamic simulation is in progress, the change in reactivity for the present time step DPEAK is computed first. The integration process of the kinetic equation is then performed for the time step DELTN which may be equal to or less than the simulation time step DELT. In the latter case, more than one integration step will take place in this subroutine before control is returned to the calling program. Following the DELTN integration step, the DELTN

time step is halved and two similar integration steps are performed. Then, the difference between the single and double integration steps is computed by,

$$\text{EPSN} = \text{DAPS} ((\text{DEN1} - \text{DPN2}) / \text{PNJ}) \quad . \quad (1)$$

If EPSN is greater than EPSN1 and less than EPSN2, the integration is successful and terminated for the present time step. If EPSN is less than EPSN1, that is single and double steps give the "same results", the next integration step will be doubled, if other parts of the simulation do not object and IDUB is set to one. If EPSN is greater than EPSN2, the time step is halved and the last integration step is repeated using the previous half step result as a full time step estimate. This process is continued until the error criteria are met. The integration process in this subroutine is continued until TIMEN is equal to TIME+DELT when the internal and external times match within HDTMIN.

(8) APPLICABILITY AND RESTRICTIONS.

This module is a self contained unit. Because it does not use the centralized integration scheme, it should be used with care. Mostly it should be used in situations where the kinetic module is used independently, as in startup accident simulation for example. The module should be placed in a NOINT loop. To use this module in the simulation, the following statement should be included.

• NEUTS1;

REFERENCES.

- [1] J. J. Kaganove, "Numerical Solution of the One-Group Space-Independent Reactor Kinetics Equations for Neutron Density Given the Excess Reactivity", Argonne National Laboratory, ANL-6132, (1960).

* NOINT1 * (ICCF)

(1) PURPOSE.

This module is a dummy integration subroutine to be used with nonintegrating loop(s) that have modules with internal integration.

(2) PARAMETERS.

LOOP Indicator of the loop in which integration is not requested.

(3) COMMON VARIABLES.

<u>NAMES</u>	<u>COMBLK</u>	<u>DESCRIPTION</u>
IDIV	TIMER	If set to one, indicates one or more time step halving by an integration routine.
IDUB	TIMFR	If set to one, indicates that the integration requests doubling of the time step.
TIME	TIMER	Current simulation time, in s.

(4) LOCAL VARIABLES.

None.

(5) OTHER SUBROUTINES OR MODULES REQUIRED.

Subroutines: DEPIV1.
Common Blocks: TIMERS.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

None.

(7) ORGANIZATION OF MODULE.

The subroutine is used in those simulation loops which do not require integration or have self integrating subroutines. It will set IDIV and IDUB to zero, increment the intermediate integration time and call DERIV1 to advance the particular loop by one time step.

(8) APPLICABILITY AND RESTRICTIONS.

This module is used in the SIMULATE DSNP control statements that have an integration method as a parameter. Each module inserted in the NOINT loop must have a self integrating algorithm compatible with the overall time control scheme.

* PIPE01 * (IP,FIN,FEX,VPID,EF,ET)

(1) PURPOSE.

This module simulates a general pipe segment with pure time delay and has three different entries: PIPE01 for initialization; PIPEL1 for updating; and PIPER1 for obtaining the exit value at any time.

(2) PARAMETERS.

EF The minimum change in a pipe inlet variable initiating a storage operation.

ET The minimum time increase since the last event initiating a storage operation, in s.

FEX A pipe exit value at any chosen time.

FIN A pipe inlet variable.

G The flow rate through the pipe, in m^3/s .

IP A pipe identification number assigned by the precompiler.

KK The degree of interpolation used to evaluate the pipe exit variable FEX if requested at times falling between two tabulated values.

VPID The volume of the pipe section, in m^3 .

(3) COMMON VARIABLES.

<u>NAMES</u>	<u>COMBLK</u>	<u>DESCRIPTION</u>
ERRMES(I)	TIMER	A vector of words to store simulation error messages.
ITERM	TIMER	If less than one, indicates that a simulation execution time error has been encountered.
TIME	TIMER	Current simulation time, in s.

(4) LOCAL VARIABLES.

EFI(I) EF for each pipe segment. I=1 to IPN.

ETI(I) ET for each pipe segment. I=1 to IPN.

FINT(I,J) A pipe storage vector. I=1 to MPST, J=1,2, and where J=1 is for the pipe inlet variable FINT and J=2 is for the time at which the corresponding variable is stored.

GVP The volumetric flowrate through the pipe, in m^3/s .

IGO An indicator to the subroutine entry.

IPN The number of pipe segments in the simulation.

KLAST(I) The location of the lowest function value used in the last interpolation for the Ith pipe.

MF The number of pipe storage locations for the present entry.

MPM MP2-1

MPS(I) The number of storage locations for each pipe. I=1 to IPN.

MPST The number of storage locations for all of the pipe segments.

MP1 The first storage location for the present pipe entry.

MP2 The last storage location for the present pipe entry.

TAU The pipe transit time, in s.

TIM The time at which the function is to be stored or evaluated according to the entry used, in s.

VPL(I) VPLD for each pipe segment. I=1 to IPN.

ZDDT The previous time interval divided by the last two time intervals.

ZFINI The linearly interpolated value for the last FIN at the last time from the present FIN. The difference between ZFIN1 and the actual value stored will determine whether a storage operation is performed.

(5) OTHER SUBROUTINES OR MODULES REQUIRED.

Subroutines: FUNCT1
Common Blocks: TIMEP5

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

This pipe function is a pure delay. A vector FINT(I,J) is used to store entry values whenever the change in the input variable FIN will cause a linear interpolation error greater than ET. The interpolated function ZFINI is calculated by,

$$ZFINI = (FIN - FINT(MPM, 1)) * ZDDT + FINT(MPM, 1) \quad , \quad (1)$$

where ZDDT is given by,

$$ZDDT = (FINT(MP2, 2) - FINT(MPM, 2)) / (TIME - FINT(MPM, 2)) \quad . \quad (2)$$

ZFINI is compared to the actual stored value FINT(MP2,1) by,

$$| (FINT(MP2, 1) - ZFINI) / FINT(MP2, 1) | > ET \quad . \quad (3)$$

If the relative error caused by inserting the last value is less than ET, only the last value is replaced by the new value of FIN and TIME, and the other pipe values are not updated.

If the time increment since the last storage occurred is greater than ET or,

$$TIME - FINT(MP2, 2) > ET \quad . \quad (4)$$

Then regardless of the error, the pipe is updated by,

$$FINT(I, J) = FINT(I+1, J) \quad , \quad I=MP1 \text{ to } MP2, \quad J=1, 2 \quad (5)$$

and the last value in the vector is set to,

$$FINT(MP2, 1) = FIN, \quad FINT(MP2, 2) = TIME \quad . \quad (6)$$

When the pipe exit values are requested, an KKth order interpolating polynomial is used by calling FUNCT1 which uses the Lagrange interpolation method. See modules FUNCT1 and LAGRN1 for details.

(7) ORGANIZATION OF MODULE.

This module has three separate entries according to the function to be performed. The various vectors used in this

subroutine are inserted by the DSNP precompiler according to the pipe segments used in this simulation.

SUBROUTINE PIPE01(IP,FIN,FEX,VPLD,EF,ET)

This is the main entry of the subroutine and causes the initialization of the pipe segment. It will store ET, EF, VPLD, and MPS into the appropriate vectors. It will set all of the pipe storage locations to FIN. It will set the last time storage location FINT(MP2,2) to the initial time and decrease TIME by ET for each previous location and will set the first lookup location KLAST(IP) to two.

$$FINT(I,2) = FINT(I+1,1) - ET, I=MP2-1 \text{ to } MP1 \quad (7)$$

ENTRY PIPE11(IP,FIN)

This is the update entry. The pipe storage will be updated as shown in Eqs. (5) and (6), if the conditions of the interpolation error and storage interval are met. Otherwise control is returned to the calling program.

ENTRY PIPEP1(IP,FEX,G,KK)

This is the retrieval entry for the pipe exit variable. First, the entry time is tested. If it is equal to the TIAST(IP), which is the previous time at which evaluation was performed, control is returned to the calling program without reevaluating FEX. The parameters of the IPth pipe segment are calculated next. Using the volumetric flow G, the pipe transit time TAU is calculated by,

$$TAU = VPL(IP) / GVP \quad . \quad (8)$$

The lookup time TIM is given by,

$$TIM = TIME - TAU \quad . \quad (9)$$

If TIM is less than the first entry time FINT(MP1,2) in the pipe storage, a warning message is issued and ITERM is set to four. Finally, the pipe exit value is calculated by calling FUNCT1, which performs the proper interpolation and control is then returned to the calling program.

(8) APPLICABILITY AND RESTRICTIONS.

Each simulated pipe segment must be defined in the DSNP declarations program segment as,

• DEFPIPEii(FIN,FEX,VPLD,EF,ET,G,KK,MP); .

The size of the vectors used by PIPE01 are determined dynamically by the precompiler. Once the above definitions

have been made, PIPE01 or any of its entries can be used either by a DSNP statement or by a FORTRAN CALL statement. There is no limit to the number of space allocations for a pipe segment and the number of pipes that can be simulated is 100. The interpolation level parameter KK used with FUNCT1 must be in the range of one to nine. Note that the library contains only a part of the actual pipe subroutine. The appropriate dimensions and data statements, as well as the calling sequences, are produced by the precompiler. Usually G is given as F/RO. To use a pipe segment in the simulation, the following statement should be included.

- PIPEii;

* RENUP1 * (D,F,CP,CK,DM,ZPE,ZPR,ZPE)

(1) PURPOSE.

This subroutine calculates the dimensionless flow parameters, namely the Reynolds, the Prandtl and the Peclet numbers.

(2) PARAMETERS.

CK Fluid conductivity, in W/m-C.
CP The fluid heat capacity, in J/kg-C.
D The hydraulic diameter of the flow channel, in m.
DM The dynamic viscosity, in kg/sec-m.
F The flow, in kg/s.
ZPE The Peclet number (Dimensionless).
ZPR The Prandtl number (Dimensionless).
ZRE The Reynolds number (Dimensionless).

(3) COMMON VARIABLES.

None.

(4) LOCAL VARIABLES.

None.

(5) OTHER SUBROUTINES OR MODULES REQUIRED.

None.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

The Reynolds number is given by the following equation.

$$ZRF = F / (D * DM) \quad (1)$$

The Prandtl number is given by the following equation.

$$ZPF = CP * DM / CK \quad (2)$$

The Peclet number is given by the following equation.

$$ZPE = ZRF * ZPF \quad (3)$$

(7) ORGANIZATION OF MODULE.

All of the subroutine input parameters used to evaluate the dimensionless flow parameters must be given in SI units.

(8) APPLICABILITY AND RESTRICTIONS.

The dimensionless parameters calculated by this subroutine are used by the heat transfer and flow subroutines. Note that D is the hydraulic diameter defined as D equals $4A/WP$, where A is the cross sectional flow area and WP is the wetted perimeter of the flow channel. To use this module in the simulation, the following statement should be included.

```
CALL RFNUP1(D,F,CK,DM,ZRF,ZPF,ZPE)
```

```
*****
* RUNGC1 *   (LCCP)
*****
```

(1) PURPOSE.

This module performs fourth order Runge-Kutta integration for a variable time step using the double step checking method to control the integration error and to adjust the time step.

(2) PARAMETERS.

LCCP An indicator of the loop on which this integration method is used.

(3) COMMON VARIABLES.

<u>NAMES</u>	<u>COMBLK</u>	<u>DESCRIPTION</u>
DYDT(I)	DERIV	Vector of the derivatives to be integrated, integrator input.
DELT	TIMER	Current integration time step, in s.
ERRMES(I)	TIMER	A vector of words to store simulation error messages.
EYYYY1(I)	DERIV	Maximum relative integration error allowed, I=1 to NODER.
EYYYY2(I)	DERIV	Maximum absolute integration error allowed, I=1 to NODER.
EYYYY3(I)	DERIV	Evaluated integration error for the Ith integrated variable, I=1 to NODER.
IDIV	TIMER	If set to one, indicates one or more time step halving by an integration routine.
IDUB	TIMER	If set to one, indicates that the integration requests doubling of the time step.
IYYYY(I)	DERIV	Integration indicator. If not equal to IOOP, no integration will be performed.

NCDER	DERIV	Number of integrations to be performed in the simulation.
TIME	TIMER	Current simulation time, in s.
YYYY(I)	DERIV	Vector of the integrated variables, integrator output.
YYYY2(I)	DERIV	Intermediate or past integration results.

(4) LOCAL VARIABLES.

DELT1 The local time step, in s.

ER The calculated error in an integration step.

EFMES1 The text "IN RUNGC".

TIME1 The local time, in s.

(5) OTHER SUBROUTINES OR MODULES REQUIRED.

Subroutines: DERIV1, RUNGE1.
Common Blocks: CONST5, DERIV5, TIMEF5.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

The same fourth order Runge-Kutta algorithm as used in RUNGE1 is used to perform the integration for a time step. This subroutine, however, has an error control algorithm that allows control of the integration time step. First, an integration step from TIME to TIME+DELT is executed by calling FUNGE1. The time step is then halved and the integral is evaluated twice. First from TIME to TIME+DELT/2 and then from TIME+DELT/2 to TIME+DELT. The result of the last step YYYY is then subtracted from the full step result YYYY2.

$$EYYYY3 = \text{DABS}(YYYY2 - YYYY) \quad (1)$$

The relative error is obtained by,

$$ER = EYYYY3 / (\text{DABS}(EYYYY1 * YYYY) + EYYYY2) \quad (2)$$

If ER is less than one half, then IDUB is set equal to one and control is returned to timer. If ER is greater than one, the time step is halved and the integration is repeated from the second step or for two half step

integrations. The previous half step integration serves as a full step estimate on the second iteration and so on until convergence is achieved.

(7) ORGANIZATION OF MODULE.

The integration is performed only on those YYYY vector elements for which IYYYY(I) is equal to 100P. After an unsuccessful time step, IDIV is set to one indicating to the time step controller that the time step was reduced in this program segment. When IDUB is equal to one, indicates a request for the doubling of the time step. After each halving of the time step, the magnitude is tested. If the new time step is less than DTMIN, an error message is printed and the simulation is terminated.

(8) APPLICABILITY AND RESTRICTIONS.

This integration method was found to be very accurate and stable. It will allow larger time steps than other integration methods. It requires 12 evaluations of the derivative for each time step, however, and is therefore more expensive than other methods. It should be used in cases where great precision is required and as a test case for other integration methods.

```
*****
* RUNGE1 *   (ICCP)
*****
```

(1) PURPOSE.

This module performs fourth order Runge-Kutta integration.

(2) PARAMETERS.

LCOP An indicator of the loop on which this integration method is used.

(3) COMMON VARIABLES.

<u>NAMES</u>	<u>CCBLK</u>	<u>DESCRIPTION</u>
DELT	TIMER	Current integration time step, in s.
DYDT(I)	DERIV	Vector of the derivatives to be integrated, integrator input.
DYDT1(I)	DERIV	Intermediate or past derivative vector.
DYDT2(I)	DERIV	Intermediate or past derivative vector.
DYDT3(I)	DERIV	Intermediate or past derivative vector.
NODER	DERIV	Number of integrations to be performed in the simulation.
TIME	TIMER	Current simulation time, in s.
YYYY(I)	DERIV	Vector of the integrated variables, integrator output.
YYYY1(I)	DERIV	Intermediate or past integration results.

(4) LOCAL VARIABLES.

DELT1 Stores DELT at the entry to the subroutine.

DT6 DELT/6.

K An index of the Runge Kutta step being calculated.

TIME1 A storage for the time at the entrance to the
 subroutine.

(5) OTHER SUBROUTINES OF MODULES REQUIRED.

Subroutines: DERIV1.

Common Blocks: CONST5, DERIV5, TIMER5.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

The classical fourth order Runge-Kutta [1] algorithm, as shown below, is used in this integration scheme.

$$YYYY = YYYYY1 + DT6 * (DYDT1 + 2 * DYDT2 + 2 * DYDT3 + DYDT) \quad (1)$$

The indexing of the variables was omitted for clarity. The various time derivatives in Eq. (1) are evaluated at,

DYDT1 at TIME and with YYYYY1 ,

DYDT2 at TIME+DELT/2 and with YYYYY1+DYDT1*DELT/2 ,

DYDT3 at TIME+DELT/2 and with YYYYY1+DYDT2*DELT/2 ,

and DYDT at TIME+DELT and with YYYYY1+DYDT3*DELT .

(7) ORGANIZATION OF MODULE.

Only those variables in the YYYY vector are integrated for which IYYYY(I) are equal to LOOP. The fourth order algorithm calls for four derivative evaluations that are performed in a four step loop by calling DERIV1 to compute the derivatives. Prior to this call, TIME and YYYY are properly updated. The last DERIV1 call is used to compute the first derivative for the next time step. This is a fixed time step integration method. The time step can be changed, however, externally with no effect on the integration process if the proper error control is used. This is actually done by RUNGC1 which then allows the use of this integration method with a variable time step.

REFERENCES.

- [1] P. Henrici, "Elements of Numerical Analysis", John Wiley, (1964).

* SAFTY1 *

(1) PURPOSE.

This module calculates the reactivity generated by the safety system and initiates the SCRAM action.

(2) PARAMETERS.

FN The function name used in computing SCRAM reactivity.

(3) COMMON VARIABLES.

<u>NAMES</u>	<u>COMBLK</u>	<u>DESCRIPTION</u>
JTIM	TIMER	Time step counter.
RKSF	SAFTY	Reactivity generated by the safety system.
TIME	TIMER	Current simulation time, in s.
TSCRAM	SAFTY	Time at which reactor SCRAM is activated, in s.
*		Any other COMMON variable from other data blocks upon which SCRAM can be activated.

(4) LOCAL VARIABLES.

LSCRAM A logical variable indicating whether the SCRAM has been already activated.

(5) OTHER SUBROUTINES OR MODULES REQUIRED.

Common Blocks: CONST5, SAFTY5, TIMER5 and any other COMMON block in which a variable activating the SCRAM can be found.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

Each plant parameter F that has a value F_{max} upon which the plant safety system will shut down the reactor by a SCRAM action will be tested by,

$$F \geq F_{max} \quad .$$

If the variable F is greater than F_{max} , the SCRAM will be activated.

(7) ORGANIZATION OF MODULE.

The user has to define by SCRAMF the plant parameters that can activate the safety system SCRAM action and the maximum value of the parameters that will activate the SCRAM. The user should also define a function FNCTii that gives the SCRAM reactivity as a function of time including the initial delays.

Upon first entry, TSCRAM is set to a large value. Statements testing for SCRAM action are inserted by the system for every parameter defined by SCRAMF. Testing is not performed once the SCRAM has been activated. If a SCRAM takes place, the negative reactivity FKSF is evaluated for the appropriate time after TSCRAM.

(8) APPLICABILITY AND RESTRICTIONS.

To use this module in the simulation, the following statement should be included.

- SAFTY1(ii);

The parameter ii is a two character name of the SCRAM reactivity function defined previously in the declarations segment by,

- FNCTii(K,N,TIME,.... RO,....);

The system variables that can activate the SCRAM have to be defined by,

- SCRAMF(F,Fmax,COMBL);

where F and F_{max} are the appropriate variable and the maximum value activating the SCRAM, and COMBL is the name of the COMMON block in which they can be found. F_{max} can be a variable, a number, or an algebraic expression. Any number of SCRAMFs can be defined in the simulation. If a COMBL has been included in a previous SCRAMF definition,

it must not be included again. Instead, an asterisk "*" should be placed for the third parameter.

```

*****
* SIMPS1 *   (ICCF)
*****

```

(1) PURPOSE.

This module performs third order Simpson integration.

(2) PARAMETERS.

LOOP Indicator of the loop on which this integration method is used.

(3) COMMON VARIABLES.

<u>NAMES</u>	<u>COMBLK</u>	<u>DESCRIPTION</u>
DELT	TIMER	Current integration time step, in s.
DYDT(I)	DERIV	Vector of the derivatives to be integrated, integrator input.
DYDT1(I)	DERIV	Intermediate or past derivative vector.
DYDT2(I)	DERIV	Intermediate or past derivative vector.
EYYYY1(I)	DERIV	Maximum relative integration error allowed, I=1 to NODER.
EYYYY2(I)	DERIV	Maximum absolute integration error allowed, I=1 to NODER.
EYYYY3(I)	DERIV	Evaluated integration error for the Ith integrated variable, I=1 to NODER.
IDIV	TIMER	If set to one, indicates one or more time step halving by an integration routine.
IDUB	TIMER	If set to one, indicates that the integration requests doubling of the time step.
ITERM	TIMER	If less than one, indicates that a simulation execution time error has been encountered.

IVYYY(I)	DERIV	Integration indicator. If not equal to LOOP, no integration will be performed.
LETH	TIMFR	If one, use fixed time step. If two, use variable time step.
NODER	DERIV	Number of integrations to be performed in the simulation.
YYYY(I)	DERIV	Vector of the integrated variables, integrator output.
YYYY1(J)	DERIV	Intermediate or past integration results.

(4) LOCAL VARIABLES.

DELTH Half time step, DELT/2.

ERMESL The text "IN SIMPS".

ERR Integration error.

(5) OTHER SUBROUTINES OR MODULES REQUIRED.

Subroutines: DERIV1.

Common Blocks: CONST5, DERIV5, TIMER5.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

The third order Simpson predictor corrector algorithm is used in this module. The predictor is computed by,

$$YYYY(I) = YYYYY1(I) + DELTH * DYDT1(I) \quad , \quad (1)$$

where the vectors with the suffix 1 are values obtained at the last time step or last iteration. This predictor leads to a derivative DYDT2 and then the corrector is computed from,

$$YYYY(I) = YYYYY(I) + DELTH * (DYDT(I) * 4 - DYDT1 * 3) \quad . \quad (2)$$

Finally, the value of the integrated variable vector is computed from,

$$YYYY(I) = YYYYY1(I) + DELT/6 * (DYDT1(I) + 4DYDT2(I) + DYDT(I)) \quad . \quad (3)$$

(7) ORGANIZATION OF MODULE.

Upon entering the subroutine, the vectors YYY1 and DYDT1 are set to the last computed value of the variables and their derivatives. The half step Euler integration [1] is performed, time is advanced by half a time step, and the derivatives DYDT2 at mid interval are evaluated.

Next, the corrector, Eq. (2), is computed by advancing the time by another half step and the new derivative DYDT is recalculated. Using Eq. (3), the integral is then evaluated. The integration error EYYY3 is computed from the absolute value of the difference between the second and third integration steps. The relative integration error is then obtained from the ratio of the integration error to the absolute allowed integration error,

$$ERR = EYYY3 / (YYY(I) * EYYY1(I) + EYYY2(I)) \quad . \quad (4)$$

If ERR is greater than one, then the time step is halved and the integration step is repeated. If ERR is less than 0.005, a request for doubling the next integration interval is made by setting IDUR to one. The size of the next time step is not changed if ERR is less than one and greater than 0.005.

(8) APPLICABILITY AND RESTRICTIONS.

This algorithm can be used with both fixed and variable time steps. The algorithm requires three evaluations of the derivative vector.

REFERENCES.

- [1] P. Henrici, "Elements of Numerical Analysis", John Wiley, (1964).

* SOCCN1 * (TEC,COS)

(1) PURPOSE.

This subroutine calculates the thermal conductivity of liquid sodium.

(2) PARAMETERS.

COS Sodium conductivity, in W/m-C.

TEC Sodium temperature, in C.

(3) COMMON VARIABLES.

None.

(4) LOCAL VARIABLES.

None.

(5) OTHER SUBROUTINES OR MODULES REQUIRED.

None.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

The thermal conductivity for liquid sodium is given in [1] by the equation,

$$COS = 90.686 - 0.0486 * TEC \quad (1)$$

(7) ORGANIZATION OF MODULE.

None.

(8) APPLICABILITY AND RESTRICTIONS.

The conductivity equation is applicable over the temperature range between 100 and 550 degrees Celsius. To use this module in the simulation, the following statement should be included.

CAIL SCON1(TEC,COS)

REFERENCES.

- [1] G. H. Golden and J. V. Tokar, "Thermodynamic Properties of Sodium", ANL 7323, (1967) .

* SODCP1 * (TEC,CPS)

(1) PURPOSE.

This subroutine calculates the specific heat of sodium.

(2) PARAMETERS.

CPS Sodium specific heat, in J/kg-C.

TEC Sodium temperature, in C.

(3) COMMON VARIABLES.

None.

(4) LOCAL VARIABLES.

None.

(5) OTHER SUBROUTINES OR MODULES REQUIRED.

None.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

The sodium specific heat is given in [1] by the equation,

$$CPS = 1437 - 1.0862 * TEC + 0.0004628 * TEC * TEC \quad (1)$$

(7) ORGANIZATION OF MODULE.

None.

(8) APPLICABILITY AND RESTRICTIONS.

The specific heat equation is applicable over the range between 100 and 680 degrees Celsius and represents the best fit for the liquid sodium viscosity between the melting point and 1370 degrees Celsius. To use this module in the simulation, the following statement should be included.

CAII SODCP1(TFC,COS)

REFERENCES.

- [1] G. H. Golder and J. V. Tokar, "Thermodynamic Properties of Sodium", ANL 7323, (1967).

* SODEN1 * (TEC,POC)

(1) PURPOSE.

This subroutine calculates the density of sodium given the temperature.

(2) PARAMETERS.

POC Sodium density, in kg/m³.

TEC Sodium temperature, in C.

(3) COMMON VARIABLES.

None.

(4) LOCAL VARIABLES.

None.

(5) OTHER SUBROUTINES OR MODULES REQUIRED.

None.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

The sodium density for a wide range of temperatures is given in [1] by the equation,

$$POC = 927.0 - 0.238 * TEC \quad . \quad (1)$$

(7) ORGANIZATION OF MODULE.

None.

(8) APPLICABILITY AND RESTRICTIONS.

The density equation is applicable over the range between 100 and 1450 degrees Celsius. To use this module in the simulation, the following statement should be included.

CALL SODEN1(TEC,ROC)

REFERENCES.

- [1] G. H. Golden and J. V. Tokar, "Thermodynamic Properties of Sodium", ANL 7323, (1967).

```

*****
* SODMU1 *   (TEC,SMU)
*****

```

(1) PURPOSE.

This subroutine calculates the dynamic viscosity of sodium given the temperature.

(2) PARAMETERS.

SMU Sodium dynamic viscosity, in kg/sec-m.

TEC Sodium temperature, in C.

(3) COMMON VARIABLES.

None.

(4) LOCAL VARIABLES.

ASMU LOG10 of SMU.

T Temperature in degrees Kelvin.

(5) OTHER SUBROUTINES OR MODULES REQUIRED.

None.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

The sodium dynamic viscosity for a wide range of temperatures is given in [1] as,

$$ASMU = .5108 + 220.65/T - 0.4925 * \text{DLOG10}(T) \quad (1)$$

and,

$$SMU = 0.001 * 10^{**ASMU} \quad (2)$$

where T is equal to TEC+273.

(7) ORGANIZATION OF MODULE.

None.

(8) APPLICABILITY AND RESTRICTIONS.

The above equations were obtained by Golden and Tokar as a best fit to several experimental and theoretical data and is the best representation for the liquid sodium viscosity between the melting point and 1370 degrees Celsius. To use this module in the simulation, the following statement should be included.

CAL1 SODMU1(TEC,SMU)

REFERENCES.

- [1] G. H. Golden and J. V. Tokar, "Thermodynamic Properties of Sodium", ANI 7323, (1967) .

* SOHXP1 * (U,ZIF,ZPF,RI,RX,D,CK,P,PD)

(1) PURPOSE.

This subroutine calculates the tube to shell side heat transfer coefficient in a sodium heat exchanger.

(2) PARAMETERS.

CK Tube conductivity, in W/m-C

D Hydraulic diameter of the equivalent channel on IHX shell side, in m.

PD F/T pitch to diameter ratio.

RI IHX tube internal radius, in m.

RX IHX tube external diameter, in m.

U Tube to primary (or external) coolant heat transfer coefficient, in W/m²-C.

ZPF Prandtl number (Dimensionless).

ZPE Reynolds number (Dimensionless).

(3) COMMON VARIABLES.

None.

(4) LOCAL VARIABLES.

ZNU1 Same as ZNU calculated when $60 \leq ZPE \leq 500$.

(5) OTHER SUBROUTINES OR MODULES REQUIRED.

Functions: UPEX1.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

The heat transfer correlation for the IHX is in [1] and [2]. For Peclet numbers where $ZPE < 60$, the following equation is used.

$$ZNU = 3.31 + 4.29 * PD + 0.888 * PD ** 2 \quad (1)$$

For Peclet numbers where $ZPE > 500$, the following equation is used.

$$ZNU = 6.66 + 3.126 * PD + 1.184 * PD ** 2 + \\ 0.0155 * (PS * ZRE * ZPE) ** 0.86 \quad (2)$$

PS is given by,

$$PS = 10.D - 10 \quad \text{if } EPS \leq 0 \text{ or} \\ PS = 1 - 1.82 / (ZPR * EPS ** 1.4) \quad \text{if } EPS > 0 \quad (3)$$

If $PS \leq 0$, $PS = 1.D - 10$,

and EPS is given by,

$$EPS = \exp(-6.4612 + 0.94182 * DLOG(ZRE) + \\ 1.3 * DLOG(1.38 / PD)) \quad (4)$$

For Peclet numbers where $60 \leq ZPE \leq 500$, the maximum of either Eq. (2) or (5) is taken and Eq. (5) is given by,

$$ZNU = -2.79 + 3.97 * PD + 1.025 * PD ** 2 + \\ (3.12 - 0.265 * DLOG(ZPE)) * DLOG(ZRE) \quad (5)$$

The heat transfer coefficient U is obtained by using either of the computed ZNU in,

$$U = 1.0 / (D / (ZNU * CK) + RX / CK * DLOG((2 * RX) / (RI + RX))) \quad (6)$$

(7) ORGANIZATION OF MODULE.

Three different heat transfer correlations can be computed according to the value of Peclet number. The ZPE value is therefore tested and the evaluation of the appropriate equation is made.

(8) APPLICABILITY AND RESTRICTIONS.

These heat transfer correlations are widely used in the literature and provide good results when the IHX primary side is heated with liquid metal. If the user wishes to use a different heat transfer correlation, he can produce his own subroutine by using the same name and locating it in the F program segment. To use this module in the simulation, the following statement should be included.

CALL SOHXP1(U,ZPE,ZPR,RI,RX,D,CK,P,PD)

REFERENCES.

- [1] M. W. Maresca and O. E. DWYER, "Heat Transfer to Mercury Flowing in Line Through a Bundle of Circular Tubes", J. Heat Transfer, 86, pp 180, (1969).
- [2] W. M. Kays and A. L. London, "Compact Heat Exchangers", McGraw Hill, (1964).

* SOHXS1 * (U,ZPF,ZPR,FI,RX,D,CK,P,PD)

(1) PURPOSE.

This subroutine calculates the sodium IHX tube to secondary coolant (or in tube flow) heat transfer coefficient.

(2) PARAMETERS.

CK Tube conductivity, in W/m-C.
D Hydraulic diameter of the equivalent channel on IHX shell side, in m.
PD P/D pitch to diameter ratio.
FI IHX tube internal radius, in m.
RX IHX tube external radius, in m.
U IHX tube to inner channel heat transfer coefficient, in W/m²-C.
ZPR Prandtl number (Dimensionless).
ZRE Reynolds number (Dimensionless).

(3) COMMON VARIABLES.

None.

(4) LOCAL VARIABLES.

None.

(5) OTHER SUBROUTINES OR MODULES REQUIRED.

None.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

The gap coefficient for a circular channel is given by,

$$H = (5 + 0.025 * ZPF^{**0.8}) * CK / (2 * FI) \quad . \quad (1)$$

The heat transfer coefficient for a circular channel is given by,

$$U = 1.0 / (1/H + FI/CK * DLOG((RX + RI)/(2 * RI))) \quad . \quad (2)$$

(7) ORGANIZATION OF MODULE.

Upon entering the subroutine, Eqs. (1) and (2) are solved.

(8) APPLICABILITY AND RESTRICTIONS.

To use this module in the simulation, the following statement should be included.

CALL SOHXS1(U,ZRF,ZPR,RI,RX,D,CK,P,PD)

REFERENCES.

- [1] E. W. Maresca and O. E. DWYER, "Heat Transfer to Mercury Flowing in Line Through a Bundle of Circular Tubes", J. Heat Transfer, 86, pp 180, (1969).
- [2] W. M. Kays and A. I. London, "Compact Heat Exchangers", McGraw Hill, (1964).

* SSOCN1 * (TEM,CON)

(1) PURPOSE.

This subroutine calculates the thermal conductivity of stainless steel.

(2) PARAMETERS.

CON Stainless steel conductivity, in W/m-C.

TEM Stainless steel temperature, in C.

(3) COMMON VARIABLES.

None.

(4) LOCAL VARIABLES.

None.

(5) OTHER SUBROUTINES OR MODULES REQUIRED.

None.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

The thermal conductivity of stainless steel is given by,

$$CON = 44.6944 - 0.0171342 * TEM \quad (1)$$

(7) ORGANIZATION OF MODULE.

Upon entering the subroutine, Eq. (1) is solved.

(8) APPLICABILITY AND RESTRICTIONS.

To use this module in the simulation, the following statement should be included.

CALL SCON1(TEM,CON)

* SSDCP1 * (TFM,CP)

(1) PURPOSE.

This subroutine calculates the specific heat of stainless steel.

(2) PARAMETERS.

CP Stainless steel specific heat, in J/kg-C.

TFM Stainless steel temperature, in C.

(3) COMMON VARIABLES.

None.

(4) LOCAL VARIABLES.

None.

(5) OTHER SUBROUTINES OR MODULES REQUIRED.

None.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

The specific heat of stainless steel is given by,

$$CP = 460 \quad . \quad (1)$$

(7) ORGANIZATION OF MODULE.

Upon entering the subroutine, Eq. (1) is solved.

(8) APPLICABILITY AND RESTRICTIONS.

The specific heat equation is assumed to be a constant over the range of present applications. To use this module in the simulation, the following statement should be included.

CALL SSDCP1(TEM,CP)

* SSDEN1 * (TEC,ROC)

(1) PURPOSE.

This subroutine calculates the specific density of stainless steel.

(2) PARAMETERS.

ROC Stainless steel specific density, in kg/m^3 .

TEC Stainless steel temperature, in C.

(3) COMMON VARIABLES.

None.

(4) LOCAL VARIABLES.

None.

(5) OTHER SUBROUTINES OR MODULES REQUIRED.

None.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

The specific density of stainless steel is given by,

$$\text{ROC} = 7812.0 \quad (1)$$

(7) ORGANIZATION OF MODULE.

Upon entering the subroutine, Eq. (1) is solved.

(8) APPLICABILITY AND RESTRICTIONS.

The specific density equation is assumed to be a constant over the range of present applications. To use this module in the simulation, the following statement should be included.

CALL SSDEN1(TEC,ROC)

* STCCN1 * (TFMF,CC,RC)

(1) PURPOSE.

This subroutine calculates the thermal conductivity of steam.

(2) PARAMETERS.

CO Steam conductivity, in W/m-C.

RC Steam density, in kg/m³.

TEMP Steam temperature, in C.

(3) COMMON VARIABLES.

None.

(4) LOCAL VARIABLES.

See WACON1.

(5) OTHER SUBROUTINES OR MODULES REQUIRED.

Subroutines: WACON1.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

See WACCN1.

(7) ORGANIZATION OF MODULE.

This module, when called, causes the inclusion of the WACON1 module. STCON1 is an entry to WACON1. See WACON1.

(8) APPLICABILITY AND RESTRICTIONS.

This subroutine is valid for temperatures between 0 and 700 C., and pressures between 0 and 5E7 Pa. To use this module in the simulation, the following statement should be included.

CALL STCON1(TEMP,CO,RO)

* STDCP1 * (TEC,CPS,P)

(1) PURPOSE.

This subroutine calculates the specific heat of steam.

(2) PARAMETERS.

CPS Specific heat of the water, in J/kg-C.

P Pressure, in Pa.

TEC Temperature of the water, in C.

(3) COMMON VARIABLES.

None.

(4) LOCAL VARIABLES.

F Helmholtz Function.

F1 First derivative of Helmholtz Function w/r to temperature (constant specific volume).

F2 First derivative of Helmholtz Function w/r to specific volume (constant temperature).

F11 Second derivative of Helmholtz Function w/r to temperature (constant specific volume).

F12 Second derivative of Helmholtz Function w/r to temperature and specific volume.

F22 Second derivative of Helmholtz Function w/r to specific volume (constant temperature).

K A constant indicating steam (100) or water (200) on first call to WASPT1.

KW An indicator to the first entry to the subroutine and initially set to .FALSE. .

TOUT Equal to TEC.

V Specific volume, in m³/kg.

(5) CITER SUBROUTINES CF MODULES REQUIRED.

Subroutines: WASPT1.
Common Blocks: CONST5.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

The specific heat of steam is computed from the absolute temperature and the appropriate Helmholtz derivatives by,

$$CPS = (TCUT + 273) * (F12 * F12 - F11) / F22 \quad . \quad (1)$$

(7) ORGANIZATION CF MODULE.

The first entry to the subroutine will set K to 100 and V to -1.0 indicating the first calculation of the steam properties by WASPT1. Subsequent entries will use the previous values of V as their initial guess. Each entry will use WASPT1 to compute the Helmholtz derivatives and to solve Eq. (1).

(8) APPLICABILITY AND RESTRICTIONS.

The specific heat equation is applicable over the range of temperatures between 0.01 and 374.15 degrees Celsius. To use this module in the simulation, the following statement should be included.

CALL STDCP1(TEC,CPS,P)

* STDEN1 * (TEC, FOC, P)

(1) PURPOSE.

This subroutine calculates the density of steam.

(2) PARAMETERS.

P Pressure, in Pa.
RCC Density of the steam, in kg/m³.
TEC Temperature of the steam, in C.

(3) COMMON VARIABLES.

None.

(4) LOCAL VARIABLES.

F Helmholtz Function.
F1 First derivative of Helmholtz Function w/r to
 temperature (constant specific volume).
F2 First derivative of Helmholtz Function w/r to
 specific volume (constant temperature).
F11 Second derivative of Helmholtz Function w/r to
 temperature (constant specific volume).
F12 Second derivative of Helmholtz Function w/r to
 temperature and specific volume.
F22 Second derivative of Helmholtz Function w/r to
 specific volume (constant temperature).
K A constant indicating steam (100) or water (200) on
 first call to WASPT1.
TCUT Equal to TEC.
V Specific volume, in m³/kg.

(5) OTHER SUBROUTINES OR MODULES REQUIRED.

Subroutines: WASPT1.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

The density of steam is calculated from the specific volume obtained from the water steam property function WASPT1 by $RCC=1/V$.

(7) ORGANIZATION OF MODULE.

This module, when called, will cause the inclusion of the WADEN1 module. see details in WADEN1.

(8) APPLICABILITY AND RESTRICTIONS.

The specific density evaluation is applicable over the temperature range of 0.01 to 374.15 degrees Celsius. To use this module in the simulation, the following statement should be included.

CALL STDEN1(TEC,ROC,P)

* STD MU1 * (VS,TEMP,RC)

(1) PURPOSE.

This subroutine calculates the dynamic viscosity of steam given the temperature and density.

(2) PARAMETERS.

RC Density of the steam, in kg/m^3 .

TEMP Temperature of the steam, in C.

VS Dynamic viscosity of the steam, in kg/sec-m .

(3) COMMON VARIABLES.

See WADMU1

(4) LOCAL VARIABLES.

See WADMU1.

(5) OTHER SUBROUTINES OR MODULES REQUIRED.

See WADMU1.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

The dynamic viscosity of steam for a wide range of temperatures is given by the equations shown in WADMU1.

(7) ORGANIZATION OF MODULE.

A call to this subroutine or an attempt to include this module will cause the WADMU1 module to be included in the simulation.

(8) APPLICABILITY AND RESTRICTIONS.

This subroutine is valid for temperatures between 0 and 700 C. and pressures between 0 and 8E7 Pa. To use this module in the simulation, the following statement should be included.

CALL STD MU1(VS,TEMP,PO)

```
*****
* STEPR1 *   (ZZ,ZZP)
*****
```

(1) PURPOSE.

This module produces five functions, namely,

STEP(ZZ) A step function,

RSTEP(ZZ) An inverse step function,

RAMP(ZZ) A ramp function,

PULSE(ZZ,ZZP) A pulse of width ZZP,

and

RPULSE(ZZ,ZZP) An inverse pulse of width ZZP.

(2) PARAMETERS.

ZZ Time at which the function is activated, in s.

ZZP Width of the pulse, in s.

(3) COMMON VARIABLES.

<u>NAMES</u>	<u>COMBLK DESCRIPTION</u>
TIME	TIMER Current simulation time, in s.

(4) LOCAL VARIABLES.

PULSE Value of the pulse function.

RAMP Value of the ramp function.

RPULSE Value of the inverse pulse function.

RSTEP Value of the inverse step function.

STEP Value of the step function.

(5) OTHER SUBROUTINES OR MODULES REQUIRED.

Common Blocks: CONST5, TIMFR5.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

The step function is generated by the following statements.

TIME<ZZ STFP=0 (1)

TIME≥ZZ STFP=1 (2)

The RSTFP function is generated by the following statements.

TIME<ZZ RSTFP=1 (3)

TIME≥ZZ RSTFP=0 (4)

The RAMP function is generated by the following statements.

TIME<ZZ RAMP=0 (5)

TIME≥ZZ RAMP=TIME-ZZ (6)

The PULSE function is generated by the following statements.

ZZ+ZZP<TIME<ZZ PULSE=0 (7)

ZZ+ZZP≥TIME≥ZZ PULSE=1 (8)

The RPULSE function is generated by the following statements.

ZZ+ZZP<TIME<ZZ RPULSE=1 (9)

ZZ+ZZP≥TIME≥ZZ RPULSE=0 (10)

(7) ORGANIZATION OF MODULE.

An IF statement is used to check relations (1) and (2) for each function.

(8) APPLICABILITY AND RESTRICTIONS.

The STEPR1 functions are always present in the simulation.

```

*****
* STHXP1 * (U,ZFF,ZER,FI,RX,D,CK,P,PD)
*****

```

(1) PURPOSE.

This subroutine calculates the tube to shell side heat transfer coefficient in a sodium heated steam generator superheater section.

(2) PARAMETERS.

CK Conductivity of the tube, in W/m-C

D Hydraulic diameter of the equivalent channel on IHX shell side, in m.

P The steam pressure, in Pa.

PD P/D pitch to diameter ratio.

RI IHX tube internal radius, in m.

RX IHX tube external diameter, in m.

U Tube to primary (or external) coolant heat transfer coefficient, in W/m²-C.

ZFF Prandtl number (Dimensionless).

ZRF Reynolds number (Dimensionless).

(3) COMMON VARIABLES.

None.

(4) LOCAL VARIABLES.

ZNU1 Same as ZNU, calculated when $60 \leq ZPE \leq 500$.

(5) OTHER SUBROUTINES OR MODULES REQUIRED.

Functions: UPIEX1.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

The heat transfer correlation for the IHX is in [1] and [2]. For Peclet numbers where $ZPE < 60$, the following equation is used.

$$ZNU = 3.31 + 4.29 * PD + 0.888 * PD ** 2 \quad . \quad (1)$$

For Peclet numbers where $ZPE > 500$, the following equation is used.

$$ZNU = 6.66 + 3.126 * PD + 1.184 * PD ** 2 + \\ 0.0155 * (PS * ZRE * ZPE) ** 0.86 \quad (2)$$

PS is given by,

$$PS = 10.D - 10 \quad \text{if } EPS \leq 0 \text{ or} \\ PS = 1 - 1.82 / (ZPE * EPS ** 1.4) \quad \text{if } EPS > 0 \quad . \quad (3)$$

If $PS \leq 0$, $PS = 1.D - 10$,

EPS is given by,

$$EPS = EXP(-6.4612 + 0.94182 * DLOG(ZPE) + \\ 1.3 * DLOG(1.38 / PD)) \quad . \quad (4)$$

For Peclet numbers where $60 \leq ZPE \leq 500$, the maximum of Eq. (2) or (5) is taken and Eq. (5) is given by,

$$ZNU = -2.79 + 3.97 * PD + 1.025 * PD ** 2 + \\ (3.12 - 0.265 * DLOG(ZPE)) * DLOG(ZPE) \quad . \quad (5)$$

The heat transfer coefficient U is obtained by using either of the computed ZNU as,

$$U = 1.0 / (D / (ZNU * CK) + RX / CK * DLOG((2 * RX) / (RI + RX))) \quad . \quad (6)$$

(7) ORGANIZATION OF MODULE.

Three different heat transfer correlations can be computed according to the value of the Peclet number. The ZPE value is therefore tested and the evaluation of the appropriate equation is made.

(8) APPLICABILITY AND RESTRICTIONS.

These heat transfer correlations are widely used in the literature for the simulation of sodium heat exchangers. If the user wishes to use a different heat transfer correlation, he can write his own subroutine, use the same name and parameters, and store it in the F program segment. To use this module in the simulation, the following statement should be included.

CALL STHXP1(U,ZFE,ZPR,RI,RX,CF,D,P,PD)

REFERENCES.

- [1] M. W. Maresca and O. E. DWYER, "Heat Transfer to Mercury Flowing in Line Through a Bundle of Circular Tubes", J. Heat Transfer, 86, pp 180, (1969).
- [2] W. M. Kays and A. I. London, "Compact Heat Exchangers", McGraw Hill, (1964).


```

*****
* STHXS1 * (U,ZPF,ZPR,RI,RX,D,CK,P,PD)
*****

```

(1) PURPOSE.

This subroutine calculates the tube to steam (or in tube flow), heat transfer coefficient in the superheater part of a sodium cooled heat exchanger.

(2) PARAMETERS.

CK Tube conductivity, in W/m-C.

D Hydraulic diameter, in m.

P The steam pressure, in Pa.

PD Dummy parameter.

PI IHX tube internal radius, in m.

RX IHX tube external radius, in m.

U Tube to inner channel heat transfer coefficient, in W/m²-C.

ZPR Prandtl number (Dimensionless).

ZRE Reynolds number (Dimensionless).

(3) COMMON VARIABLES.

None.

(4) LOCAL VARIABLES.

None.

(5) OTHER SUBROUTINES OR MODULES REQUIRED.

Functions: UPIIN1.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

The Nusselt number for pressures below $10E7$ Pa is given by,

$$ZNU=0.0133*ZRE**0.84*ZPR**0.333 \quad . \quad (1)$$

For pressures above $10E7$ Pa, the equation is,

$$ZNU=0.0073*ZRF**0.866*ZPR**0.61 \quad . \quad (2)$$

(7) ORGANIZATION OF MODULE.

Upon entering the subroutine, a check is performed on the magnitude of the steam pressure P to decide whether Eq. (1) or (2) should be used. The heat transfer coefficient U is then computed by the function $UPIIN1$.

(8) APPLICABILITY AND RESTRICTIONS.

These heat transfer correlations are widely used in the literature for the simulation of sodium heat exchangers. If the user wishes to use a different heat transfer correlation, he can write his own subroutine, use the same name and parameters, and store it in the F program segment. To use this module in the simulation, the following statement should be included.

CALL STXK1(U,ZPE,ZPR,RI,RX,CK,D,P,PD)

REFERENCES.

- [1] M. W. Maresca and O. E. DWYER, "Heat Transfer to Mercury Flowing in Line Through a Bundle of Circular Tubes", J. Heat Transfer, 86, pp 180, (1969).
- [2] W. M. Kays and A. L. London, "Compact Heat Exchangers", McGraw Hill, (1964).

```

*****
* STIFF1 * (ICCP)
*****

```

(1) PURPOSE.

This module gives a method to integrate differential equations with widely spread Eigenvalues, that is, STIFF equations.

(2) PARAMETERS.

LOOP Indicator of the loop on which this integration method is used.

(3) COMMON VARIABLES.

<u>NAMES</u>	<u>COMB1K</u>	<u>DESCRIPTION</u>
DELT	TIMER	Current integration time step, in s.
DTMAX	TIMER	Maximum time step allowed, in s.
DTMIN	TIMER	Minimum time step allowed, in s.
DYDT(I)	DERIV	Vector of the derivatives to be integrated, integrator input.
DYDT1(I)	DERIV	Intermediate or past derivative vector.
DYDT2(I)	DERIV	Intermediate or past derivative vector.
DYDT3(I)	DERIV	Intermediate or past derivative vector.
DYDT4(I)	DERIV	Intermediate or past derivative vector.
DYDT5(I)	DERIV	Intermediate or past derivative vector.
EYYYY1(I)	DERIV	Maximum relative integration error allowed, I=1 to NODER.
EYYYY2(I)	DERIV	Maximum absolute integration error allowed, I=1 to NODER.

EYYYY3(I)	DERIV	Evaluated integration error for the Ith integrated variable, I=1 to NODER.
IDIV	TIMER	If set to one, indicates one or more time step halving by an integration routine.
ICUB	TIMER	If set to one, indicates that the integration requests doubling of the time step.
ITERM	TIMER	If less than one, indicates that a simulation execution time error has been encountered.
IYYYY(I)	DERIV	Integration indicator. If not equal to LOOP, no integration will be performed.
NODER	DERIV	Number of integrations to be performed in the simulation.
YYYY(I)	DERIV	Vector of the integrated variables, integrator output.
YYYY2(I)	DERIV	Intermediate or past integration results.
YYYY3(I)	DERIV	Intermediate or past integration results.
YYYY4(I)	DERIV	Intermediate or past integration results.
YYYY5(I)	DERIV	Intermediate or past integration results.

(4) ICCAI VARIABLES.

AWH	The value of $WH/2+1$.
DELT1	A temporary storage for DELT at entry time.
DT4	The value of $DELT1/4$.
DY1	The value of $DYDT-DYDT1$.
DY4	The value of $DYDT4(I) * SDEL$
D4T	The Euler integration time step, initially is set to $DELT/4$, and upon each entry readjusted to the smaller of $DELT/4$ or the previous D4t.

ERMESL The text "IN STIFF".
 ERR Relative integration error.
 EWJ The value of WH+1.
 JENTRY Indicates the first or subsequent entries to the subroutine and initially is set to zero.
 JF Indicator to the integration step performed. Set to zero upon first entry. Set to two after an Euler integration step. Set to three after a time step is halved. Set to one upon successful completion of a time step.
 JP1 Counter of the integration time steps.
 JP2 Indicator set to one if all of the errors in the Euler integration step were less than 0.005.
 JP3 Indicator that, if positive, allows the next Euler integration time step to be increased. Initially is set to -6, and is reset to -6 whenever a time step is changed. Incremented by one after each successful integration step.
 JP4 Will be set to one if at least one value of the integration error is less than 0.005.
 JP5 Same as JP3 but for the main integration step.
 JP6 Counts the number of times during which the Euler integration error was less than 0.005.
 RDEL Temporary storage for D4T.
 RDEL1 Temporary storage for DELT1.
 RDEL2 The value of 2*D4T.
 SDDEL Temporary storage for D4T.
 SDEL Temporary storage for SDDEL.
 TIME1 Temporary storage for TIME.
 WH The value of W1*RDEL1.
 W1 The value of YYYY3(I).

(5) OTHER SUBROUTINES OF MODULES REQUIRED.

Subroutines: DERIV1.

Common Blocks: CONST5, DERIV5, TIMEP5.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

The algorithm used for solving STIFF equations in DSNP is the same as that used in CSMP III [1]. It is recommended that this algorithm be used whenever a stiff equation system is suspected to exist in a simulation. The notation used below is h equals the integration time step, q equals an intermediate Euler integration step initially set to $h/4$, d indicates a derivative with respect to time and dd indicates a second derivative with respect to time. The algorithm used in STIFF1 is given by,

$$Y(t+h) = Y(t) + h * (dYA + a * ddYA) \quad , \quad (1)$$

where,

$$dYA = [Y(t) - Y(t-h)] \quad , \quad (2)$$

$$ddYA = dY(t) - dY(t-h) \quad , \quad (3)$$

$$YE(t+q) = Y(t) + q * dY(t) \quad , \quad (4)$$

(Euler integration with $q = h/4$)

$$ddYE(t) = [dY(t+q) - dY(t)] / q \quad , \quad (5)$$

$$I = ddYE / ddYA \text{ if } ddYA \leq 0 \text{ and } L = 0 \text{ if } ddYA = 0 \quad , \quad (6)$$

$$a = (\exp(I * h) - 1) / (L * h) \text{ if } L < 0 \text{ and } a = 1 + L * h / 2 \text{ if } L \geq 0 \quad , \quad (7)$$

$$b = \exp(I * h) \text{ if } L < 0 \text{ and } b = 1 + L * h \text{ if } L \geq 0 \quad . \quad (8)$$

The integration error is given by,

$$ER = h * [dY(t+h) - dYA(t) - b * ddYA] \quad , \quad (9)$$

and for the Euler intermediate integration step by,

$$ERE = 2 * q * [dYE(t+q) - dY(t)] \quad . \quad (10)$$

The magnitude of the next integration step is controlled by both integration errors.

(7) ORGANIZATION OF MODULE.

When the subroutine is first entered, all of the JP indicators are set to zero and the Euler integration step,

q , is set to $h/4$. The integration process on each subsequent entry will start by performing an Euler integration using the smaller of the new and previous q step interval, that is Eqs. (2), (3), and (4) are solved. Then, the first call to DERIV1 is made to obtain the vector of the derivatives at $t+q$. The integration error, ERE, for this step is computed by Eq. (10). If ERE is greater than 1.0, the time step q is halved, the variables and their derivatives are reset to their value at the beginning of the interval, and the Euler integration step is repeated. If all of the ERE are less than 0.005, the next integration step is doubled, or when q is equal to $2q$, if more than five integration steps were made since the last change in time step size. In addition, to allow time step doubling, $4q$ must be less than DTMAX and the number of executed time steps must be even.

Following the Euler or "perturbation" step, the "asymptotic" integration step is performed. First, the various coefficients, a , b , and L , are evaluated. As shown in Eqs. (5), (6), (7), and (8), the coefficients are related and depend strongly on the value of the derivative of each variable. The variables are then integrated according to Eq. (1) to obtain the integrated variables at time $t+h$ and the DERIV1 subroutine is called for the second time to obtain the derivatives at $t+h$.

Next, the full step integration error, ER, is computed using Eq. (9). The same convergence criteria as above are used to determine the size of the next time step except that h can not be doubled in the subroutine. Rather a request for doubling, or when IDUB is equal to one, is made indicating to the master time step controller that the time step can now be doubled if other criteria do not prevent such an operation. If at any time the time step h becomes less than DTMIN, the simulation will be terminated and an error message will be printed.

(8) APPLICABILITY AND RESTRICTIONS.

This is the most efficient method presently available in DSNP for the integration of stiff equations.

REFERENCES.

- [1] M. E. Fowler and R. M. Warten in "A Numerical Integration technique for Ordinary Differential Equations With Widely Separated Eigenvalues", IBM Journal of Research and Development, 11, 5, pp 537, (1967).

* TERMN1 *

(1) PURPOSE.

This module prints error messages and optionally terminates the simulation, if a simulation time error is detected.

(2) PARAMETERS.

None.

(3) COMMON VARIABLES.

<u>NAMES</u>	<u>COMBLK</u>	<u>DESCRIPTION</u>
ERRMES(I)	TIMER	A vector of words to store simulation error messages.
IC6	TIMER	Standard FORTRAN output unit (usually 6).
ITERM	TIMER	If less than one, indicates that a simulation execution time error has been encountered.
TIME	TIMER	Current simulation time, in s.
TIMEF	TIMER	Time at which the simulation is terminated, in s.

(4) LOCAL VARIABLES.

I	Integer value of TIME or other integers passed by ERRMES.
---	---

(5) OTHER SUBROUTINES OR MODULES REQUIRED.

Common Blocks: TIMERS.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

None.

(7) ORGANIZATION OF MODULE.

Several of the DSNP modules generate termination or warning code when ITERM is not equal to zero. The generated message is stored in the alphanumeric vector ERRMES(15). The subroutine will be entered from the main program if ITERM is greater than zero.

If a simulation termination is requested by a particular error, TIMEF is set to TIME and, upon returning to the main program, the simulation will be terminated.

When ITERM=1, the CPU time is about to run out. In this case, TIMEF is set to the nearest upper integer value of TIME and an appropriate message is printed.

When ITERM=2, the time step, DELT, was reduced below DTMIN and the simulation is terminated.

When ITERM=3, a timing error or a mismatch in time between the master time and the timing in one of the integration loops is indicated.

When ITERM=4, insufficient data is stored in a pipe function. The requested output value is not available.

(8) APPLICABILITY AND RESTRICTIONS.

This module produces simulation time messages to help the user to determine the cause of an abnormal simulation termination. There are many other simulation time messages that will be transformed into this module in the future.

```

*****
*  TIMER1  *
*****

```

(1) PURPOSE.

This module controls the execution of the dynamic part of the simulation program and the magnitude of the master time step.

(2) PARAMETERS.

None.

(3) COMMON VARIABLES.

<u>NAMES</u>	<u>COMBIK</u>	<u>DESCRIPTION</u>
DELT	TIMER	Current integration time step, in s.
DELTM	TIMER	Master simulation time step, in s. Integration output must fit this step.
ERRMES(I)	TIMER	A vector of words to store simulation error messages.
IDIV	TIMER	If set to one, indicates one or more time step halving by an integration routine.
IDUB	TIMER	If set to one, indicates that the integration requests doubling of the time step.
IISTEP(I)	DERIV	Number of subtime steps in the Ith loop to integrate from TIME to TIME+DELTM.
IISTPS(I)	DERIV	Number of time steps during which DELT=DELTM.
ITERM	TIMER	If less than one, indicates that a simulation execution time error has been encountered.
JTIM	TIMER	Time step counter.
LCOPS	DERIV	Number of loops in the simulation.

TCNTEL	TIMER	Time at which the change in external control reactivity is activated.
TIME	TIMER	Current simulation time, in s.
TIMEM	TIMER	Master simulation time, in s.
TERIN	TIMER	Time at which the next printing of the requested variables should take place, in s.
ZDELTL(I)	TIMER	The time step DELT of the Ith integration loop, in s.
ZTIME1(I)	TIMER	The recent time in the Ith integration loop, in s.

(4) LOCAL VARIABLES.

DELT1	Value of DELT prior to entering an integration subroutine.
IL	Counts the number of substeps executed in an integration loop to cover the range from TIMEM to TIMEM+DELT.
IIL	Halving indicator of the number of times the time step was halved in an integration subroutine or DELT1/DELT.
ILP	Number of DELT time steps required to integrate over the DELTM time interval.
J11	Number of time steps left before the next output interval.
KC	The difference in DTMIN units between the master time step, DELTM, and ZDELTX or (ZDELTX-DELTM)/dtmin.
KFIRST	Logical indicator of the first entry to the subroutine.
KLUP	Loop indicator in the master loop control algorithm.
KMU	DELT/DELT.
TIMENX	Next time interval to be matched precisely by the integration process. Set by the output or perturbation requirements.
ZDELTX	Contains the largest of the time steps used in

the different LOOPS.

ZDTM TIMFNX-TIMEM.

(5) OTHER SUBROUTINES OF MODULES REQUIRED.

Subroutines: DERIV1, and the integration subroutines for each loop.

Common Blocks: CONST5, DERIV5, TIMER5.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

This is the main timing control subroutine. It supervises the execution of the integration process. The algorithm to control the master time step is based on the following requirements:

- a) A master time step, DELTM, must exactly meet the next output or interrupt interval such that the number of time steps required to cover any fixed interval, JI1, must be an integer.
- b) The time step can not be greater than DTMAX or less than DTMIN.
- c) An attempt will be made to set DELTM to the largest requested time step in any of the integration loops, ZDELTX, provided it does not violate other restrictions.

(7) ORGANIZATION OF MODULE.

Upon first entry to the subroutine, when KFIRST=.FALSE., the various counters and indexing variables are reset. The master time, TIMFM, and the master time step, DELTM, are initialized, and the initial derivative of all the variables to be integrated are computed by calling DERIV1 once for each loop defined.

For the first ten time steps, the master integration interval is kept constant by setting I1STPS=10. Integration subintervals can, however, be decreased if so requested in the integration routine by the appropriate error criteria. Starting from the eleventh time step, each loop is tested and the largest of all the individual loop time steps, ZDELT(LOOP), is stored in ZDELTX. The last variable is then compared to the current DELTM. If DELTM is greater than ZDELTX, then DELTM is set equal to ZDELTX and the number of new time steps remaining until

the next output is appropriately increased. If DELTM is less than ZDELTX, the ratio KMU equal to ZDELTX divided by DELTM is first computed. Then, a check is made to find out whether ZDELTX will exactly match the remaining time until the next output. If matching is possible, DELTM is set equal to ZDELTX. Increasing DELTM is, however, limited to DEITMX. The number of new time steps remaining until the output is reset by JI1 equals JI1 divided by KMU.

If TIMENX has been met or JI1 is equal to zero, the next time interval to be matched is sought. It can be either the end of the simulation time, TIMEF, the next output, TPRIN, or the next perturbation time, TCNTRL, of some externally controlled event that has to take place at a precise time. If necessary, a new master time step is produced to exactly match the next TIMENX.

After the master time step has been adjusted, a DO loop having IOCPS repetitions is entered to control the integration of each loop defined by SIMULATE. The local time and time step for the particular LOOP are now determined. This time step can be either equal to or an integer fraction of DEITM. The internal time step counters are reset and the appropriate integration scheme is entered.

When control from the integration subroutine is returned to TIMER, a check is performed to find out whether the time step is decreased or whether an increase in the time step is required. If DELT is less than DELTM, an increase in DELT will be allowed constrained by the requirement of matching TIMEF plus DELTM at the end of the integration loop. The error indicator, ITEPM, will also be tested and, if so requested, the simulation will be terminated.

(8) APPLICABILITY AND RESTRICTIONS.

This module is always present in the simulation. The size of the loop control part depends upon the number of dynamic simulation loops present in the simulation.

* TPMUL1 * (X,P,R)

(1) PURPOSE.

This module calculates the two phase friction multiplier using the Lockhart-Martinelli correlation.

(2) PARAMETERS.

P Pressure, in Pa.

R Density, in kg/m³.

X The steam quality or gas weight fraction of the two-phase flow.

(3) COMMON VARIABLES.

None.

(4) LOCAL VARIABLES.

AL The value of $X \cdot V_{SS} / V_{AV}$.

DPT Derivative of pressure w/r to temperature.

GA The value of $(1-X)/(1-AL)$.

T Saturated temperature at pressure P, in C.

TPMUL1 Two phase friction multiplier.

VAV Average specific volume of the two phase fluid, in m³/kg.

VSS Specific volume of the saturated steam, in m³/kg.

VSW Specific volume of the saturated water, in m³/kg.

(5) OTHER SUBROUTINES OR MODULES REQUIRED.

Subroutines: WASAT1, WASPT1.

Common Blocks: CONST5.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

The two phase friction factor multiplier can be estimated from the following equation,

$$TPMUL1 = (1 + GA + GA * GA) / 3 \quad , \quad (1)$$

where GA is given above.

(7) ORGANIZATION OF MODULE.

Upon entering the subroutine, the saturation temperature for the pressure P is obtained by calling WASAT1. Then, the saturated steam and saturated water specific volumes are computed using subroutine WASPT1 and are used to compute the two phase friction multiplier using Eq. (1).

(8) APPLICABILITY AND RESTRICTIONS.

This subroutine uses a simple relationship to obtain the two phase friction multiplier used in computing the frictional pressure drop in pipe segments where two phase flow prevails. The Lochart-Martinelli correlation is used in the present model. Several other correlations are now available, however, and can be used to replace the present equation. To use this module in the simulation, the following statement should be included.

CALL TPMUL1(X,P,F)

* TPOWER1 *

(1) PURPOSE.

This module simulates the core power distribution.

(2) PARAMETERS.

None.

(3) COMMON VARIABLES.

<u>NAMES</u>	<u>COMBLK</u>	<u>DESCRIPTION</u>
JTIM	TIMER	Time step counter.
PNJ	NFUTR	Normalized neutronic power at the Jth time step.
PNO	NFUTR	Normalized initial neutronic power level.
PWJ	TPOWER	Thermal power generated by the reactor, in W.
PWO	TPOWER	Initial PWJ, in W.

(4) LOCAL VARIABLES.

WQT Conversion factor from normalized neutronic power
PNJ to thermal power PWJ, in W.

(5) OTHER SUBROUTINES OR MODULES REQUIRED.

Common Blocks: CONST5, NEUTR5, TIMER5, TPOWER5.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

At present, only a lumped parameter approximation to the core power distribution is used. As a result, the only operation performed by this module is the conversion of

the normalized neutronic power PNJ into thermal power PWJ by,

$$PWJ=WQT*PNJ \quad .$$

(7) ORGANIZATION OF MODULE.

During the IC calculations, the conversion factor from normalized neutronic power to thermal power is computed from the ratio of the two initial power values,

$$WQT=PW0/PN0 \quad .$$

Upon each entry during the dynamic part of the simulation, Eq. (1) is solved.

(8) APPLICABILITY AND RESTRICTIONS.

This subroutine only multiplies the neutronic power by a constant. To use this module in the simulation, the following statement should be included.

• TPOWER1;

* TPRIN1 *

(1) PURPOSE.

This subroutine performs the user requested output operations. It will print a column of output variables for every predetermined time interval.

(2) PARAMETERS.

None.

(3) COMMON VARIABLES.

All of the variables to be printed by the DSNP "P" statement must be in one of the system common blocks.

(4) LOCAL VARIABLES.

BYTO(J)	Printing interval information as defined by the DSNP BYTO statement. J is set to the total number interval segments.
IVEC	Total number of variables to be printed.
JP	Total number of storage operations.
FNAMES(2*IVFC)	Contains the names of the variables to be printed.
VECT(IVEC,10)	Stores up to ten values of the variables to be printed, and IVEC is the total number of these variables.

(5) OTHER SUBROUTINES OR MODULES REQUIRED.

Common Blocks: TIMER5, and all of the other common blocks which have variables that are to be printed.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

None.

(7) ORGANIZATION OF MODULE.

This subroutine is created by the precompiler as a result of the user defined "P" statements. The library module contains only a few general statements. The subroutine will first check whether TIMEM matches TPPIN. If the times do not match, control is returned to the MAIN program. If the times match, the requested variables to be printed are stored in the VECT vector and the JP counter is increased by one. If ten storage operations were performed or TIME equals TIMEF, the vector is printed, and a page of output containing information for up to ten time intervals will be produced.

The subroutine will also compute the next printing interval TPPIN. The first three variables printed in each column of output are TIMEM, DELTM, and JTIM. These are followed by the user requested output in the user defined order.

(8) APPLICABILITY AND RESTRICTIONS.

This subroutine produces an output table for ten time steps. It will take a page or more of the output, depending upon the number of variables for which time dependent output is requested. The variable names will all be printed on the left side of the page. Single variable vectors can also be printed and a standard title can be printed on each page. For further details, see the users manual.

```

*****
* TRAPZ1 *   (LCCP)
*****

```

(1) PURPOSE.

This subroutine performs integration of a vector of first order differential equations using the trapezoidal integration method with variable time steps limited by the integration error.

(2) PARAMETERS.

LOOP Indicator of the loop on which this integration method is used.

(3) COMMON VARIABLES.

<u>NAMES</u>	<u>COMBLK</u>	<u>DESCRIPTION</u>
DELT	TIMER	Current integration time step, in s.
DYDT(I)	DERIV	Vector of the derivatives to be integrated, integrator input.
DYDT1(I)	DERIV	Intermediate or past derivative vector.
EYYYY1(I)	DERIV	Maximum relative integration error allowed, I=1 to NODER.
EYYYY2(I)	DERIV	Maximum absolute integration error allowed, I=1 to NODER.
EYYYY3(I)	DERIV	Evaluated integration error for the Ith integrated variable, I=1 to NODER.
IDUB	TIMER	If set to one, indicates that the integration requests doubling of the time step.
IDIV	TIMER	If set to one, indicates one or more time step halving by an integration routine.
IYYYY(I)	DERIV	Integration indicator. If not equal to LOOP, no integration will be performed.

JTIM	TIMER	Time step counter.
NCDER	DERIV	Number of integrations to be performed in the simulation.
TIME	TIMER	Current simulation time, in s.
YYYY(I)	DERIV	Vector of the integrated variables, integrator output.
YYYY1(I)	DERIV	Intermediate or past integration results.

(4) LOCAL VARIABLES.

ERMESL The text "IN TRAPZ".

ERR Ratic of computed to allowed integration error.

(5) OTHER SUBROUTINES OF MODULES REQUIRED.

Subroutines: DERIV1.

Common Blocks: CONST5, DERIV5, TIMER5.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

A trapezoidal predictor corrector method is used. The predictor is given by,

$$YYYY(I) = YYYY1(I) + DELT * DYDT1(I) \quad , \quad (1)$$

and the corrector is given by,

$$YYYY(I) = YYYY1(I) + DELT/2 * (DYDT(I) + DYDT1(I)) \quad , \quad (2)$$

where YYYY1 and DYDT1 are the values at the beginning of the integration interval, while YYYY and DYDT are the values at the end of the integration interval.

(7) ORGANIZATION OF MODULE.

The variables to be integrated and their last derivative at TIME are stored in YYYY and DYDT, respectively. Upon entering the integration subroutine, these variables will be stored in the YYYY1 and DYDT1 vectors, respectively. A check of IYYYY(I) is performed for each vector member to see whether integration is required.

The predictor is calculated first using Eq. (1) and the value is stored in EYYYY3. The integration time is advanced by DELT and the DERIV1 subroutine is called to obtain the derivatives at TIME+DELT. The corrector YYYY is now calculated using Eq. (2). The absolute differences between the predictor and the corrector are then computed by,

$$EYYYY3(I) = DABS(EYYYY3(I) - YYY(I)) \quad , \quad (3)$$

and the ratio between the integration error and the maximum allowed relative and absolute errors is now evaluated by,

$$ERR = EYYYY3(I) / (DABS(YYY(I)) * EYYYY1(I) + EYYYY2(I)) \quad .(4)$$

If ERR is greater than one, the integration process is repeated with half of the previous time step, and the time division indicator, IDIV, is set to one. If ERR is less than one, The integration process is completed by recalling the DERIV1 subroutine. If ERR is less than 0.005, the time doubling indicator, IDUB, is set to one.

(8) APPLICABILITY AND RESTRICTIONS.

This integration method can be used in loops where slow transients are expected. With fast transients this method will reduce the time step significantly. The integration algorithm can be used with fixed time steps. If LETM is equal to cre, the error calculating section will be skipped.

* UCORC1 * (ZPE,PD,CK,RI,RX,D,U)

(1) PURPOSE.

This subroutine calculates the heat transfer coefficient between a fuel pin and the coolant in a fuel bundle.

(2) PARAMETERS.

CK Thermal conductivity of the cladding, in W/m-C.
D Hydraulic diameter of the equivalent coolant channel, in m.
U Computed heat transfer coefficient, in W/m²-C.
PD P/D-pitch to diameter ratio in the fuel bundle.
RI Fuel pin cladding internal radius, in m.
RX Fuel pin radius, in m.
ZPE Peclet number (Dimensionless).

(3) COMMON VARIABLES.

None.

(4) LOCAL VARIABLES.

ZNU Nusselt number (Dimensionless).

(5) OTHER SUBROUTINES OR MODULES REQUIRED.

None.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

As recommended by Kazimi for the CFBFP [1], the West [2] correlation is used for calculating the heat transfer coefficients for fuel subassemblies where P/D varies

between 1.1 and 1.4. The Schad and modified Schad [3] correlations were used for radial blanket subassemblies when P/D varies between 1.05 and 1.2. The Schad correlation for $1.05 \leq P/D \leq 1.2$ and $150 \leq ZPE \leq 1000$ is given by,

$$ZNU = (-16.15 + 24.96*PD - 8.55*PD**2)*ZPE**0.3 \quad , \quad (1)$$

and for $ZPE \leq 150$, the modified correlation [1] is used.

$$ZNU = 4.496*(-16.15 + 24.96*PD - 8.55*PD**2) \quad (2)$$

The West correlation for $1.1 \leq PD \leq 1.4$ and $10 \leq ZPE \leq 5000$ is given by,

$$ZNU = 4.0 + 0.33*PD**3.8*(ZPE/100)**0.86 + 0.16*PD**5 \quad . \quad (3)$$

These correlations are then used to calculate the heat transfer coefficient U from,

$$U = 1 / (D / (ZNU * CK) + RX / CK * DLOG((2 * RX) / (RI + RX))) \quad . \quad (4)$$

(7) ORGANIZATION OF MODULE.

Upon entering the subroutine, a check is made on the P/D ratio. If the ratio is less than 1.15, then the Schad or modified Schad correlation is used according to the value of the Peclet number ZPE or else the West correlation is used. The heat transfer coefficient is computed from the Nusselt number ZNU using Eq. (4).

(8) APPLICABILITY AND RESTRICTIONS.

The specific purpose of the module is to compute the heat transfer coefficient U from the fuel pin cladding to the coolant and is limited to $1.05 \leq P/D \leq 1.5$. To use other heat transfer correlations for the fuel bundle, this module or its content should be replaced. The same calling parameters must be used when the subroutine is replaced. To use this module in the simulation, the following statement should be included.

CALL UCCRC1(ZPE,PD,CK,D,PI,RX,U)

REFERENCES.

- [1] M. S. Kazi, "Heat Transfer Correlation for Analysis of CRBRP Assemblies", WARD-D-0034, LRA-74-114, (1974).
- [2] H. West, "Heat Transfer Correlation Recommendation for

FFTF", WAFD-FRT-60, (1969).

- [3] H. C. Schad, "Heat Transfer Correlation for LMFBF Design", AFD-TH-426, (1969).

```

*****
* UCORL1 *   (CKF,CKC,FJ,RX,U)
*****

```

(1) PURPOSE.

This subroutine calculates the fuel pin to cladding heat transfer coefficient.

(2) PARAMETERS.

CKC Fuel pin cladding conductivity, in W/m-C.
 CKF Fuel conductivity, in W/m-C.
 RI Fuel pin cladding inner radius, in m.
 RX Fuel pin outer radius, in m.
 U Calculated heat transfer coefficient, in W/m²-C.

(3) COMMON VARIABLES.

None.

(4) LOCAL VARIABLES.

H Fuel to cladding gap coefficient (8373.6), in W/m²-C.

(5) OTHER SUBROUTINES OR MODULES REQUIRED.

None.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

The fuel to cladding heat transfer coefficient is given by the following equation.

$$U = 1 / (RI / (8 * CKF) + 1 / H + RI / CKC * DLOG((RI + RX) / (2 * RI))) \quad (1)$$

A proper expression to calculate the gap coefficient H should be supplied.

(7) ORGANIZATION OF MODULE.

The variables H and U are computed in this subroutine and are based upon the expression given above but can be replaced by the user if he wishes to use a different expression for the heat transfer coefficient.

(8) APPLICABILITY AND RESTRICTIONS.

The gap heat transfer coefficient is given in this subroutine as a constant $H=8373.6 \text{ W/m}^2\text{-C}$. The relationship should be replaced by an appropriate expression when available. To use this module in the simulation, the following statement should be included.

CALL UCORL1(CKC,CKF,RI,RX,U)

* UPIEX1 * (RI, RX, DH, CK, ZNU)

(1) PURPOSE.

This module computes the heat transfer coefficients for a internal and external pipe wall gap.

(2) PARAMETERS.

CK Pipe wall conductivity, in W/m^2-C .

DH Hydraulic diameter, in m.

RX Pipe external radius, in m.

RI Pipe internal radius, in m.

ZNU Nusselt number (Dimensionless).

(3) COMMON VARIABLES.

None.

(4) LOCAL VARIABLES.

UPIEX1 The external tube wall gap heat transfer coefficient, in W/m^2-C .

UPIIN1 The internal tube wall gap heat transfer coefficient, in W/m^2-C .

(5) OTHER SUBROUTINES OR MODULES REQUIRED.

None.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

The outer tube wall to coolant heat transfer coefficient is given by,

$$UPIEX1 = CK / (DH / ZNU + RX * DLOG(2 * RX / (RI + RX))) \quad (1)$$

The inner tube wall to coolant heat transfer coefficient is given by,

$$UPIIN1 = CK / (DH / ZNU + RI * DLOG((RI + RX) / (2 * RI))) \quad .(2)$$

(7) ORGANIZATION OF MODULE.

This function has two entries. The main function entry UPIEX1 calculates the tube external gap coefficient using Eq. (1), and the second entry UPIIN1 calculates the internal tube gap coefficient using Eq. (2).

(8) APPLICABILITY AND RESTRICTIONS.

The gap heat transfer coefficients calculated by this function are applicable to tube and shell heat exchangers using sodium or water. Their precision depends upon how well the Nusselt number ZNU is calculated. To use either of the above functions in the simulation, the following statements should be included.

$$H = UPIFX1(RI, RX, DH, CK, ZNU)$$

and

$$H = UPIIN1(RI, RX, DH, CK, ZNU)$$

```

*****
* UPLEN1 * (LCOP)
*****

```

(1) PURPOSE.

This module simulates the thermodynamics of the core outlet upper plenum.

(2) PARAMETERS.

LCOP Indicator to the loop from which this subroutine is called.

(3) COMMON VARIABLES.

<u>NAMES</u>	<u>COMBLK</u>	<u>DESCRIPTION</u>
CC	CORTP	Primary coolant specific heat, in J/kg-C.
CN	UPLEN	Specific heat of the metal in the upper plenum, in J/kg-C.
QUPF	UPLEN	Weight of metal in the upper plenum, in kg.
RCCN	UPLEN	Coolant average density in the upper plenum, in kg/m ³ .
TENA	UPLEN	Average coolant temperature in the upper plenum, in C.
TENFA	UPLEN	Average metal temperature in the upper plenum, in C.
TENI	UPLEN	Upper plenum inlet temperature, in C.
TENK	UPLEN	Upper plenum coolant exit temperature, in C.
UNA	UPLEN	Total heat transfer from metal to coolant in the upper plenum, in W/C.
VNC	UPLEN	Volume of coolant in the upper plenum, in m ³ .

(4) LOCAL VARIABLES.

CNC Total upper plenum heat capacity given by Eq. (4),
in J/C.

CNF Total upper plenum metal node heat capacity given by
Eq. (3), in J/C.

(5) OTHER SUBROUTINES OR MODULES REQUIRED.

Subroutines: SODEN1.

Common Blocks: CORTP5, DERIV5, TIMEP5, UPLEN5.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

This is a simplified lumped parameter model of the core exit or upper plenum. It is assumed to have a metal node that includes all of the metal parts in the upper plenum and a coolant node that includes the total volume of the upper plenum coolant.

The metal and coolant node temperatures are calculated from the appropriate heat balance equations. The coolant average temperature TENA is given by,

$$TENA = \frac{\text{INTEGRAL}[CC*GC*(TENI - TENA) + UNA*(TENFA - TENA)]}{CNC} \quad (1)$$

and the metal node average temperature is given by,

$$TENFA = \text{INTEGRAL}[UNA*(TENA - TENFA)/CNF] \quad (2)$$

It is assumed that ideal mixing prevails in the upper plenum and that the exit temperature is equal to the coolant average temperature.

(7) ORGANIZATION OF MODULE.

Upon the first entry to the module (JTIM=0), the initial conditions are calculated. Since there is no heat source in the upper plenum, all of the temperatures are initially assumed to be equal to the module inlet temperature TENI. This value should be supplied either as input data or by a communication command connecting TENI to another module where this temperature is calculated. The metal heat capacity CNF is also calculated in the initial section by,

$$CNF = CN * OLPF \quad (3)$$

since it is assumed to be constant.

The coolant density ρ_{CN} is the first value calculated in a dynamic entry by calling $SODEN1$ and is used in calculating the total coolant heat capacity by,

$$CNC = VNC * CC * \rho_{CN} \quad . \quad (4)$$

Next, the metal node and coolant node average temperatures are calculated by integrating Eqs. (1) and (2). A centralized integration method is used as defined by the user. Finally, the module exit temperature T_{EX} is made equal to the coolant average temperature.

(8) APPLICABILITY AND RESTRICTIONS.

The upper plenum inlet temperature must be supplied either as input or by a communication module connecting T_{ENI} to another module where this temperature is calculated. It is also suggested that the exit of this module be connected to another module by a pipe delay to account for the error introduced by the ideal mixing assumption. To use this module in the simulation, the following statement should be included.

- $UPLN1$


```

*****
* VISCO1 *   (VS,ZMAT,TEMP,RO)
*****

```

(1) PURPOSE.

This subroutine calculates the dynamic viscosity of material, ZMAT, by calling the appropriate material property subroutine.

(2) PARAMETERS.

RC Material density, in kg/m³.

TEMP Temperature at which the viscosity is to be evaluated, in C.

VS Material viscosity, in Pa-s.

ZMAT Alphanumeric indicator of the material for which the viscosity is to be evaluated.

(3) COMMON VARIABLES.

<u>NAMES</u>	<u>COMBLK</u>	<u>DESCRIPTION</u>
IC6	TIMER	Standard FORTRAN output unit (usually 6).
ZZZSO	CONST	"SO" indicating sodium properties.
ZZZSS	CONST	"SS" indicating stainless steel properties.
ZZZST	CONST	"ST" indicating steam properties.
ZZZWA	CONST	"WA" indicating water properties.

(4) LOCAL VARIABLES.

V Specific volume at TEMP, in m³/kg.

(5) OTHER SUBROUTINES OR MODULES REQUIRED.

Subroutines: SODMU1, STD MU1, WADMU1.
Common Blocks: CONST5, TIMER5.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

The viscosity of the requested material is evaluated by calling the appropriate material property function.

(7) ORGANIZATION OF MODULE.

The content of ZMAT is tested for the indicator of the material for which the viscosity is to be evaluated. If ZMAT=SO, the viscosity of sodium is computed by calling SODMU1. If ZMAT=SS, an error message is printed. If ZMAT=WA or ST, the viscosity of water or steam is computed by calling WADMU1 or STD MU1. If ZMAT does not contain any of the above names, an error message is printed.

(8) APPLICABILITY AND RESTRICTIONS.

At present, the viscosity of only four materials, as shown above, can be computed. The number can, however, be easily expanded by adding additional material property subroutines. The RO parameter is required only for water or steam property calculations. To use this module in the simulation, the following statement should be included.

CALL VISC01(VS,ZMAT,TEMP,RO)

* WACCN1 * (TEMP,CO,RO)

(1) PURPOSE.

This subroutine calculates the thermal conductivity of water.

(2) PARAMETERS.

CO Water conductivity, in W/m-C.

RO Material density, in kg/m³.

TEMP Water temperature, in C.

(3) COMMON VARIABLES.

None.

(4) LOCAL VARIABLES.

B(16) Vector of coefficients used in computing water or steam conductivity.

EL1,EL2 Intermediate variables used by the conductivity.
EL3,EL4 equation.

R1 Normalized density equal to RO/1000.

R2 R1*R1.

T1 Normalized temperature equal to TEMP/100. If T1 is less than 0.01, then T1 is set equal to 0.01.

T2 T1*T1.

T3 T1*T1*T1.

T4 T1*T1*T1*T1.

(5) OTHER SUBROUTINES OR MODULES REQUIRED.

None.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

A single semiempirical equation is used to calculate the water or steam conductivity.

$$CO = (EL1 + EL2 + EL3 + EL4) / 1000 \quad (1)$$

And,

$$EL1 = E(1) + E(2) * T1 + B(3) * T2 + E(4) * T3 \quad (2)$$

$$EL2 = E(5) * P1 + B(6) * R2 \quad (3)$$

$$EL3 = (P1 * (1 - P1) ** 3) * T3 * (B(7) * EXP(-B(8) * T1) + P(9) * EXP(-P(10) * T1)) \quad (4)$$

$$EL4 = -(E(11) * EXP(-B(12) * T4) + B(13) * EXP(-B(14) * T2) + E(15) * EXP(-P(16) * T1)) * R1 \quad (5)$$

(7) ORGANIZATION OF MODULE.

The parameters of the conductivity equation are stored in the vector P by an appropriate DATA statement. The normalized temperatures T and densities D are computed first, followed by the various powers of these variables. The conductivity of either water or steam, or their mixture is then computed. For logical reasons, the subroutine has two entries to the same point. One entry WACON1 is for water and the other one STCON1 for steam.

(8) APPLICABILITY AND RESTRICTIONS.

The above equations are valid within the temperature range between 0 and 700 C., and the pressure range from 0 to 5E7 Pa. To use this module in the simulation, the following statement should be included.

CALL WACON1(TEMP,CO,RO)

* WADCP1 * (TEC,CPS,P)

(1) PURPOSE.

This subroutine calculates the specific heat of water.

(2) PARAMETERS.

CPS Water specific heat, in J/kg-C.

P Pressure, in Pa.

TEC Water temperature, in C.

(3) COMMON VARIABLES.

None.

(4) LOCAL VARIABLES.

F Helmholtz function.

F1 First derivative of the Helmholtz function w/r to temperature (constant specific volume).

F2 First derivative of the Helmholtz function w/r to specific volume (constant temperature).

F11 Second derivative of the Helmholtz function w/r to temperature (constant specific volume).

F12 Second derivative of the Helmholtz function w/r to temperature and specific volume.

F22 Second derivative of the Helmholtz function w/r to specific volume (constant temperature).

K A constant indicating steam (100) or water (200) on first call to WASPT1.

KW An indicator to the first entry to the subroutine and initially is set to .FALSE. .

TOUT Equal to TEC.

V Specific volume, in m³/kg.

(5) OTHER SUBROUTINES OR MODULES REQUIRED.

Subroutines: WASPT1.
Common Blocks: CONST5.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

The specific heat of water is computed from the absolute water temperature and the appropriate Helmholtz derivatives from,

$$CPS = (TCUT + 273) * (F12 * F12 - F11) / F22 \quad . \quad (1)$$

(7) ORGANIZATION OF MODULE.

The first entry to the subroutine will set K to 100 and V to -1.0 indicating the first calculation of the steam properties by WASPT1. Subsequent entries will use previous values of V as their initial guess. Each entry will use WASPT1 to compute the Helmholtz derivatives and to solve Eq. (1).

(8) APPLICABILITY AND RESTRICTIONS.

The specific heat equation is applicable over the range between 0.01 and 374.15 C. To use this module in the simulation, the following statement should be included.

CALL WADCP1(TEC,CPS,P)

* WADEN1 * (TEC,FOC,P)

(1) PURPOSE.

This subroutine calculates the density of water given the temperature and pressure.

(2) PARAMETERS.

P Pressure, in Pa.
FOC Water density, in kg/m³.
TEC Water temperature, in C.

(3) COMMON VARIABLES.

None.

(4) LOCAL VARIABLES.

K1 Steam (100) or water (200) indicator.
TCUT Return temperature from WASPT1 and is the same as
 TEC, in C.
V Specific volume, kg/m³.

(5) OTHER SUBROUTINES OR MODULES REQUIRED.

Subroutines: WASPT1.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

The density of water is computed by using the steam-water property subroutine WASPT1 which provides the specific volume. The density is then calculated from,

$$\text{FOC} = 1/V \quad .$$

(1)

(7) ORGANIZATION OF MODULE.

This subroutine has two entries. The main entry is WADEN1, and the entry through which the steam density STDEN1 is calculated. When the WADEN1 entry is used, K1 is set to 100, while if the STDEN1 entry is used, K1 is set to 200. If the density is in the range between 1 and 1000 kg/m³, that is, the expected range for the water and steam densities, V is set to 1/ROC. Otherwise V is set to -1 in order to initialize the WASPT1 subroutine.

(8) APPLICABILITY AND RESTRICTIONS.

The density evaluation is applicable over the range between 0.01 and 374.15 C. To use this module in the simulation, the following statement should be included.

CALL WADEN1(TEC,ROC,P)

* WADMU1 * (VS,TEMP,RO)

(1) PURPOSE.

This subroutine calculates the dynamic viscosity of water given the temperature and density.

(2) PARAMETERS.

RO Water density, in kg/m^3 .
VS Water dynamic viscosity, in kg/sec-m .
TEMP Water temperature, in C.

(3) COMMON VARIABLES.

None.

(4) LOCAL VARIABLES.

A(14) Semiempirical constants used in evaluating the water and steam viscosity.
EL1,EL2 Intermediate variables used in computing VS.
EI3,EL4
R1 The value of $RO/100$.
R2 the value of $R1*R1$.
R3 The value of $R1*R1*R1$.
T1 The value of $TEMP/100$.
T6 The value of $T1*T1*T1*T1*T1*T1$.

(5) OTHER SUBROUTINES OR MODULES REQUIRED.

None.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

The dynamic viscosity of water for a wide range of temperatures [1] is given by,

$$VS = (EL1 + EL2 + EL3 + EL4) * 1.D-7 \quad , \quad (1)$$

where,

$$EL1 = A(1) + A(2) * T1 \quad , \quad (2)$$

$$EL2 = A(3) * P1 + A(4) * P2 \quad , \quad (3)$$

$$EL3 = -A(5) * R1 * (1 - R1) ** 3 * \text{DEXP}(-A(6) * T6) \quad , \quad (4)$$

and,

$$EL4 = (A(7) * \text{DEXP}(-A(8) * T1) + A(9) * \text{DEXP}(-A(10) * T1) + A(11) * \text{DEXP}(-A(12) * T1) + A(13) * (1 - R) * \text{DEXP}(-A(14) * T1)) * R3 \quad . \quad (5)$$

(7) ORGANIZATION OF MODULE.

None.

(8) APPLICABILITY AND RESTRICTIONS.

The above equations are valid within the temperature range between 0 and 700 C., and the pressure range between 0 and 8E7 Pa. To use this module in the simulation, the following statement should be included.

CALL WADMU(VS,TEMP,RO)

REFERENCES.

[1] 1968 IFC Formulation For Scientific and General Use.

* WAHXP1 * (U,ZFE,ZPR,RI,RX,D,CK,P,PD)

(1) PURPOSE.

This subroutine calculates the gap heat transfer coefficient between the outside tube wall and the liquid.

(2) PARAMETERS.

CK Tube conductivity, in W/m-C.
D Hydraulic diameter of the equivalent channel on IHX shell side, in m.
P Pressure, in Pa. (not used)
PD P/D pitch to diameter ratio.
RI IHX tube internal radius, in m.
RX IHX tube external diameter, in m.
U Tube to primary (or external) coolant heat transfer coefficient, in W/m²-C.
ZPR Prandtl number, (dimensionless).
ZFE Reynolds number, (dimensionless).

(3) COMMON VARIABLES.

None.

(4) LOCAL VARIABLES.

None.

(5) OTHER SUBROUTINES OR MODULES REQUIRED.

Functions: UPIEX1.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

The Nusselt number is given by,

$$ZNU=0.023*ZRE**0.8*ZPR**0.4 \quad . \quad (1)$$

(7) ORGANIZATION OF MODULE.

Upon entering the subroutine, Eq. (1) is solved to obtain the Nusselt number. Next, the function UPIEX1 is used to obtain the heat transfer coefficient U.

(8) APPLICABILITY AND RESTRICTIONS.

The heat transfer coefficient can be used for straight tube counterflow heat exchangers. To use this module in the simulation, the following statement should be included.

CALL WAHXP1(U,ZRE,ZPR,RI,RX,D,CK,P.PD)

* WAHXS1 * (U,ZPF,ZPR,RI,RX,D,CK,P,PD)

(1) PURPOSE.

This subroutine calculates the gap heat transfer coefficient in a heat exchanger between the tube inner wall and the fluid.

(2) PARAMETERS.

CK Tube conductivity, in W/m-C.

D Hydraulic diameter of the equivalent channel on IHX shell side, in m.

P Pressure, in Pa. (not used)

PD P/D pitch to diameter ratio.

RI IHX tube internal radius, in m.

RX IHX tube external diameter, in m.

U Tube to secondary (or internal) coolant heat transfer coefficient, in W/m²-C.

ZPR Prandtl number (Dimensionless).

ZRE Reynolds number (Dimensionless).

(3) COMMON VARIABLES.

None.

(4) LOCAL VARIABLES.

None.

(5) OTHER SUBROUTINES OR MODULES REQUIRED.

Functions: UPJIN1.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

The Nusselt number is given by,

$$ZNu = 0.023 * ZRe^{0.8} * ZPr^{0.4} \quad . \quad (1)$$

(7) ORGANIZATION OF MODULE.

Upon entering the subroutine, Eq. (1) is solved to obtain the Nusselt number. Next, the function UPIIN1 is used to obtain the heat transfer coefficient U. The hydraulic diameter is taken as 2*RI.

(8) APPLICABILITY AND RESTRICTIONS.

This heat transfer correlation can be used for straight tube counterflow heat exchangers. To use this module in the simulation, the following statement should be included.

CALL WAHXS1(U,ZRE,ZPR,RI,RX,D,CK,P,PD)

* WASAP1 * (VAR1,VAR2,QUO)

(1) PURPOSE.

This subroutine calculates the water vapor pressure as a function of temperature along the saturation line.

(2) PARAMETERS.

QUO Derivative of pressure with respect to temperature.

VAR1 Pressure, in Pa.

VAR2 Temperature, in C.

(3) COMMON VARIABLES.

None.

(4) LOCAL VARIABLES.

FUN Normalized pressure equal to $\text{LOG}(\text{VAR1}/2.212\text{D7})$.

THETA Normalized temperature equal to $(\text{VAR2}+273.15)/647.3$.

(5) OTHER SUBROUTINES OF MODULES REQUIRED.

Common Blocks: CONST5.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

Not available at the present time. The text will be supplied in the next revision of this document.

(7) ORGANIZATION OF MODULE.

Not available at the present time. The text will be supplied in the next revision of this document.

(8) APPLICABILITY AND RESTRICTIONS.

Applicability is limited to the range of temperatures defining the saturation line from 0.01 to 374.15 C. To use this module in the simulation, the following statement should be included.

CALL WASAP1(VAR1,VAR2,QUO)

REFERENCES.

[1] 1968 IFC Formulation For Scientific and General Use.

* WASAT1 * (VAR1,VAR2,QUO)

(1) PURPOSE.

This subroutine calculates the temperature along the saturation line for a given water vapor pressure.

(2) PARAMETERS.

QUO Derivative of pressure with respect to temperature.

VAR1 Pressure, in Pa.

VAR2 Temperature, in C.

(3) COMMON VARIABLES.

Ncne.

(4) LOCAL VARIABLES.

FUN Normalized pressure equal to $\text{LOG}(\text{VAR1}/2.21207)$.

THETA Normalized temperature equal to $(\text{VAR2}+273.15)/647.3$.

(5) OTHER SUBROUTINES OR MODULES REQUIRED.

Common Blocks: CONST5.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

Not available at the present time. The text will be supplied in the next revision of this document.

(7) ORGANIZATION OF MODULE.

Not available at the present time. The text will be supplied in the next revision of this document.

(8) APPLICABILITY AND RESTRICTIONS.

Applicability is limited to the range of temperatures defining the saturation line from 0.01 to 374.15 C. To use this module in the simulation, the following statement should be included.

CALL WASAT1(VAR1,VAR2,QUO)

REFERENCES.

[1] 1968 IFC Formulation For Scientific and General Use.

* WASHP1 * (VI1,VI2,VO1,VO2,Z,Z1,Z2,Z11,Z12,Z22,L)

(1) PURPOSE.

This subroutine computes the Helmholtz function for single-phase water (liquid or vapor) as a function of enthalpy and pressure.

(2) PARAMETERS.

L Number of iterations needed for solution.

VI1 Enthalpy, in J/kg.

VI2 Pressure, in Pa.

VO1 Temperature, in C.

VO2 Specific volume, in m³/kg.

Z The Helmholtz function.

Z1 The first derivative of the Helmholtz function w/r to temperature (constant specific volume).

Z11 The second derivative of the Helmholtz function w/r to temperature (constant specific volume).

Z12 The second derivative of the Helmholtz function w/r to temperature and specific volume.

Z2 The first derivative of the Helmholtz function w/r to specific volume (constant temperature).

Z22 The second derivative of the Helmholtz function w/r to specific volume (constant temperature).

(3) COMMON VARIABLES.

<u>NAMES</u>	<u>COMBLK</u>	<u>DESCRIPTION</u>
F1	WASPR	The difference between the estimated and given state variable VI1.
F11	WASPR	The derivative of F1 w/r to temperature with constant specific volume.

F12	WASPR	The derivative of F1 w/r to specific volume with constant temperature.
F2	WASPR	The difference between the estimated and given state variable VI2.
F21	WASPR	The derivative of F2 w/r to temperature with constant specific volume.
F22	WASPR	The derivative of F2 w/r specific volume with constant temperature.
IS2	WASPR	Numerical indicator to the WAS property routine being used.
TES1	WASPR	Upper error limit in steam-water property evaluations and usually set to $10E-4$.
TES2	WASPR	Upper error limit in steam-water property evaluations and usually set to $10E-4$.
TNEW	WASPR	Recent value of the temperature computed in one of the "WAS" routines, in C.
TOLD	WASPR	Temperature value of the previous iteration in the "WAS" routines, in C.
VNEW	WASPR	Recent value of the specific volume computed in one of the "WAS" routines, in m^3/kg .
VCID	WASPR	Specific volume of the previous iteration in the "WAS" routines, in m^3/kg .

(3) COMMON VARIABLES.

Ncne.

(4) LOCAL VARIABLES.

IRET An indicator that, if zero, indicates that convergence was achieved or that the maximum number of allowed iterations was exceeded.

T Absolute temperature equal to TOLD+273.15, in K.

(5) OTHER SUBROUTINES OR MODULES REQUIRED.

Subroutines: WASPR1.

Common Blocks: CONST5, TIMER5, WASPR5.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

Newton's approximation method is used to solve the implicit set of thermodynamic relations to obtain the Helmholtz function and its derivatives. The following relations are used in this subroutine.

$$F1 = (Z - T * Z1 - VOLD * Z2) - VI1 \quad (1)$$

$$F2 = -Z2 - VI2 \quad (2)$$

$$F11 = (-T * Z11 - VOLD * Z12) \quad (3)$$

$$F12 = (-T * Z12 - VOLD * Z22) \quad (4)$$

$$F21 = -Z12 \quad (5)$$

$$F22 = -Z22 \quad (6)$$

(7) ORGANIZATION OF MODULE.

Upon entering the subroutine, TNEW and VNEW are set to the last known value of VO1 and VO2 respectively. Next, the convergence criteria TES1 and TES2 are set. These can be reset by the user to any desired value. Then, the WASPR1 subroutine is called to obtain the first approximation to the Helmholtz function and the first and second derivatives. These approximations are then used in Eqs. (1) to (6) to calculate the differences between the given and estimated values of the state variables, and the derivatives w/r to temperature and specific volume. The WASPR1 subroutine is called again to obtain the new estimates for the enthalpy and pressure using the differences and derivatives obtained in Eqs. (1) to (6). If convergence is not achieved or when IRET is not equal to zero, new values of the Helmholtz function are computed and Eqs. (1) to (6) are recalculated. The iterative process is terminated when IRET is equal to zero.

(8) APPLICABILITY AND RESTRICTIONS.

Values of VI1 and VI2 must correspond to single-phase water (liquid or vapor) at temperatures less than 1300 C and pressures less than 100 MPa. Approximate values of

VO1 and VO2 must be provided when calling the program. The speed of convergence will depend upon the closeness of these values to the solution. Upon return, VO1 and VO2 will correspond to VI1 and VI2. To use this module in the simulation, the following statement should be included.

```
CALL WASHP1(VI1,VI2,VO1,VO2,Z,Z1,Z2,Z11,Z12,Z22,L)
```

REFERENCES.

- [1] J. H. Keenan, F. G. Keyes, P. G. Hill, and J. G. Moore, "Steam Tables", John Wiley, (1969).

* WASHV1 * (VI1,VI2,VC1,VC2,Z,Z1,Z2,Z11,Z12,Z22,L)

(1) PURPOSE.

This subroutine computes the Helmholtz function for single-phase water (liquid or vapor) as a function of enthalpy and specific volume.

(2) PARAMETERS.

L Number of iterations needed for solution.

VI1 Enthalpy, in J/kg.

VI2 Specific volume, in m³/kg.

VO1 Temperature, in C.

VC2 Specific volume, in m³/kg.

Z The Helmholtz function.

Z1 The first derivative of the Helmholtz function w/r to temperature (constant specific volume).

Z11 The second derivative of the Helmholtz function w/r to temperature (constant specific volume).

Z12 The second derivative of the Helmholtz function w/r to temperature and specific volume.

Z2 The first derivative of the Helmholtz function w/r to specific volume (constant temperature).

Z22 The second derivative of the Helmholtz function w/r to specific volume (constant temperature).

(3) COMMON VARIABLES.

<u>NAMES</u>	<u>COMBLK</u>	<u>DESCRIPTION</u>
F1	WASPR	The difference between the estimated and given state variable VI1.
F11	WASPR	The derivative of F1 w/r to temperature with constant specific volume.

F12	WASPR	The derivative of F1 w/r to specific volume with constant temperature.
F2	WASPR	The difference between the estimated and given state variable VI2.
F21	WASPR	The derivative of F2 w/r to temperature with constant specific volume.
F22	WASPR	The derivative of F2 w/r specific volume with constant temperature.
IS2	WASPR	Numerical indicator to the WAS property routine being used.
TES1	WASPR	Upper error limit in steam-water property evaluations and usually set to 10E-4.
TES2	WASPR	Upper error limit in steam-water property evaluations and usually set to 10E-4.
TNEW	WASPR	Recent value of the temperature computed in one of the "WAS" routines, in C.
TCLD	WASPR	Temperature value of the previous iteration in the "WAS" routines, in C.
VNEW	WASPR	Recent value of the specific volume computed in one of the "WAS" routines, in m ³ /kg.
VCID	WASPR	Specific volume of the previous iteration in the "WAS" routines, in m ³ /kg.

(4) LOCAL VARIABLES.

IRET An indicator that, if zero, indicates that convergence was achieved or that the maximum number of allowed iterations was exceeded.

T Absolute temperature equal to TOLD+273.15, in K.

(5) OTHER SUBROUTINES OR MODULES REQUIRED.

Subroutines: WASPR1.

Common Blocks: CONST5, TIMER5, WASPR5.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

Newton's approximation method is used to solve the implicit set of thermodynamic relations to obtain the Helmholtz function and its derivatives. The following relations are used in this subroutine.

$$F1 = (2 - T * Z1 - VI2 * Z2) - VI1 \quad (1)$$

$$F11 = (-T * Z11 - VI2 * Z12) \quad (2)$$

(7) ORGANIZATION OF MODULE.

Upon entering the subroutine, TNEW and VNEW are set to the last known value of VO1 and VO2 respectively. Next, the convergence criteria TES1 and TES2 are set. These can be reset by the user to any desired value. Then, the WASPR1 subroutine is called to obtain the first approximation to the Helmholtz function and the first and second derivatives. These approximations are then used in Eqs. (1) and (2) to calculate the differences between the given and estimated values of the state variables, and the derivatives w/r to temperature and specific volume. The WASPR1 subroutine is called again to obtain the new estimates for the enthalpy using the differences and derivatives obtained in Eqs. (1) and (2). If convergence is not achieved or when IRET is not equal to zero, new values of the Helmholtz function are computed and Eqs. (1) and (2) are recalculated. The iterative process is terminated when IRET is equal to zero.

(8) APPLICABILITY AND RESTRICTIONS.

Values of VI1 and VI2 must correspond to single-phase water (liquid or vapor) at temperatures less than 1300 C and pressures less than 100 MPa. Approximate values of VO1 and VO2 must be provided when calling the program. The speed of convergence will depend upon the closeness of these values to the solution. Upon return, VO1 and VO2 will correspond to VI1 and VI2. To use this module in the simulation, the following statement should be included.

CALL WASHV1(VI1,VI2,VO1,VO2,Z,Z1,Z2,Z11,Z12,Z22,I)

REFERENCES.

- [1] J. H. Keenan, P. G. Keyes, P. G. Hill, and J. G. Moore,
"Steam Tables", John Wiley, (1969).

(* WASIN1 * (VI1,VI2,TCID,VOLD,L)

(1) PURPOSE.

This subroutine provides initial estimates of the temperature and specific volume as functions of VI1 and VI2.

(2) PARAMETERS.

L Parameter to be used if independent variables are pressure and temperature. Then if,
I equals 100, a value of VOLD will be calculated corresponding to the compressed liquid at VI1 and VI2, or if
L equals 200, a value of VOLD will be calculated corresponding to the superheated vapor at VI1 and VI2.

TCID Temperature, in C.

VI1 First independent variable.

VI2 Second independent variable.

VOLD Specific volume, in m³/kg.

(3) COMMON VARIABLES.

Ncne.

(4) LOCAL VARIABLES.

AL0 A constant of 15.74373327.

AI1 A constant of -34.17061978.

AL2 A constant of 19.31380707.

PCID Last value of pressure, in Pa.

PSATF Saturation pressure at the temperature TOLD, in Pa.

PSI Pressure at the water region boundary corresponding

to the TOLD temperature, in Pa.

PX An arbitrary pressure close to the critical pressure
and set to 2E7, in Pa.

T Normalized temperature.

(5) OTHER SUBROUTINES OR MODULES REQUIRED.

Subroutines: WASAP1.

Common Blocks: CONST5, TIMER5.

Functions: WASTD1, WASTL1.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

This subroutine uses the function WASTL1 to estimate the specific volume in the liquid region, and the function WASTD1 to estimate the specific volume in the vapor region.

(7) ORGANIZATION OF MODULE.

Upon entering the subroutine, L is tested. If L is either 100 for liquid or 200 for vapor, the specific volume is estimated as explained above. If L is any other value, the location of the point having a pressure VI1 and temperature VI2 in the P-T phase diagram is located and the appropriate specific volume estimated.

(8) APPLICABILITY AND RESTRICTIONS.

The values of VI1 and VI2 must correspond to single-phase water (liquid or vapor) at temperatures less than 1300 C and pressures less than 100 MPa. To use this module in the simulation, the following statement should be included.

CALL WASIN1(VI1,VI2,TOLD,VOLD,L)

* WASPR1 * (IGO,VI1,VI2,VO1,VO2,Z,Z1,Z2,Z11,Z12,Z22,L,IRET)

(1) PURPOSE.

This subroutine calculates the water properties VO1 and VC2 as functions of VI1 and VI2.

(2) PARAMETERS.

IGO A parameter that allows this program to perform three functions. If IGO equals one and if VI1 or VI2 are negative, the initial values of VO1 and VO2 will be calculated. If IGO equals two, the Helmholtz function and the derivatives are calculated. If IGO equals three, the Newton method is used to estimate the new values for temperature and specific volume.

IRET If the returned value is equal to zero, then convergence has been obtained or the specified number of iterations have been exceeded. If the returned value is equal to one, iteration will continue.

L Number of iterations needed for the solution. When VI1 and VI2 correspond to the pressure and temperature and VNEW is less than one and IGO equals one, then if,
L equals 100, a value of VNEW will be calculated corresponding to the compressed liquid at VI1 and VI2, or if
L equals 200, a value of VNEW will be calculated corresponding to the superheated vapor at VI1 and VI2.

VI1 First independent variable.

VI2 Second independent variable.

VO1 Temperature, in C.

VC2 Specific volume, in m³/kg.

Z The Helmholtz function.

Z1 The first derivative of the Helmholtz function w/r to temperature (constant specific volume).

Z11 The second derivative of the Helmholtz function w/r

to temperature (constant specific volume).

- Z12 The second derivative of the Helmholtz function w/r to temperature and specific volume.
- Z2 The first derivative of the Helmholtz function w/r to specific volume (constant temperature).
- Z22 The second derivative of the Helmholtz function w/r to specific volume (constant temperature).

(3) COMMON VARIABLES.

<u>NAMES</u>	<u>COMELK</u>	<u>DESCRIPTION</u>
F1	WASPR	The difference between the estimated and given state variable VI1.
F11	WASPR	The derivative of F1 w/r to temperature with constant specific volume.
F12	WASPR	The derivative of F1 w/r to specific volume with constant temperature.
F2	WASPR	The difference between the estimated and given state variable VI2.
F21	WASPR	The derivative of F2 w/r to temperature with constant specific volume.
F22	WASPR	The derivative of F2 w/r specific volume with constant temperature.
IO6	TIMER	Standard FORTRAN output unit (usually 6).
IS2	WASPR	Numerical indicator to the WAS property routine being used.
JTIM	TIMER	Time step counter.
TES1	WASPR	Upper error limit in steam-water property evaluations and usually set to 10E-4.
TES2	WASPR	Upper error limit in steam-water property evaluations and usually set to 10E-4.
TIME	TIMER	Current simulation time, in s.

TNEW	WASPR	Recent value of the temperature computed in one of the "WAS" routines, in C.
TOLD	WASPR	Temperature value of the previous iteration in the "WAS" routines, in C.
VNEW	WASPR	Recent value of the specific volume computed in one of the "WAS" routines, in m ³ /kg.
VCLD	WASPR	Specific volume of the previous iteration in the "WAS" routines, in m ³ /kg.

(4) LOCAL VARIABLES.

DELV	The increment in the New estimate of the specific volume using the newton method.
DIFF	The relative error in the last estimate of the first state
DIF1	The relative error in the last estimate of the second state variable, VI2, used in testing convergence.
DNOM	The Jakobian of the Helmholtz function.
ZDELTE	The increment in the new estimate of the temperature using the Newton method. variable, VI1, used in testing convergence.

(5) OTHER SUBROUTINES OR MODULES REQUIRED.

Subroutines: WASIN1, WASR21.
Common Blocks: CONST5, TIMER5, WASPP5.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

The Newton iterative method to calculate the temperature TNEW and the specific volume VNEW from a two dimensional set of thermodynamic relations is given by the following equations.

$$TNEW = TOLD + (F2 * F12 - F1 * F22) / (F11 * F22 - F12 * F21) \quad (1)$$

$$VNEW = VCLD + (F1 * F21 - F2 * F11) / (F11 * F22 - F12 * F21) \quad (2)$$

(7) ORGANIZATION OF MODULE.

The computations in this subroutine depend upon the value of IGO.

If IGO=1, a first estimate for TNEW and VNEW is requested if either TNEW or VNEW is less than zero. This estimate is obtained by calling WASIN1. Next, the Helmholtz function is calculated by WASR21.

If IGO=2, only the Helmholtz function is evaluated by calling WASR21.

If IGO=3, new values of the temperature and specific volume corresponding to the state variables VI1 and VI2 are estimated from Eqs. (1) and (2). The relative errors in the estimates are then calculated by,

$DIFF = |F1 * 100 / VI1|$ for the first state variable, and

$DIFF = |F2 * 100 / VI2|$ for the second state variable.

Then, if DIFF is less than TES1 and DIF1 is less than TES2, convergence is achieved and IFET is set to zero. If more than 20 iterations have been performed, an error message is printed and control is returned to the calling subroutine. If neither of the above happens, the WASR21 subroutine is called to calculate the Helmholtz function and its derivatives.

(8) APPLICABILITY AND RESTRICTIONS.

Values of VI1 and VI2 must correspond to single-phase water (liquid or vapor) at temperatures less than 1300 C and pressures less than 100 MPa. Approximate values of VO1 and VO2 must be provided when calling the program. The speed of convergence will depend upon the closeness of these values to the solution. To use this module in the simulation, the following statement should be included.

CALL WASPF1 (IGO,VI1,VI2,VO1,VO2,Z,Z1,Z2,Z11,Z12,Z22,L,IFET)

* WASPS1 * (VI1,VI2,VC1,VO2,Z,Z1,Z2,Z11,Z12,Z22,I)

(1) PURPOSE.

This subroutine computes the Helmholtz function for single-phase water (liquid or vapor) as a function of entropy and pressure.

(2) PARAMETERS.

I Number of iterations needed for solution.

VI1 Entropy, in J/kg.

VI2 Pressure, in Pa.

VC1 Temperature, in C.

VC2 Specific volume, in m³/kg.

Z The Helmholtz function.

Z1 The first derivative of the Helmholtz function w/r to temperature (constant specific volume).

Z11 The second derivative of the Helmholtz function w/r to temperature (constant specific volume).

Z12 The second derivative of the Helmholtz function w/r to temperature and specific volume.

Z2 The first derivative of the Helmholtz function w/r to specific volume (constant temperature).

Z22 The second derivative of the Helmholtz function w/r to specific volume (constant temperature).

(3) COMMON VARIABLES.

<u>NAMES</u>	<u>COMBIK</u>	<u>DESCRIPTION</u>
F1	WASPR	The difference between the estimated and given state variable VI1.
F11	WASPR	The derivative of F1 w/r to temperature with constant specific volume.

F12	WASPR	The derivative of F1 w/r to specific volume with constant temperature.
F2	WASPR	The difference between the estimated and given state variable VI2.
F21	WASPR	The derivative of F2 w/r to temperature with constant specific volume.
F22	WASPR	The derivative of F2 w/r specific volume with constant temperature.
IS2	WASPR	Numerical indicator to the WAS property routine being used.
TES1	WASPR	Upper error limit in steam-water property evaluations and usually set to 10E-4.
TES2	WASPR	Upper error limit in steam-water property evaluations and usually set to 10E-4.
TNEW	WASPR	Recent value of the temperature computed in one of the "WAS" routines, in C.
TOLD	WASPR	Temperature value of the previous iteration in the "WAS" routines, in C.
VNEW	WASPR	Recent value of the specific volume computed in one of the "WAS" routines, in m ³ /kg.
VOLD	WASPR	Specific volume of the previous iteration in the "WAS" routines, in m ³ /kg.

(4) ICCAL VARIABLES.

IRET An indicator that, if zero, indicates that convergence was achieved or that the maximum number of allowed iterations was exceeded.

T Absolute temperature equal to TOLD+273.15, in K.

(5) OTHER SUBROUTINES OR MODULES REQUIRED.

Subroutines: WASPR1.

Common Blocks: CONST5, TIMER5, WASPR5.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

Newton's approximation method is used to solve the implicit set of thermodynamic relations to obtain the Helmholtz function and its derivatives. The following relations are used in this subroutine.

$$F1 = -Z2 - VI1 \quad (1)$$

$$F2 = -Z1 - VI2 \quad (2)$$

$$F11 = -Z12 \quad (3)$$

$$F12 = -Z22 \quad (4)$$

$$F21 = -Z11 \quad (5)$$

$$F22 = -Z12 \quad (6)$$

(7) ORGANIZATION OF MODULE.

Upon entering the subroutine, TNEW and VNEW are set to the last known value of VO1 and VO2 respectively. Next, the convergence criteria TES1 and TES2 are set. These can be reset by the user to any desired value. Then, the WASPR1 subroutine is called to obtain the first approximation to the Helmholtz function and the first and second derivatives. These approximations are then used in Eqs. (1) to (6) to calculate the differences between the given and estimated values of the state variables, and the derivatives w/r to temperature and specific volume. The WASPR1 subroutine is called again to obtain the new estimates for the entropy and pressure using the differences and derivatives obtained in Eqs. (1) to (6). If convergence is not achieved or when IRET is not equal to zero, new values of the Helmholtz function are computed and Eqs. (1) to (6) are recalculated. The iterative process is terminated when IRET is equal to zero.

(8) APPLICABILITY AND RESTRICTIONS.

Values of VJ1 and VI2 must correspond to single-phase water (liquid or vapor) at temperatures less than 1300 C and pressures less than 100 MPa. Approximate values of

V01 and V02 must be provided when calling the program. The speed of convergence will depend upon the closeness of these values to the solution. Upon return, V01 and V02 will correspond to VI1 and VI2. To use this module in the simulation, the following statement should be included.

CALL WASPS1(VI1,VI2,V01,V02,Z,Z1,Z2,Z11,Z12,Z22,L)

REFERENCES.

- [1] J. H. Keenan, F. G. Keyes, P. G. Hill, and J. G. Moore, "Steam Tables", John Wiley, (1969).

* WASPT1 * (VI1,VI2,VO1,VO2,Z,Z1,Z2,Z11,Z12,Z22,L)

(1) PURPOSE.

This subroutine computes the Helmholtz function for single-phase water (liquid or vapor) as a function of pressure and temperature.

(2) PARAMETERS.

L Number of iterations needed for solution.

VI1 Pressure, in Pa.

VI2 Temperature, in C.

VO1 Temperature, in C.

VO2 Specific volume, in m³/kg.

Z The Helmholtz function.

Z1 The first derivative of the Helmholtz function w/r to temperature (constant specific volume).

Z11 The second derivative of the Helmholtz function w/r to temperature (constant specific volume).

Z12 The second derivative of the Helmholtz function w/r to temperature and specific volume.

Z2 The first derivative of the Helmholtz function w/r to specific volume (constant temperature).

Z22 The second derivative of the Helmholtz function w/r to specific volume (constant temperature).

(3) COMMON VARIABLES.

<u>NAMES</u>	<u>COMBLK</u>	<u>DESCRIPTION</u>
F1	WASPR	The difference between the estimated and given state variable VI1.
F11	WASPR	The derivative of F1 w/r to temperature with constant specific volume.

F12	WASPR	The derivative of F1 w/r to specific volume with constant temperature.
F2	WASPR	The difference between the estimated and given state variable VI2.
F21	WASPR	The derivative of F2 w/r to temperature with constant specific volume.
F22	WASPR	The derivative of F2 w/r specific volume with constant temperature.
IS2	WASPR	Numerical indicator to the WAS property routine being used.
TES1	WASPR	Upper error limit in steam-water property evaluations and usually set to 10E-4.
TFS2	WASPR	Upper error limit in steam-water property evaluations and usually set to 10E-4.
TNEW	WASPR	Recent value of the temperature computed in one of the "WAS" routines, in C.
TCLD	WASPR	Temperature value of the previous iteration in the "WAS" routines, in C.
VNEW	WASPR	Recent value of the specific volume computed in one of the "WAS" routines, in m ³ /kg.
VCLD	WASPR	Specific volume of the previous iteration in the "WAS" routines, in m ³ /kg.

(4) LOCAL VARIABLES.

IRET An indicator that, if zero, indicates that convergence was achieved or that the maximum number of allowed iterations was exceeded.

T Absolute temperature equal to TOLD+273.15, in K.

(5) OTHER SUBROUTINES OR MODULES REQUIRED.

Subroutines: WASPR1.

Common Blocks: CONST5, TIMER5, WASPP5.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

Newton's approximation method is used to solve the implicit set of thermodynamic relations to obtain the Helmholtz function and its derivatives. The following relations are used in this subroutine.

$$F1 = -Z2 - VI1 \quad (1)$$

$$F12 = -Z22 \quad (2)$$

(7) ORGANIZATION OF MODULE.

Upon entering the subroutine, TNEW and VNEW are set to the last known value of VO1 and VO2 respectively. Next, the convergence criteria TES1 and TES2 are set. These can be reset by the user to any desired value. Then, the WASPR1 subroutine is called to obtain the first approximation to the Helmholtz function and the first and second derivatives. These approximations are then used in Eqs. (1) and (2) to calculate the differences between the given and estimated values of the state variables, and the derivatives w/r to temperature and specific volume. The WASPR1 subroutine is called again to obtain the new estimates for the temperature and pressure using the differences and derivatives obtained in Eqs. (1) and (2). If convergence is not achieved or when IRET is not equal to zero, new values of the Helmholtz function are computed and Eqs. (1) and (2) are recalculated. The iterative process is terminated when IPET is equal to zero.

(8) APPLICABILITY AND RESTRICTIONS.

Values of VI1 and VI2 must correspond to single-phase water (liquid or vapor) at temperatures less than 1300 C and pressures less than 100 MPa. Approximate values of VO1 and VO2 must be provided when calling the program. The speed of convergence will depend upon the closeness of these values to the solution. Upon return, VO1 and VO2 will correspond to VI1 and VI2. To use this module in the simulation, the following statement should be included.

CALL WASPT1(VI1,VI2,VO1,VO2,Z,Z1,Z2,Z11,Z12,Z22,L)

* WASPU1 * (VI1,VI2,VC1,VO2,Z,Z1,Z2,Z11,Z12,Z22,L)

(1) PURPOSE.

This subroutine computes the Helmholtz function for single-phase water (liquid or vapor) as a function of pressure and internal energy.

(2) PARAMETERS.

L Number of iterations needed for solution.

VI1 Pressure, in Pa.

VI2 Internal energy, J/kg.

VC1 Temperature, in C.

VC2 Specific volume, in m³/kg.

Z The Helmholtz function.

Z1 The first derivative of the Helmholtz function w/r to temperature (constant specific volume).

Z11 The second derivative of the Helmholtz function w/r to temperature (constant specific volume).

Z12 The second derivative of the Helmholtz function w/r to temperature and specific volume.

Z2 The first derivative of the Helmholtz function w/r to specific volume (constant temperature).

Z22 The second derivative of the Helmholtz function w/r to specific volume (constant temperature).

(3) COMMON VARIABLES.

<u>NAMES</u>	<u>COMBLK</u>	<u>DESCRIPTION</u>
F1	WASPR	The difference between the estimated and given state variable VI1.
F11	WASPR	The derivative of F1 w/r to temperature with constant specific volume.

F12	WASPR	The derivative of F1 w/r to specific volume with constant temperature.
F2	WASPR	The difference between the estimated and given state variable VI2.
F21	WASPR	The derivative of F2 w/r to temperature with constant specific volume.
F22	WASPR	The derivative of F2 w/r specific volume with constant temperature.
IS2	WASPR	Numerical indicator to the WAS property routine being used.
TES1	WASPR	Upper error limit in steam-water property evaluations and usually set to 10E-4.
TES2	WASPR	Upper error limit in steam-water property evaluations and usually set to 10E-4.
TNEW	WASPR	Recent value of the temperature computed in one of the "WAS" routines, in C.
TOLD	WASPR	Temperature value of the previous iteration in the "WAS" routines, in C.
VNEW	WASPR	Recent value of the specific volume computed in one of the "WAS" routines, in m ³ /kg.
VOLD	WASPR	Specific volume of the previous iteration in the "WAS" routines, in m ³ /kg.

(4) LOCAL VARIABLES.

IRET An indicator that, if zero, indicates that convergence was achieved or that the maximum number of allowed iterations was exceeded.

T Absolute temperature equal to TOLD+273.15, in K.

(5) OTHER SUBROUTINES OR MODULES REQUIRED.

Subroutines: WASPR1.

Common Blocks: CONST5, TIMER5, WASPR5.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

Newton's approximation method is used to solve the implicit set of thermodynamic relations to obtain the Helmholtz function and its derivatives. The following relations are used in this subroutine.

$$F1 = -Z2 - VI1 \quad (1)$$

$$F2 = (Z - T*Z1) - VI2 \quad (2)$$

$$F11 = -Z12 \quad (3)$$

$$F12 = -Z22 \quad (4)$$

$$F21 = -T*Z11 \quad (5)$$

$$F22 = Z2 - T*Z12 \quad (6)$$

(7) ORGANIZATION OF MODULE.

Upon entering the subroutine, TNEW and VNEW are set to the last known value of VO1 and VO2 respectively. Next, the convergence criteria TES1 and TES2 are set. These can be reset by the user to any desired value. Then, the WASPR1 subroutine is called to obtain the first approximation to the Helmholtz function and the first and second derivatives. These approximations are then used in Eqs. (1) to (6) to calculate the differences between the given and estimated values of the state variables, and the derivatives w/r to temperature and specific volume. The WASPR1 subroutine is called again to obtain the new estimates for the internal energy and pressure using the differences and derivatives obtained in Eqs. (1) to (6). If convergence is not achieved or when IRET is not equal to zero, new values of the Helmholtz function are computed and Eqs. (1) to (6) are recalculated. The iterative process is terminated when IRET is equal to zero.

(8) APPLICABILITY AND RESTRICTIONS.

Values of VI1 and VI2 must correspond to single-phase water (liquid or vapor) at temperatures less than 1300 C and pressures less than 100 MPa. Approximate values of

VO1 and VO2 must be provided when calling the program. The speed of convergence will depend upon the closeness of these values to the solution. Upon return, VO1 and VO2 will correspond to VI1 and VI2. To use this module in the simulation, the following statement should be included.

CALL WASPU1(VI1,VI2,VO1,VO2,Z,Z1,Z2,Z11,Z12,Z22,L)

REFERENCES.

- [1] J. H. Keenan, F. G. Keyes, P. G. Hill, and J. G. Moore, "Steam Tables", John Wiley, (1969).

* WASPV1 * (VI1,VI2,VC1,VO2,Z,Z1,Z2,Z11,Z12,Z22,L)

(1) PURPOSE.

This subroutine computes the Helmholtz function for single-phase water (liquid or vapor) as a function of pressure and specific volume.

(2) PARAMETERS.

L Number of iterations needed for solution.

VI1 Pressure, in Pa.

VI2 Specific volume, in m³/kg.

VO1 Temperature, in C.

VC2 Specific volume, in m³/kg.

Z The Helmholtz function.

Z1 The first derivative of the Helmholtz function w/r to temperature (constant specific volume).

Z11 The second derivative of the Helmholtz function w/r to temperature (constant specific volume).

Z12 The second derivative of the Helmholtz function w/r to temperature and specific volume.

Z2 The first derivative of the Helmholtz function w/r to specific volume (constant temperature).

Z22 The second derivative of the Helmholtz function w/r to specific volume (constant temperature).

(3) COMMON VARIABLES.

<u>NAMES</u>	<u>COMBLK</u>	<u>DESCRIPTION</u>
F1	WASPR	The difference between the estimated and given state variable VI1.
F11	WASPR	The derivative of F1 w/r to temperature with constant specific volume.

F12	WASPR	The derivative of F1 w/r to specific volume with constant temperature.
F2	WASPR	The difference between the estimated and given state variable VI2.
F21	WASPR	The derivative of F2 w/r to temperature with constant specific volume.
F22	WASPR	The derivative of F2 w/r specific volume with constant temperature.
IS2	WASPR	Numerical indicator to the WAS property routine being used.
TES1	WASPR	Upper error limit in steam-water property evaluations and usually set to 10E-4.
TES2	WASPR	Upper error limit in steam-water property evaluations and usually set to 10E-4.
TNEW	WASPR	Recent value of the temperature computed in one of the "WAS" routines, in C.
TCID	WASPR	Temperature value of the previous iteration in the "WAS" routines, in C.
VNEW	WASPR	Recent value of the specific volume computed in one of the "WAS" routines, in m ³ /kg.
VOLD	WASPR	Specific volume of the previous iteration in the "WAS" routines, in m ³ /kg.

(4) LOCAL VARIABLES.

IFET An indicator that, if zero, indicates that convergence was achieved or that the maximum number of allowed iterations was exceeded.

T Absolute temperature equal to TOLD+273.15, in K.

(5) OTHER SUBROUTINES OR MODULES REQUIRED.

Subroutines: WASPR1.

Common Blocks: CONST5, TIMER5, WASPP5.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

Newton's approximation method is used to solve the implicit set of thermodynamic relations to obtain the Helmholtz function and its derivatives. The following relations are used in this subroutine.

$$F1 = -Z2 - VI1 \quad (1)$$

$$F11 = -Z12 \quad (2)$$

(7) ORGANIZATION OF MODULE.

Upon entering the subroutine, TNEW and VNEW are set to the last known value of VO1 and VO2 respectively. Next, the convergence criteria TES1 and TES2 are set. These can be reset by the user to any desired value. Then, the WASPR1 subroutine is called to obtain the first approximation to the Helmholtz function and the first and second derivatives. These approximations are then used in Eqs. (1) and (2) to calculate the differences between the given and estimated values of the state variables, and the derivatives w/r to temperature and specific volume. The WASPP1 subroutine is called again to obtain the new estimates for the pressure using the differences and derivatives obtained in Eqs. (1) and (2). If convergence is not achieved or when IRET is not equal to zero, new values of the Helmholtz function are computed and Eqs. (1) and (2) are recalculated. The iterative process is terminated when IRET is equal to zero.

(8) APPLICABILITY AND RESTRICTIONS.

Values of VI1 and VI2 must correspond to single-phase water (liquid or vapor) at temperatures less than 1300 C and pressures less than 100 MPa. Approximate values of VC1 and VC2 must be provided when calling the program. The speed of convergence will depend upon the closeness of these values to the solution. Upon return, VO1 and VO2 will correspond to VI1 and VI2. To use this module in the simulation, the following statement should be included.

CALL WASPV1(VI1,VI2,VO1,VO2,Z,Z1,Z2,Z11,Z12,Z22,L)

REFERENCES.

- [1] J. H. Keenan, F. G. Keves, P. G. Hill, and J. G. Moore,
"Steam Tables", John Wiley, (1969).

* WASR21 * (TCENT, V, F, DFDT, DFDV, DFDTT, DFDTV, DFDVV)

(1) PURPOSE.

This subroutine computes the Helmholtz function for single-phase water (liquid or vapor) as a function of temperature and specific volume.

(2) PARAMETERS.

DFDT First derivative of the Helmholtz function w/r to temperature (constant specific volume).

DFDTT Second derivative of the Helmholtz function w/r to temperature (constant specific volume).

DFDTV Second derivative of the Helmholtz function w/r to temperature and specific volume.

DFDV First derivative of the Helmholtz function w/r to specific volume (constant temperature).

DFDVT Second derivative of the Helmholtz function w/r to specific volume (constant temperature).

F Helmholtz function.

TCENT Temperature, in C.

V Specific volume, in m³/kg.

(3) COMMON VARIABLES.

None.

(4) LOCAL VARIABLES.

This is not a full list of the local variables.

A(10,7) Constants [1] used with the Helmholtz function.

AC(8) Constants [1] used with the Helmholtz function.

AIT Lca of T.

AIE Log of RH.
 DEO Derivative of P0 w/r to temperature.
 DTA The value of $-TA/T$.
 D2P0 Second derivative of P0 w/r to temperature.
 D2TA The value of $-2*DTA/T$.
 E The constant 4.8.
 ER The value of $\exp(-RH*E)$.
 P The Helmholtz function.
 EC1 A conversion factor, 0.000001.
 PC The first component of the Helmholtz function given by Eq. (2).
 Q The second component of the Helmholtz function given by Eq. (3).
 R The universal gas constant equal to 0.4615.
 RAJ(7) Constants [1] used with the Helmholtz function.
 RH Water or steam density given by $1/VN$, in kg/m^3 .
 S1 S2 S3 Partial sums of P0.
 T Temperature given by $TCENT+273.15$, in K.
 TA Normalized inverted temperature given by $1000/T$.
 TAD The value of $TA-TC$.
 TAJ(7) Constants [1] used with the Helmholtz function.
 TC The value of $1000/T_{crit}=1.54491$.
 TC1 The constant 1000.
 VN The value of $V*1000$.

(5) OTHER SUBROUTINES OR MODULES REQUIRED.

None.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

The Helmholtz equation as given in reference 1 are solved in this subroutine.

$$P = P_0 + R * T * (DLOG(RH) + RH * Q), \quad (1)$$

where

$$P_0 = \sum_{I=1}^6 [AC(I) / TA^{**}(I-1)] + AC(7) * DLOG(T) + AC(8) * DLOG(T) / TA, \quad (2)$$

$$Q = (TA - TC) * \sum_{J=1}^7 (TA - TAJ(J))^{**}(J-1) + \sum_{I=1}^8 A(I, J) * (RH - RAJ(J))^{**}(I-1) + DEXP(-E * RH) * ((A9, J) + A(10, j) * RH) \quad (3)$$

(7) ORGANIZATION OF MODULE.

Not available at the present time. The text will be supplied in the next revision of this document.

(8) APPLICABILITY AND RESTRICTIONS.

Values of VI1 and VI2 must correspond to single-phase water (liquid or vapor) at temperatures less than 1300 C and pressures less than 100 MPa. To use this module in the simulation, the following statement should be included.

CALL WASR21(VI1,VI2,Z,Z1,Z2,Z11,Z12,Z22)

REFERENCES.

- [1] J. H. Keenan, F. G. Keyes, P. G. Hill, and J. G. Moore, "Steam Tables", John Wiley, (1969).

* WASTD1 * (PSF,TCENT)

(1) PURPOSE.

This subroutine provides a relationship for the specific volume of superheated steam as a function of pressure and temperature, WASTD1, and a relationship for the specific volume of subcooled water as a function of pressure and temperature, WASTL1.

(2) PARAMETERS.

PSF Pressure, in Pa.

TCENT Temperature, in C.

(3) COMMON VARIABLES.

<u>NAMES</u>	<u>COMBLK DESCRIPTION</u>
--------------	---------------------------

Not available at the present time. The text will be supplied in the next revision of this document.

(4) ICCAI VARIABLES.

Not available at the present time. The text will be supplied in the next revision of this document.

(5) OTHER SUBROUTINES OR MODULES REQUIRED.

Common Blocks: CONST5, TIMER5.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

Not available at the present time. The text will be supplied in the next revision of this document.

(7) ORGANIZATION OF MODULE.

Not available at the present time. The text will be supplied in the next revision of this document.

(8) APPLICABILITY AND RESTRICTIONS.

Applicability is limited to either temperatures less than 374.15 C. and pressures less than or equal to the saturation value or temperatures between 374.15 and 800.0 C. and pressures below 100 MPa.

REFERENCES.

[1] 1968 IFC Formulation For Scientific and General Use.

* WASUV1 * (VI1,VI2,VC1,VO2,Z,Z1,Z2,Z11,Z12,Z22,L)

(1) PURPOSE.

This subroutine computes the Helmholtz function for single-phase water (liquid or vapor) as a function of internal energy and specific volume.

(2) PARAMETERS.

I Number of iterations needed for solution.

VI1 Internal energy, J/kg.

VI2 Specific volume, in m³/kg.

VC1 Temperature, in C.

VO2 Specific volume, in m³/kg.

Z The Helmholtz function.

Z1 The first derivative of the Helmholtz function w/r to temperature (constant specific volume).

Z11 The second derivative of the Helmholtz function w/r to temperature (constant specific volume).

Z12 The second derivative of the Helmholtz function w/r to temperature and specific volume.

Z2 The first derivative of the Helmholtz function w/r to specific volume (constant temperature).

Z22 The second derivative of the Helmholtz function w/r to specific volume (constant temperature).

(3) COMMON VARIABLES.

<u>NAMES</u>	<u>COMBLK</u>	<u>DESCRIPTION</u>
F1	WASPR	The difference between the estimated and given state variable VI1.
F11	WASPR	The derivative of F1 w/r to temperature with constant specific volume.

F12	WASPR	The derivative of F1 w/r to specific volume with constant temperature.
F2	WASPR	The difference between the estimated and given state variable VI2.
F21	WASPP	The derivative of F2 w/r to temperature with constant specific volume.
F22	WASPR	The derivative of F2 w/r specific volume with constant temperature.
IS2	WASPR	Numerical indicator to the WAS property routine being used.
TES1	WASPR	Upper error limit in steam-water property evaluations and usually set to 10E-4.
TES2	WASPR	Upper error limit in steam-water property evaluations and usually set to 10E-4.
TNEW	WASPR	Recent value of the temperature computed in one of the "WAS" routines, in C.
TCID	WASPR	Temperature value of the previous iteration in the "WAS" routines, in C.
VNEW	WASPR	Recent value of the specific volume computed in one of the "WAS" routines, in m ³ /kg.
VCID	WASPR	Specific volume of the previous iteration in the "WAS" routines, in m ³ /kg.

(4) ICCAI VARIABLES.

IFET An indicator that, if zero, indicates that convergence was achieved or that the maximum number of allowed iterations was exceeded.

T Absolute temperature equal to TOLD+273.15, in K.

(5) OTHER SUBROUTINES OR MODULES REQUIRED.

Subroutines: WASPR1.

Common Blocks: CONST5, TIMERS5, WASPF5.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

Newton's approximation method is used to solve the implicit set of thermodynamic relations to obtain the Helmholtz function and its derivatives. The following relations are used in this subroutine.

$$F1 = (Z - T \cdot Z1) - VI1 \quad (1)$$

$$F11 = -T \cdot Z11 \quad (2)$$

(7) ORGANIZATION OF MODULE.

Upon entering the subroutine, TNFW and VNEW are set to the last known value of VO1 and VO2 respectively. Next, the convergence criteria TES1 and TES2 are set. These can be reset by the user to any desired value. Then, the WASPR1 subroutine is called to obtain the first approximation to the Helmholtz function and the first and second derivatives. These approximations are then used in Eqs. (1) and (2) to calculate the differences between the given and estimated values of the state variables, and the derivatives w/r to temperature and specific volume. The WASPR1 subroutine is called again to obtain the new estimates for the internal energy using the differences and derivatives obtained in Eqs. (1) and (2). If convergence is not achieved or when IRET is not equal to zero, new values of the Helmholtz function are computed and Eqs. (1) and (2) are recalculated. The iterative process is terminated when IRET is equal to zero.

(8) APPLICABILITY AND RESTRICTIONS.

Values of VI1 and VI2 must correspond to single-phase water (liquid or vapor) at temperatures less than 1300 C and pressures less than 100 MPa. Approximate values of VO1 and VO2 must be provided when calling the program. The speed of convergence will depend upon the closeness of these values to the solution. Upon return, VO1 and VO2 will correspond to VI1 and VI2. To use this module in the simulation, the following statement should be included.

CALL WASUV1(VI1,VI2,VO1,VO2,Z,Z1,Z2,Z11,Z12,Z22,L)

REFERENCES.

- [1] J. H. Keenan, F. G. Keyes, F. G. Hill, and J. G. Moore,
"Steam Tables", John Wiley, (1969).

* WBHXP1 * (ZRE,ZPR,P,CK,D,RI,RX,X,VF,VG,U)

(1) PURPOSE.

This subroutine calculates the gap heat transfer coefficient in a boiler (film boiling region) assuming that the boiling takes place outside the tube or on the shell side.

(2) PARAMETERS.

CK Tube conductivity, in W/m-C.
D Hydraulic diameter, in m.
P The steam pressure, in Pa.
G Tube to external channel gap heat transfer coefficient, in W/m²-C.
VF Water volume fraction in the two phase steam-water flow.
VG Steam volume fraction in the two phase steam-water flow.
X Steam quality.
ZPR Prandtl number (Dimensionless).
ZRE Reynolds number (Dimensionless).

(3) COMMON VARIABLES.

None.

(4) LOCAL VARIABLES.

None.

(5) OTHER SUBROUTINES OR MODULES REQUIRED.

Functions: UPIEX1.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

The Nusselt number is given by,

$$\text{ZNU} = 0.0193 * \text{ZFE} ** 0.8 * \text{ZPR} ** 1.23 * (\text{X} + (1 - \text{X}) * \text{VF} / \text{VG}) ** 0.68 \\ * (\text{VF} / \text{VG}) ** 0.068 \quad (1)$$

(7) ORGANIZATION OF MODULE.

Upon entering the subroutine, Eq. (1) is solved to obtain the Nusselt number. Next, the function UPIEX1 is used to obtain the heat transfer coefficient U.

(8) APPLICABILITY AND RESTRICTIONS.

The heat transfer coefficient can be used for straight tube counterflow boilers. To use this module in the simulation, the following statement should be included.

CALL WBHXP1(ZFE,ZPR,P,CK,D,RT,RI,RX,X,VF,VG,U)

* WBHXS1 * (ZRE, ZPR, P, CK, D, RI, RX, X, VF, VG, U)

(1) PURPOSE.

This subroutine calculates the gap heat transfer coefficient in a steam generator (film boiling region) inside the tube or on the secondary side.

(2) PARAMETERS.

CK Tube conductivity, in W/m-C.
D Hydraulic diameter, in m.
P The steam pressure, in Pa.
U Tube to inner channel heat transfer coefficient, in W/m²-C.
VF Water volume fraction in the two phase steam-water flow.
VG Steam volume fraction in the two phase steam-water flow.
X Steam quality.
ZPR Prandtl number (Dimensionless).
ZRE Reynolds number (Dimensionless).

(3) COMMON VARIABLES.

None.

(4) LOCAL VARIABLES.

None.

(5) OTHER SUBROUTINES OR MODULES REQUIRED.

Functions: UPIIN1.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

The Nusselt number is given by,

$$ZNU = 0.0193 * ZPE^{**0.8} * ZPR^{**1.23} * (X + (1-X) * VF/VG)^{**0.68} * (VF/VG)^{**0.068} \quad (1)$$

(7) ORGANIZATION OF MODULE.

Upon entering the subroutine, Eq. (1) is solved to obtain the Nusselt number. Next, the function UPIEX1 is used to obtain the heat transfer coefficient U. The hydraulic diameter is taken as 2*RI.

(8) APPLICABILITY AND RESTRICTIONS.

The heat transfer coefficient can be used for straight tube counterflow boilers. To use this module in the simulation, the following statement should be included.

CALL WBHXS1(ZFE,ZPR,P,CK,D,PI,PX,X,VF,VG,U)

* WSATP1 * (VI1,VI2,VC1,VC2,Z,Z1,Z2,Z11,Z12,Z22,L)

(1) PURPOSE.

This subroutine calculates the specific volume of subcooled water as a function of pressure and temperature.

(2) PARAMETERS.

L Number of iterations needed for solution.

VI1 Internal energy, in J/kg.

VI2 Specific volume, in m³/kg.

VC1 Temperature, in C.

VC2 Specific volume, in m³/kg.

Z The Helmholtz function.

Z1 The first derivative of the Helmholtz function w/r to temperature (constant specific volume).

Z11 The second derivative of the Helmholtz function w/r to temperature (constant specific volume).

Z12 The second derivative of the Helmholtz function w/r to temperature and specific volume.

Z2 The first derivative of the Helmholtz function w/r to specific volume (constant temperature).

Z22 The second derivative of the Helmholtz function w/r to specific volume (constant temperature).

(3) COMMON VARIABLES.

<u>NAMES</u>	<u>COMBLK</u>	<u>DESCRIPTION</u>
F1	WASPF	The difference between the estimated and given state variable VI1.
F11	WASPR	The derivative of F1 w/r to temperature with constant specific volume.

F12	WASPR	The derivative of F1 w/r to specific volume with constant temperature.
F2	WASPR	The difference between the estimated and given state variable VI2.
F21	WASPR	The derivative of F2 w/r to temperature with constant specific volume.
F22	WASPR	The derivative of F2 w/r specific volume with constant temperature.
IS2	WASPR	Numerical indicator to the WAS property routine being used.
TFS1	WASPR	Upper error limit in steam-water property evaluations and usually set to 10E-4.
TFS2	WASPR	Upper error limit in steam-water property evaluations and usually set to 10E-4.
TCID	WASPR	Temperature value of the previous iteration in the "WAS" routines, in C.
VCID	WASPR	Specific volume of the previous iteration in the "WAS" routines, in m ³ /kg.

(4) LOCAL VARIABLES.

IFET An indicator that, if zero, indicates that convergence was achieved or that the maximum number of allowed iterations was exceeded.

PCALC Estimate for pressure, in Pa.

O Derivative of pressure w/r to temperature.

(5) OTHER SUBROUTINES OR MODULES REQUIRED.

Subroutines: WASAP1, WASPR1.

Common Blocks: CONST5, TIMER5, WASPR5.

(6) MATHEMATICAL EQUATIONS AND ALGORITHM.

Newton's approximation method is used to solve the implicit set of thermodynamic relations to obtain the Helmholtz function and the derivatives. The following relations are used in this subroutine.

$$PCALC = -72 \quad (1)$$

$$F1 = VI1 - PCALC \quad (2)$$

$$F11 = 0 + Z12 \quad (3)$$

$$F12 = 722. \quad (4)$$

(7) ORGANIZATION OF MODULE.

Upon entering the subroutine, TOLD and VOLD are set to the last known value of VO1 and VO2 respectively. Next, the convergence criteria TES1 and TES2 are set. These can be reset by the user to any desired value. The WASPR1 subroutine is called to obtain the first approximation to the Helmholtz function and the first and second derivatives followed by a call to the saturation line subroutine WASAP1. These values are then used in Eqs. (1) to (4) to calculate the differences between the given and estimated values of the state variables and the derivatives w/r to temperature and specific volume. Then, the WASPR1 subroutine is called again to obtain new estimates for the pressure and internal energy using the differences and derivatives obtained in Eqs. (1) to (4). If convergence is not achieved, IRET is not equal to zero and new values of the Helmholtz function is computed and Eqs. (1) to (4) are recalculated. The iterative process is terminated when IRET is equal to zero.

(8) APPLICABILITY AND RESTRICTIONS.

Applicability is limited to temperatures between 0.01 and 340 C. and pressures between saturation and 100 MPa.

REFERENCES.

[1] 1968 IFC Formulation For Scientific and General Use.

APPENDIX A: LIST OF SYMBOLS IN THE SYSTEM COMMON BLOCKS.

Whenever a "*" appears in the name of a variable, it indicates that it belongs to a macro. When the name is actually used in the simulation program, the "*" will be replaced by an appropriate character chosen by the user as part of the module name.

<u>NAMES</u>	<u>COMBLK</u>	<u>DESCRIPTION</u>
AR*P	THX**	Total IHX primary cross sectional flow area (shell side), in m.
BETA(I)	NEUTR	Delayed neutron fraction, I=1...6.
BGM(I)	GAMAR	Fraction of the fission product produced by the Ith group.
B*W	IHX**	Thermal conductivity of the IHX tube, in W/m-C.
CC	CORFP	Primary coolant specific heat, in J/kg-C.
CF	CORFP	Core fuel specific heat, in J/kg-C.
CL	CORFP	Fuel cladding specific heat, in J/kg-C.
CLJ(I)	NEUTR	Delayed precursor concentration at the Jth time step, I=1...6.
CM	IPLFN	Specific heat of the metal in the lower plenum, in J/kg-C.
CN	UPLFN	Specific heat of the metal in the upper plenum, in J/kg-C.
C*W	IHX**	Specific heat of the IHX tube metal, in J/kg-C.
DCL(I)	NEUTR	Derivative of the delayed neutron precursors, I=1...6.
DELT	TIMER	Current integration time step, in s.
DELTN	TIMER	Master simulation time step, in s. Integration output must fit this step.
DPGMA(I)	GAMAR	Time derivative of PGMA.
DREAK	NEUTR	Reactivity change during the current time step.
DTMAX	TIMER	Maximum time step allowed, in s.
DTMIN	TIMER	Minimum time step allowed, in s.

DYDT(I)	DERIV	Vector of the derivatives to be integrated, integrator input.
DYDTM1(I)	DEFIV	Vector storing past derivatives.
DYDTM2(I)	DERIV	Vector storing past derivatives.
DYDTM3(I)	DERIV	Vector storing past derivatives.
DYDT1(I)	DEFIV	Intermediate or past derivative vector.
DYDT2(I)	DEFIV	Intermediate or past derivative vector.
DYDT3(I)	DERIV	Intermediate or past derivative vector.
DYDT4(I)	DERIV	Intermediate or past derivative vector.
DYDT5(I)	DEFIV	Intermediate or past derivative vector.
EL	NFUTR	Prompt neutron life-time, in s.
ELAMDA(I)	NFUTR	Delayed precursor decay constants, I=1...6.
EPSN	NFUTR	Integration step error in NEUTS1.
EPSN1	NEUTR	Lower integration error limit in NEUTS1. Double time step if EPSN is less than EPSN1.
EPSN2	NEUTR	Upper integration error limit in NEUTS1. Halve time step if EPSN is larger than EPSN2.
ERRMES(I)	TIMER	A vector of words to store simulation error messages.
EXREC(I)	CNTRL	External reactivity inserted to initiate a transient, in absolute units.
EYYYY1(I)	DEFIV	Maximum relative integration error allowed, I=1 to NODER.
EYYYY2(I)	DEFIV	Maximum absolute integration error allowed, I=1 to NODER.
EYYYY3(I)	DEFIV	Evaluated integration error for the Ith integrated variable, I=1 to NODER.
E#PSHA	IHX#	Maximum absolute error allowed in the IHX temperature calculation.
E#PSHR	IHX#	Maximum relative error allowed in the temperature calculation of IHX.

F1	WASPR	The difference between the estimated and given state variable VI1.
F11	WASPR	The derivative of F1 w/r to temperature with constant specific volume.
F12	WASPR	The derivative of F1 w/r to specific volume with constant temperature.
F2	WASPR	The difference between the estimated and given state variable VI2.
F21	WASPR	The derivative of F2 w/r to temperature with constant specific volume.
F22	WASPR	The derivative of F2 w/r specific volume with constant temperature.
F#HI	INX#	Secondary INX flow, in kg/s.
F#FI	INX#	Primary INX flow, in kg/s.
GA	GAMAR	Fraction of the decay heat liberated in the coolant.
GAM(I)	GAMAR	Disintegration constant of the Ith fission product.
HDTMIN	TIMER	Half of the minimum time step size.
IDATA	TIMER	Input data set for the simulation.
IDIV	TIMER	If set to one, indicates one or more time step halving by an integration routine.
IDUB	TIMER	If set to one, indicates that the integration requests doubling of the time step.
IEXR	CNTPL	Indicates which type of external reactivity inserted. 1-step, 2-ramp, 3-function.
IFORT	TIMER	Not used.
IGAM	GAMAR	Number of radioactive FP decay groups assumed in the model.
ILSTEP(I)	DEPIV	Number of subimesteps in the Ith loop to integrate from TIME to TIME+DELT.
ILSTPS(I)	DEPIV	Number of time steps during which DELT=DELT.
IO5	TIMER	Standard FORTRAN input unit (usually 5).

IO6	TIMER	Standard FORTRAN output unit (usually 6).
ISCR1	TIMER	Not used.
ISCR2	TIMER	Not used.
ISCR3	TIMER	Not used.
IS2	WASPR	Numerical indicator to the WAS property routine being used.
ITERM	TIMER	If less than one, indicates that a simulation execution time error has been encountered.
ITI#MX	IHXM#	Maximum number of inner iterations in the IHX initial temperature calculation.
ITC#MX	IHXM#	Maximum number of external iterations in the IHX initial temperature calculation.
IYyyy(I)	DEPIV	Integration indicator. If not equal to LOOP, no integration will be performed.
JTIM	TIMER	Time step counter.
KDEGFO	CNTRL	Interpolation level to be used with external reactivity function.
KREFUN	CNTRL	Number of values in the given reactivity function.
LETM	TIMER	If one, use fixed time step. If two, use variable time step.
LIB1	TIMER	Not used.
LIB2	TIMER	Not used.
LIB3	TIMER	Not used.
LIB4	TIMER	Not used.
LOCPS	DERIV	Number of loops in the simulation.
ISTR	TIMER	Not used.
NODER	DERIV	Number of integrations to be performed in the simulation.
PGMA(I)	GAMAR	Fraction of the decay heat produced by the Ith FP group.
PNE	NEUTR	Not used.

PWJ	NEUTR	Normalized neutronic power at the Jth time step.
PNC	NEUTR	Normalized initial neutronic power level.
PWJ	TPGWR	Thermal power generated by the reactor, in W.
PW0	TPGWR	Initial PWJ, in W.
QLPF	IPLEN	Weight of metal in the lower plenum, in kg.
OMF	CORTP	Total weight of fuel in the core, in kg.
QML	CORTP	Total weight of cladding in the core, in kg.
QUFF	UPLEN	Weight of metal in the upper plenum, in kg.
BEAK	NEUTR	Total core excess reactivity.
RKCN	CNTFL	Reactivity generated by the control system.
RKD	FDPFK	Doppler reactivity feedback.
RKDC	FDBEK	Coefficient of doppler reactivity.
RKE	FDBEK	Core expansion reactivity feedback.
PKEC	FDBEK	Coefficient of core expansion reactivity.
RKFF	FLBPK	Total feedback reactivity.
RKS	FDBEK	Sodium density reactivity feedback.
RKSC	FDBEK	Coefficient of sodium expansion reactivity.
RKSF	SAFTY	Reactivity generated by the safety system.
RKV	FDPFK	Core void reactivity.
RKVC	FDBEK	Core void reactivity coefficient.
ROCA	CORTP	Coolant average density in the core, in kg/m ³ .
ROCM	IPLEN	Coolant average density in the lower plenum, in kg/m ³ .
ROCN	UPLEN	Coolant average density in the upper

plenum, in kg/m^3 .

R#W	IXM#	IX tube metal density, in kg/m^3 .
SOURCE	NFUTR	Value of neutron source in core startup simulations.
SUMC	NEUTR	Sum of $\text{CLJ(I)} * \text{ELAMDA(I)}$ for $\text{I}=1 \dots 6$.
TCNTRL	TIMFR	Time at which the change in external control reactivity is activated.
TCPU	TIMFR	CPU time required for orderly termination of the simulation and initially set to five seconds.
TEBPR#	BCIL#	Boiler primary fluid boiling region temperature, in C.
TEBPI#	BCIL#	Boiler primary side fluid inlet temperature, in C.
TEBPN#	BCIL#	Boiler primary fluid subcooled region temperature, in C.
TEBPX#	BCIL#	Boiler primary side exit temperature, in C.
TEBWB#	BCIL#	Boiler wall boiling region temperature, in C.
TEBWN#	BCIL#	Boiler wall nonboiling region temperature, in C.
TECA	CORTP	Core coolant average temperature, in C.
TECAO	CORTP	Initial TECA, in C.
TECI	CORTP	Core coolant inlet temperature, in C.
TECX	CORTP	Core coolant exit temperature, in C.
TEFA	CORTP	Core fuel average temperature, in C.
TEFAO	CORTP	Initial TEFA, in C.
TELA	CORTP	Core cladding average temperature, in C.
TEMA	LPLEN	Average coolant temperature in the lower plenum, in C.
TEMFA	LPLEN	Average metal temperature in the lower plenum, in C.
TEMI	LPIFN	Lower plenum inlet temperature, in C.

TFMX	IPLFN	Lower plenum exit temperature, in C.
TENA	UPIFN	Average coolant temperature in the upper plenum, in C.
TENFA	UPLFN	Average metal temperature in the upper plenum, in C.
TFNI	UPLFN	Upper plenum inlet temperature, in C.
TENX	UPLFN	Upper plenum coolant exit temperature, in C.
TEPSN	TIMFR	EPSN/EPSN1.
TES1	WASPR	Upper error limit in steam-water property evaluations and usually set to 10E-4.
TES2	WASPR	Upper error limit in steam-water property evaluations and usually set to 10E-4.
TE#H(I)	IHXM#	IHX secondary side temperature distribution, in C.
TE#HI	IHXM#	IHX secondary coolant inlet temperature, in C.
TE#HX	IHXM#	IHX secondary coolant exit temperature, in C.
TE#P(I)	IHXM#	Primary IHX coolant temperature, in C.
TE#PI	IHYM#	IHX primary sodium inlet temperature, in C.
TE#PX	IHXM#	IHX primary exit temperature, in C.
TE#W(I)	IHXM#	IHX tube wall temperature distribution, in C.
TICPU	TIMFR	Amount of CPU time left for orderly program termination, in s.
TIMEF	TIMFR	Current simulation time, in s.
TIMEDE	TIMFR	At this time, printing of each time step ends, in s.
TIMFDS	TIMFR	At this time, printing of requested variables starts at each time step.
TIMEF	TIMFR	Time at which the simulation is terminated, in s.
TIMEI	TIMFR	Simulation start time, in s.

TIMEM	TIMFR	Master simulation time, in s.
TNEW	WASPR	Recent value of the temperature computed in one of the "WAS" routines, in C.
TOLD	WASPR	Temperature value of the previous iteration in the "WAS" routines, in C.
TPRIN	TIMFR	Time at which the next printing of the requested variables should take place, in s.
TSCRAM	SAFTY	Time at which reactor SCRAM is activated, in s.
UCA	CORTP	Total heat transfer between the fuel and the cladding, in W/C.
ULA	CORTP	Total heat transfer between the fuel cladding and the core coolant, in W/C.
UMA	LPLEN	Total heat transfer from metal to coolant in the lower plenum, in W/C.
UNA	UPLEN	Total heat transfer from metal to coolant in the upper plenum, in W/C.
VC	CORTP	Volume of coolant in the core, in m ³ .
VMC	LPLEN	Volume of coolant in the lower plenum, in m ³ .
VNC	UPLEN	Volume of coolant in the upper plenum, in m ³ .
VNEW	WASPR	Recent value of the specific volume computed in one of the "WAS" routines, in m ³ /kg.
VOLD	WASPR	Specific volume of the previous iteration in the "WAS" routines, in m ³ /kg.
WGN	GAMAR	Total normalized radiation power produced by the fission products.
WGP	GAMAR	Power produced by the decay of fission products, in W.
WGPO	GAMAR	Initial WGP, in W.
WGT	GAMAR	Power produced by the decay of fission products, in W.
WGTO	GAMAR	Initial WGT, in W.

X*RH	IXM#	IX tube inner radius, in m.
X*RP	IXM#	IX tube external radius, in m.
X*WL	IXM#	IX tube length, in m.
YYYY(I)	DEFIV	Vector of the integrated variables, integrator output.
YYYYM1(I)	DEFIV	A vector storing past values of the Ith integrated variable.
YYYYM2(I)	DEFIV	A vector storing past values of the Ith integrated variable.
YYYYM3(I)	DEFIV	A vector storing past values of the Ith integrated variable.
YYYY1(I)	DEFIV	Intermediate or past integration results.
YYYY2(I)	DEFIV	Intermediate or past integration results.
YYYY3(I)	DEFIV	Intermediate or past integration results.
YYYY4(I)	DEFIV	Intermediate or past integration results.
YYYY5(I)	DEFIV	Intermediate or past integration results.
ZDELT(I)	TIMFR	The time step DELT of the Ith integration loop, in s.
ZDPFL(I)	HFLOW	Total pressure drop across a flow path, in Pa.
ZDPH(I)	HPIPE	Frictional and static pressure drop across a pipe, in Pa.
ZDXH#	HCA**	Level change due to flow variations, in m.
ZEGIB#	ECII#	Power transmitted into the boiler primary side, in W.
ZEGID#	DFUM#	Power transmitted into the steam drum, in W.
ZEGII#	IXM#	Power transmitted into the heat exchanger primary, in W.
ZEGIX#	IXM#	Power leaving the heat exchanger secondary side, in W.
ZEGXB#	ECII#	Power leaving the boiler secondary side, in W.

ZEGXC	CORTP	Power leaving the core through the coolant, in W.
ZEGXD#	DRUM#	Power leaving the steam drum, in W.
ZFALF(I)	HPIPE	A constant in the pipe friction factor calculation. Use 1 for sodium, 2 for water, or 3 for steam.
ZFEP#	BOII#	Boiler primary side flow, in kg/s.
ZFD#	BOIL#	Boiler secondary or downcomer flow, in kg/s.
ZFDCN#	FDWT#	The feedwater controller integration part.
ZFDRI#	DRUM#	Steam drum steam-water mixture inlet flow, in kg/s.
ZFDRX#	DRUM#	Steam drum exit flow to the downcomer, in kg/s.
ZFDW#	DRUM#	Feedwater inlet flow, in kg/s.
ZFICOR	CORTP	Flow through the core, in kg/s.
ZFLOR(I)	HPUMP	Rated flow through the pump, in kg/s.
ZFLOW(I)	HFLOW	Magnitude of the flow in the Ith flow path, in kg/s.
ZFR#	BOIL#	Boiler secondary riser flow, in kg/s.
ZFRK(I)	HPIPE	A constant in the pipe friction factor calculation. Use 1 for sodium, 2 for water, or 3 for steam.
ZFSTX#	DRUM#	Steam exit flow, in kg/s.
ZHBI#	BOIL#	Secondary coolant inlet enthalpy, in J/kg.
ZHESX#	BOIL#	Boiler exit enthalpy, in J/kg.
ZHCX	CORTP	Core exit enthalpy, in J/kg.
ZHDRI#	DRUM#	Steam drum inlet enthalpy, in J/kg.
ZHDRX#	DRUM#	Water average and exit enthalpy in the steam drum, in J/kg.
ZHFDW#	DRUM#	Feedwater inlet enthalpy, in J/kg.
ZHRPU(I)	HPUMP	Rated head generated by the pump at rated

flow, in m.

ZHRZF(I)	HPUMP	Rated head produced by the pump at zero flow and rated speed, in m.
ZHSTX#	DFUM#	Steam drum steam exit enthalpy, in J/kg.
ZINERF(I)	HFLOW	Equivalent inertance used in Eq. (1), in FLINT1, in 1/m.
ZINERT(I)	HPIPE	Pipe inertance, in 1/m.
ZIEUME(I)	HPUMP	Pump inertia, in kg/s.
ZBBP#	BCII#	Boiler primary side pressure, in Pa.
ZPES#	BOIL#	Boiler secondary side pressure, in Pa.
ZPCORA	CORTP	Average pressure in core, in Pa.
ZPDR#	DRUM#	Pressure in the steam drum, in Pa.
ZPDRX#	DRUM#	Pressure in steam drum at exit to downcomer, in Pa.
ZPH#A	IHXM#	Heat exchanger secondary side average pressure, in Pa.
ZPPLE#	MYPI#	Pressure in a mixing plenum, in Pa.
ZPP#A	IHXM#	Primary side average pressure, in Pa. Important only when water is the working fluid.
ZPTRIP	HPUMP	A motor torque multiplier, that if becomes smaller than one during the simulation, causes the pump to trip.
ZPUMAT(I)	HPUMP	Material indicator of the fluid flowing through the pump.
ZPU1(I)	HPUMP	Constant in the pump efficiency calculation.
ZPU2(I)	HPUMP	Constant in the pump efficiency calculation.
ZQDS#	DRUM#	Mass of steam in the steam drum, in kg.
ZQDW#	DRUM#	Mass of water in the steam drum, in kg.
ZQHPIP(I)	HPIPE	Mass of fluid in the pipe, in kg.
ZREPX#	BCII#	Boiler primary coolant exit density, in kg/m ³ .

ZRESX#	BOIL#	Boiler steam water mixture exit density, in kg/m^3 .
ZREV(I)	HPUMP	Speed of the pump, in radians/s.
ZREVR(I)	HPUMP	Rated speed of the pump, in radians/s.
ZROCK	CORTP	Core exit coolant density, in kg/m^3 .
ZRODS#	DRUM#	Steam drum steam density, in kg/m^3 .
ZRODW#	DRUM#	Steam drum water density, in kg/m^3 .
ZROHX#	IHX#	Heat exchanger secondary exit density, in kg/m^3 .
ZROPU(I)	HPUMP	Fluid inlet density to the pump, in kg/m^3 .
ZROPX#	IHX#	Heat exchanger primary side fluid exit density, in kg/m^3 .
ZRPIE#	MXPI#	Mixing plenum fluid density, in kg/m^3 .
ZTEBS#	BOIL#	Boiler secondary side saturation temperature, in C.
ZTECBI	CORTP	Core coolant boiling temperature, in C.
ZTED#	BOIL#	Boiler water inlet temperature, in C.
ZTEDR#	DRUM#	Steam drum steam exit temperature, in C.
ZTEPIN(I)	HPUMP	Pump fluid inlet temperature, in C.
ZTDRX#	DRUM#	Steam drum water exit temperature, in C.
ZTIMEI(I)	TIMER	The recent time in the Ith integration loop, in s.
ZTM#A	MXPI#	Mixing plenum average inlet temperature, in C.
ZTORP(I)	HPUMP	Pump torque, in N-m.
ZTORPR(I)	HPUMP	Rated pump torque, N-m.
ZTPF#	MXPI#	Mixing plenum fluid average temperature, in C.
ZTPM#	MXPI#	Mixing plenum metal average temperature, in C.
ZVOLFC	CORTP	Core coolant void volume fraction.

ZXEB#	BCII#	The length of the boiler boiling region, in m.
ZXDL#	DRUM#	Initial water level in the steam drum, in m.
ZXHC#	HCA##	Fluid level in a cavity, in m.
ZXNB#	BCII#	Boiler subcooled part length, in m.
ZXQX#	BOIL#	Boiler exit steam quality.
ZZZAST	CONST	The text "0*****".
ZZZEP	CONST	1.D-4, default error limit to many operations.
ZZZNUM	CONST	The text "0*****".
ZZZPLS	CCNST	The text "0++++++".
ZZZSO	CONST	"SO" indicating sodium properties.
ZZZSS	CONST	"SS" indicating stainless steel properties.
ZZZST	CCNST	"ST" indicating steam properties.
ZZZWA	CONST	"WA" indicating water properties.
ZZZC	CONST	0.D0
ZZZ01	CONST	1.D-1
ZZZ10	CONST	1.D1
ZZZ1	CONST	1.D0
ZZZ2	CONST	2.D0
ZZZ273	CONST	273.15, 0 C. in K.
ZZZ3	CONST	3.D0
ZZZ314	CONST	3.141592654D0
ZZZ4	CONST	4.D0
ZZZ5	CONST	5.D0
ZZZ6	CONST	6.D0
ZZZ7	CONST	7.D0
ZZZ8	CONST	7.D0

ZZZ9	CONST	9.D0
ZZZ981	CCNST	9.81D0
Z#PDI	IHM#	Not used.

APPENDIX B: LISTING OF THE SYSTEM COMMON BLOCKS.

Whenever a "*" appears in the name of a variable, it indicates that it belongs to a macro. When the name is actually used in the simulation program, the "*" will be replaced by one or two characters chosen by the user as part of the module name. Whenever a "%" appears as the dimension of a vector it will be replaced by the actual vector size as determined by the system. Only part of the DERIV5 COMMON block shown below is usually present in the simulation, depending on what integration method is used. The COMMON blocks are listed here in DSNP form and in the same order as they are in the DSNP modules.

```
COMMON /BCIL#5/ ZPBS#,ZTEBS#,TEBPI#,TEBPX#,ZBPB#,ZFD#,
ZFBP#,TEBWB#,TEBWN#,ZXNB#,TEBPB#,ZHBI#,ZXQX#,
TEBPN#,ZFR#,ZHBSX#,ZFBPX#,ZFBXS#,ZEGIB#,ZEGXB#,
ZXBB#;

COMMON /CNTPL5/ EXPEC(25),RKCEN,TIMEIR(25),IEIR,KREFUN,KDEGFI;

COMMON /CONST5/ ZZZ0,ZZZ1,ZZZ2,ZZZ3,ZZZ4,ZZZ5,ZZZ6,ZZZ7,ZZZ8,ZZZ9,
ZZZ314,ZZZ10,ZZZ01,ZZZ981,ZZZ50,ZZZWA,ZZZSS,ZZZER,ZZZNUM,
ZZZ273,ZZZST,ZZZINF,ZZZAST,ZZZPLS;

COMMON /CORTP5/ TECA,TECA0,TECI,TECX,TEFA,TEFA0,TELA,CF,
CL,CC,QMF,QML,VC,ULA,UCA,ROCA,ZFLCOR,ZROCK,ZPCORA,ZEGXC,
ZTECBI,ZVOLFC,ZHCX;

COMMON /DERIV5/ YYYY(%),DYDT(%),EYYYY1(%),EYYYY2(%),EYYYY3(%),
ZTIME1(%),ZDELT1(%),YYYY1(%),YYYY2(%),YYYY3(%),YYYY4(%),
YYYY5(%),DYDT1(%),DYDT2(%),DYDT3(%),DYDT4(%),DYDT5(%),
YYYYM1(%),YYYYM2(%),YYYYM3(%),DYDTM1(%),DYDTM2(%),DYDTM3(%),
IYYYY(%),NODER,LOOPS,ILSTEP(%),ILSTPS(%);

COMMON /DRUM#5/ ZHDRI#,ZPDF#,ZFDRI#,ZFSTX#,ZHSTX#,
ZFDW#,ZHFDW#,ZFDRX#,ZHDRX#,ZXD1#,ZTEDR#,ZTDRX#,
ZQDW#,ZQDS#,ZRODW#,ZRODS#,ZEGID#,ZEGXD#,ZPDRX#;

COMMON /FDBFK5/ RKFB,RKS,RKSC,RKD,RKDC,RKE,RKEC;

COMMON /FDWT#5/ ZFDCN#;

COMMON /GAMAF5/ WGT,WGT0,PGMA(5),DPGMA(5),BGM(5),
WGP,WGP0,WGN,GAM(5),GA,IGAM;

COMMON /HCA#5/ ZDXH#,ZXHC#;

COMMON /HFLOW5/ ZFLOW(%),ZDPFL(%),ZINERF(%);

COMMON /HPIPE5/ ZINERT(%),ZOHPIF(%),ZDPH(%),
ZFPK(3),ZFALF(3);

COMMON /HFUME5/ ZFLCH(%),ZHRPU(%),ZHRZF(%),ZIPUMP(%),
ZREVR(%),ZPU1(%),ZPU2(%),ZTORPR(%),ZROPU(%)
```

```

      ZFUMAT(%),ZTEPIN(%),ZTORP(%),ZPTRIP(%),ZREV(%);

COMMON /INXM#5/ TE#PI,TE#PX,TE#HI,TE#HX,AR#P,X#PH,
      X#RP,R#W,C#W,FP#I,PH#I,X#WL,Z#PDI,B#W,
      TE#P(%),TE#W(%),TE#H(%),E#PSHA,E#PSHR,ZPP#A,
      ZPH#A,ZRCPX#,ZROHX#,ZEGII#,ZEGIX#,ITO#MX,ITI#MX;

COMMON /LPLEN5/ TENA,TENFA,VNC,CLPF,CN,UNA,TENI,TENX,ROCN;

COMMON /MXPL#5/ ZTPM#,ZTPF#,ZTM#A,ZRPLE#,ZPPLE#;

COMMON /NEUTR5/ BETA(6),ELAMDA(6),EL,PNO,SUMC,
      EPSN1,EPSN2,PNJ,PNE,REAK,DREAK,EPSN,
      CIJ(6),DCL(6);

COMMON /SAFTY5/ RKSF,TSCRAM;

COMMON /TIMER5/ TIMEF,TCNTRL,DTMAX,DTMIN,TPRIN,TIMEM,
      TIME,TEPSN,DELT,TIMEI,HDTMIN,TIMEDS,TICPU,
      ERRMES(15),TIMEDE,DELTN,LSTR,JTIM,LETN,IDUB,
      IDIV,ITFRM,IData,IO5,IO6,IFORT,ISCR1,ISCR2,
      ISCR3,LIB1,LIB2,LIB3,LIB4;

COMMON /TPOWER5/ PW0,PWJ;

COMMON /UPLEN5/ TENA,TENFA,VNC,QUPE,CN,UNA,TENI,TENX,ROCN;

COMMON /WASPR5/ F1,F2,F11,F12,F22,F21,TES1,TES2,TNEW,VNEW,TOLD,
      VOLD,IS2;

```