14690-H002-RO-00

077132

CR 134272

SYSTEMS IMPROVED NUMERICAL DIFFERENCING ANALYZER

(NASA-CR-134272) SYSTEMS IMPROVED NUMERICAL DIFFERENCING ANALYZER (SINDA): ENGINEERING-PROGRAM MANUAL (TRW Systems Group) 354 p

N74-75224

Unclas

00/99 39316

Engineering-Program Manual

T. ISHIMOTO and L.C. FINK

JUNE 1971

NASA Contract 9-10435

TRW SYSTEMS
ONE SPACE PARK • REDONDO BEACH, CALIFORNIA

Systems Improved Numerical Differencing Analyzer
Engineering-Program Manual

T. Ishimoto and L. C. Fink

June 1971

NASA Contract 9-10435

Prepared for:
National Aeronautics and Space Administration
Manned Spacecraft Center
Under Contract NASA 9-10435

Prepared by:

T. Ishimoto

L. C. Fink

Approved by:

R. G. Payne, Program Manager

Approved by:

J. T. Bevans, Manager
Heat Transfer & Thermodynamics Department

Approved by:

R. L. Dotts
NASA Technical Monitor
NASA Manned Spacecraft Center

ACKNOWLEDGEMENT

This engineering-program manual includes the efforts of several individuals who in no small way contributed to the publication of this document. Many thanks are due to:

J. D. Gaski, who as the prime mover behind the SINDAS (SINDA, CINDA-3G and CINDA, provided valuable counseling on the inner workings of the program, expecially the execution routines. Many of the computational features in these routines are original with him; it is through his cooperative assistance that the writing of this document became a reasonable venture.

R. L. Dotts, who must be given special mention not because of his position as NASA/MSC technical monitor but because of his sincere interest in the use of SINDA and his willingness to provide assistance whenever and wherever possible.

Mrs. Dorothy Gramlich, who typed this manuscript with a cheerful, professional attitude and technique that made this phase of the program a more pleasant one.

TABLE OF CONTENTS

		Page

FIGURES

TABLES

PREFACE

The present SINDA computer program has evolved from the CINDA-3G program, which in turn evolved from the CINDA program, etc. With each major program revision an updated user manual was generated, but a more in-depth presentation of programming considerations and the theoretical development of the numerous subroutines were not generated. This SINDA program manual represents a preliminary effort to fill some of the existing void by describing the program structure, by identifying the major functions of each processor routine with a functional flow chart, and by a more in-depth mathematical description of the numerical solution subroutines. It is not the intent of this engineering-program manual, however, to provide sufficient detailed information for a user to make modifications and/or additions to the existing subprograms.

1. NOMENCLATURE AND MNEMONICS

1.1 Nomenclature

a_{ij} = $k(A/\ell)_{ij}$, conduction coefficient between nodes i and j.

A = array

A = area

$(A/\ell)_{ij}$ = effective ratio of cross-sectional area to distance between nodes i and j.

b_{ij} = radiation factor between nodes i and j (composed of radiation interchange factor and area)

C_i = capacity of ith node

$\overline{C}_i$ = $C_i/\Delta t$, capacity of ith node divided by time-step

DD = 1 - DN (allows certain fraction of "old" temperature to be included as part of temperature change for current time-step, refer to Section 6.2.5.1)

DN ≡ DAMPA (user control constant, refer to Sections 6.2.5.1 and 6.2.3.2)

F,F1,F2 = multiplying factors, either user constants or literals, refer to Tables 6.2-1, 6.2-2 and 6.2-3).

G_{ij} = $a_{ij} + \sigma b_{ij} (T_i^2 + T_j^2)(T_i + T_j)$

G_k = a_{ij} or σb_{ij}, conduction or radiation coefficient.

k = thermal conductivity

L = a literal multiplying factor

N = number of variable temperature nodes (NNA + NND)

NNA = number of arithmetic-nodes

NND = number of diffusion nodes

R = resistance

p = total number of nodes

q_i = impressed heat load into the ith node

t = time

Δt = time-step

t_m = (TIMEØ + TIMEN)/2.0, mean time

T = temperature (°F or °R)

T_m = $(T_i + T_j)/2.0$, mean temperature (°F or °R)

x, y	= coordinate
α	= (k/C), thermal diffusivity
α	= $\sum_{j=1}^{p} G_{ij}/C_i$, refer to equation 6.3-7
β	= factor that ranges from 0 to 1/2 (refer to equation 6.2-7)
β'	= 2β (used in subroutine CNVARB)
$\mathcal{F}_{ij}$	= radiation interchange factor including inter-reflections between nodes i and j.
σ	= Stefan-Boltzmann constant ($.1714 \times 10^{-8}$ Btu/hr $°R^4 ft^2$)
τ_n	= $\dfrac{\Delta t_n}{\Delta t_{n-1} + \Delta t_n}$, weighting factor (equation 6.3-13)

$(A^i:t_m)$	Interpolated	value	of	array	A	using	t_m	as	the	independent	variable
$(A^i:T_i)$	"	"	"	"	"	"	T_i	"	"	"	"
$(A^i:T_j)$	"	"	"	"	"	"	T_j	"	"	"	"
$(A^i:T_m)$	"	"	"	"	"	"	T_m	"	"	"	"
$(A^P:T_i)$	Polynomial	"	"	"	"	"	T_i	"	"	"	"
$(A^P:T_i)$	"	"	"	"	"	"	T_j	"	"	"	"
$(A^P:T_m)$	"	"	"	"	"	"	T_m	"	"	"	"

$(A^b:T_i,t_m)$ Interpolated value of the bivariate array A using T_i and t_m as independent variables.

$(A^b:T_m,t_m)$ Interpolated value of the bivariate array A using T_m and t_m as independent variables.

Subscripts

i	= ith node
j	= jth node
ij	= between nodes i and j
i,n	= updating of ith temperature, source, etc. at time-step n.
i,k	= updating of ith temperature, etc. at kth iteration.
ij,n	= updating of coefficient between nodes i and j at time-step n
ij,k	= updating of coefficient between nodes i and j at kth iteration
m	= mean

1.2 Mnemonics

1.2.1 Control constants (refer to Sections 6.2.3.1 and 6.2.3.2)

ARLXCA = allowable arithmetic node relaxation temperature change

ARLXCC = calculated maximum arithmetic node relaxation temperature change

ATMPCA = allowable arithmetic node temperature change

ATMPCC = calculated maximum arithmetic node temperature change

BACKUP = back switch

BALENG = specified system energy balance

CSGFAC = time-step factor

CSGMAX = maximum value of $C_i/\Sigma\ G_{ij}$

CSGMIN = minimum value of $C_i/\Sigma\ G_{ij}$

CSGRAL = allowable range between CSGMIN and CSGMAX

DAMPA = arithmetic node damping factor

DAMPD = diffusion node damping factor

DRLXCA = allowable diffusion node relaxation temperature change

DRLXCC = calculated diffusion node relaxation temperature change

DTIMEH = allowable maximum time-step

DTIMEI = specified time-step for implicit solutions

DTIMEL = allowable minimum time-step

DTIMEU = contains computed time-step

DTMPCA = allowable diffusion node temperature change

DTMPCC = calculated maximum diffusion node temperature change

ENGBAL = calculated system energy balance

ITEST = contains dummy integer constant

JTEST = contains dummy integer constant

KTEST = contains dummy integer constant

LAXFAC = number of iterations for linearized lumped parameter system, CINDSM only.

LINECT = line counter location for program output

LØØPCT = contains number of iterations performed

NØCØPY = contains no copy switch for matrix users

NLØØP = number of specified iteration loops

ØPEITR = output each iteration switch

PAGECT = page counter location for program output

3<

RTEST = contains dummy floating point constants

STEST = contains dummy floating point constants

TIMEM = (TIMEØ + TIMEN)/2.0, mean time for computational interval

TIMEN = TIMEN + DTIMEU, new time at the end of computational interval

TIMEND = problem stop time

TIMEØ = old time at the start of the computational interval

TTEST = contains dummy floating point constants

UTEST = contains dummy floating point constants

VTEST = contains dummy floating point constants

1.2.2 Numerical Solution Routines (refer to Sections 6.3 - 6.5)

CINDSS = steady state routine, refer to Section 6.5.1

CINDSL = steady state routine, refer to Section 6.5.2

CINDSM = steady state routine, refer to Section 6.5.3

CNBACK = implicit routine, refer to Section 6.4.1

CNDUFR = explicit routine, refer to Section 6.3.4

CNEXPN = explicit routine, refer to Section 6.3.3

CNFAST = explicit routine, refer to Section 6.3.2

CNFRDL = explicit routine, refer to Section 6.3.1

CNFWBK = implicit routine, refer to Section 6.4.2

CNFWRD = explicit routine, refer to Section 6.3.1

CNQUIK = explicit routine, refer to Section 6.3.5

CNVARB = implicit routine, refer to Section 6.4.3

1.2.3 Options (used in Tables 6.2-1 - 6.2-3)

BIV = Bivariate Interpolation Variable

DIT = Double Interpolation with Time as variable

DIV = Double Interpolation Variable

DPV = Double Polynomial Variable

DTV = Double interpolation with Time and Temperature as Variables

SIT = Single Interpolation with Time as variable

SIV = Single Interpolation Variable

SPV = Single Polynomial Variable

1.2.4 Routines and Subroutines of Preprocessor

SINDA = routine that specifies overlay of preprocessor to system allocator.

PREPRØ = main routine for preprocessor; initializes counters and FØRTRAN logical units; sets length of dynamic storage array and controls major logic.

ALPINT = subroutine that accepts an integer in alphanumeric format and converts it to integer format; determines relative number of this actual number and converts it back to alphanumeric format.

BLKCRD = subroutine that formats the five generated FØRTRAN routines (SINDA, EXECTN, VARBL1, VARBL2, and ØUTCAL) in 507 word blocks.

CØDERD = Subroutine that reads and checks the block header cards for the data blocks.

CØNVRT = subroutine that converts Hollerith data to integer data.

DATARD = subroutine that scans the data block card images under an A format and determines appropriate format to reread the card images.

ERRMES = subroutine that prints most of the error messages generated within the data blocks.

FINDRM = subroutine that moves the data in the dynamic storage array either up or down by 100 words.

GENLNK = subroutine that generates the driver (FØRTRAN routine named SINDA) for the user's program.

GENUK = subroutine that generates user constants.

INCØRE = subroutine that reads data into the dynamic storage array for the parameter-runs option.

MXTØFN = subroutine that processes data for the "m" option (converts card images from mixed FØRTRAN/SINDA notation to FØRTRAN notation.

NØDEDA = subroutine that processes data for node and conductor data blocks.

PCS2 = subroutine that packs the FØRTRAN addresses for the array and constants locations required by the second pseudo-compute sequence.

PRESUB = subroutine that reads and checks the block header cards for the operations blocks and generates the non-executable FØRTRAN cards for each of the operations blocks via a call to BLKCRD.

PSEUDØ = subroutine that forms the first and second pseudo-compute sequences.

QDATA = subroutine that checks and processes all data input in the source data block.

RELACT = subroutine that finds the relative node numbers from the actual node number; computes the FORTRAN address for arrays and user constants from the actual number.

SEARCH = subroutine that retains a relative number for nodes, conductors, user constants, and arrays, given the actual number.

SETFMT = subroutine that processes the card for the "new format" option; that is, it sets up the format for data cards as specified by the cards with an "N" in column one.

SINDA4 = subroutine that reads and processes the user input cards from the operations blocks.

SKIP = subroutine that is used when a problem is RECALLED; it positions the tape to the proper problem as specified on the first card of the data deck.

SPLIT = subroutine that reads the data from the RECALL tape and splits the RECALL information onto the proper data "tape" and the dictionary "tape."

SQUEEZ = subroutine that compresses the specified data groups in the dynamic storage array.

STFFB = subroutine that fills out a card image in array KBLK with Hollerith blanks.

TYPCHK = subroutine that checks the input from the data blocks for the correct type (integer, floating point, or alphanumeric.

WRTDTA = subroutine that writes the program data "tape" in the format required by INPUTT or INPUTG.

WRTPMT = subroutine that writes the required data for parameter runs on the parameter "runs" "tape" and the dictionary "tape."

WRTBLK = subroutine that writes the 507 word blocks contained in array KBLK on the program FØRTRAN "tape."

1.2.5 Others

SINDA = Systems Improved Numerical Differencing Analyzer

LPCS = Long Pseudo-Compute Sequence

SPCS = Short Pseudo-Compute Sequence

LPCS2 = Second Long Pseudo-Compute Sequence

PCS1 = Pseudo-Compute Sequence One

PCS2 = Pseudo-Compute Sequence Two

TSUM = elapsed time from last printout

TPRINT = time of last printout.

2. BACKGROUND ON SINDA

The original CINDA*[1] (Chrysler Improved Numerical Differencing Analyzer) computer program was developed by the Thermodynamics Section of the Aerospace Physics Branch of Chrysler Corporation Space Division at NASA Michoud Assembly Facility and was coded in FORTRAN-II and FAP for the IBM-7094 computers. CINDA was the product of an intensive analytical, engineering and programming effort that surveyed numerous thermal analyzer-type programs and studied several in-depth. The foundation for CINDA was the storage and addressing of information required only for the network solution and the systems features which allowed the re-utilization of core storage area and brought into core only those instructions necessary for the solution of a particular problem. A systems compiler computer program that automatically optimized the utilization of computer core space was developed. This meant the generation of an integrated operation of relative addressing, packing features, peripheral tape storage units and overlay features.

CINDA evolved into CINDA-3G[2] which was developed by the same group that generated CINDA with a major portion of the work done under contract NASA/MSC NAS9-7043. CINDA-3G represented (essentially) a complete rework of CINDA in order to take advantage of the improved systems software and machine speeds of the 3rd generation computers. CINDA was unsuitable for standard operation on third generation computers since it was virtually a self-contained program having its own Update, Monitor and Compiler. On the other hand, CINDA-3G consisted of a preprocessor (written in FORTRAN) which accepted the user input data and the block data input. The user input data was converted into advanced FORTRAN language subroutines and block data input was passed onto the system FORTRAN Compiler. This required a double pass on data where previously only one was required, but the increased speed and improved software of the third generation machines more than compensated for the double pass.

SINDA[3] (Systems Improved Numerical Differencing Analyzer) was developed by the Heat Transfer and Thermodynamics Department of TRW Systems Group, Redondo Beach. Most of the improvements and subroutine additions to CINDA-3G was done as part of the NASA/MSC contract NAS 9-8389,

* Superscript numbers refer to the literature cited in the Reference Section.

entitled, "Development of Digital Computer program for Thermal Network Correction." Programming and systems integration were directed to the UNIVAC-1108 computer.

SINDA relied quite heavily on CINDA-3G and data deck compatibility was rigorously followed; as a result, CINDA-3G data decks should, in the main, be directly operational on the SINDA program although some differences exist. For example, properties are updated before VARIABLES 1 call in CINDA-3G, whereas the properties are updated within the numerical solution routines after VARIABLES 1 call in SINDA. The primary differences between SINDA and CINDA-3G are: (1) elimination wherever possible of assembly language coding; (2) increased mnemonic options to aid the program user in data input; (3) inclusion of a second pseudo computer sequence for evaluation of nonlinear network elements; and (4) additional subroutines such as STEP (sensitivity analysis) and KALØBS-KALFIL (Kalman filtering). Most of the changes and additions to CINDA-3G were required in order to integrate the thermal network correction subroutine package into the existing CINDA-3G program.

During the development of SINDA a number of useful improvements became apparent. As a result, modifications to SINDA were made a part of the NASA/MSC contract NASA 9-10435 entitled, "Development of an Advanced SINDA Thermal Analyzer System." These changes that include a variable input format, simplified parameter runs, and generated user constants were made by the same group that developed SINDA. These improvements are reported in an updated SINDA users manual.[4]

3. GENERAL SINDA PROGRAM DESCRIPTION

3.1 SINDA Operating System

SINDA is more like an operating system rather than applications program. SINDA is programmed as a preprocessor in order to accommodate the desired operations relative to overlay features, data packing, dynamic storage allocation and subroutine library file, but yet be written in FØRTRAN. This preprocessor operates in an integral fashion with a library of numerous and varied subroutines,[3,4] which may be called in any desired sequence but yet operate in an integrated manner. The preprocessor reads the input data, assigns relative numbers, packs this information, forms a pseudo-compute sequence(s) (which will be described briefly in a later paragraph of this section and is described in more detail in Section 4, called Preprocessor), and writes the operations blocks on a peripheral unit as FØRTRAN source language with all of the data values dimensioned exactly in labeled common. In turn, controls are shifted to the system FØRTRAN compiler which compiles the constructed subroutines and enters execution. The FØRTRAN allocator has access to the SINDA subroutine library and loads only those subroutines called by the problem being processed.

As a result of this type of systems operation SINDA is extremely dependent upon the systems software. However, once the program is operational on a particular computer, the user-prepared problem data deck can be confined to the control cards and deck set-up requirements at a particular installation.

It should be recognized that the use of a preprocessor provides a computer with a large capability and considerable flexibility, but because of the numerous options that are generally offered, user instructions are more difficult than other thermal analyzer-type programs which have less flexibility.

3.2 Use of Lumped-Parameter Concept

Use of SINDA is based on a lumped parameter representation of a physical system. This means that SINDA does not solve a set of partial differential equations that represents a distributive system, but rather SINDA numerically solves a set of ordinary (and in general) nonlinear differential equations that represent a lumped parameter system. The procedure for the formulation and the numerical solutions of the lumped parameter equations are reported extensively in literature and basic considerations are presented in Section 5. For the discussion to follow on the pseudo compute sequence it is convenient to indicate a general set of ordinary linear differential heat balance equations,

$$\frac{dT_i}{dt} = \frac{1}{C_i}\left[q_i + \sum_{j=1}^{p} a_{ij}\ (T_j - T_i)\right] \qquad (3.2\text{-}1)$$

$i = 1,2,\ldots,N$ (number of variable temperatures)

$T_j =$ constant, $N < j \leq p$

where, C_i = the ith nodal capacity

q_i = the heat load into node i (impressed)

a_{ij} = the conduction coefficient between nodes i and j $[= k\left(\frac{A}{\ell}\right)_{ij}]$

t = time

Suppose an implicit numerical method as discussed in Section 5.2.2 of this manual is chosen; the implicit finite difference form becomes after letting,

$dT_i/dt \cong (T_{i,n+1} - T_{i,n})/\Delta t$, $T_j = T_{j,n+1}$ and $T_i = T_{i,n+1}$,

$$\frac{C_i}{\Delta t}\ (T_{i,n+1} - T_{i,n}) = q_i + \sum_{j=1}^{p} a_{ij}\ (T_{j,n+1} - T_{i,n+1}) \qquad (3.2\text{-}2)$$

where, $T_{i,n}$ = temperature at time point t_n

$T_{i,n+1}$ = temperature at time point $t_{n} + \Delta t$

Δt = time-step

Rearrangement of equation (3.2-2) yields,

$$(\overline{C}_i + \sum_{\substack{j=1\\ j\neq i}}^{p} a_{ij})\ T_{i,n+1} - \sum_{\substack{j=1\\ j\neq i}}^{N} a_{ij}\ T_{j,n+1} = q_i + \overline{C}_i\ T_{i,n} + \sum_{j=N+1}^{p} a_{ij}\ T_{j,n} \qquad (3.2\text{-}3)$$

$i = 1,2,\ldots,N$

$T_{j,n} =$ constant, $N < j \leq p$

where, $\overline{C}_i = C_i/\Delta t$, average capacity of node i over Δt time-step

3.3 Pseudo-Compute Sequence (PCS)

A pseudo-compute sequence as generated by the SINDA preprocessor is a list of numbers that indicates the position of required data values in various arrays such as conductance, temperature and capacitance. This meaning will become clearer by formulating equation (3.2-3) in a matrix form. The matrix formulation is straightforward since temperatures at time-step n+1 are the unknowns and terms on the right side of equation (3.2-3) represent the forcing function. Let us expand equation (3.2-3) to show this,

$$(\overline{C}_1 + \sum_{j=1}^{p} a_{1j}) T_{1,n+1} - a_{12}\, T_{2,n+1}, \ldots, -a_{1N}\, T_{N,n+1} = q_1 + \overline{C}_1\, T_{1,n} + \sum_{j=N+1}^{p} a_{1j}\, T_{j,n}$$

$$-a_{21}\, T_{1,n+1} + (\overline{C}_2 + \sum_{j=1}^{p} a_{2j}) T_{2,n+1}, \ldots, -a_{2N}\, T_{N,n+1} = q_2 + \overline{C}_2\, T_{2,n} + \sum_{j=N+1}^{p} a_{2j}\, T_{j,n}$$

$$\vdots \qquad\qquad\qquad\qquad \vdots$$

$$-a_{N1}\, T_{1,n+1} - a_{N2}\, T_{2,n+1}, \ldots, (\overline{C}_{N1} + \sum_{j=1}^{p} a_{Nj}) T_{N,n+1} = q_N + \overline{C}_N\, T_{N,n} + \sum_{j=N+1}^{p} a_{Nj}\, T_{j,n}$$

Thus the matrix form of equation (3.2-3) becomes,

$$[\beta]\ \{T'\} = \{Q\} \qquad (3.3\text{-}1)$$

where,

$$\beta = \begin{bmatrix} \sum\limits_{\substack{j=2 \\ j\neq 1}}^{p} (\overline{C}_i + a_{1j}), & -a_{12} & , \ldots, & -a_{1N} \\ -a_{21} & , \sum\limits_{\substack{j=1 \\ j\neq 2}}^{p} (\overline{C}_2 + a_{2j}) & , \ldots, & -a_{2N} \\ \vdots & \vdots & & \vdots \\ -a_{N1} & , \quad -a_{N2} & , \ldots, & \sum\limits_{\substack{j=1 \\ j\neq 2}}^{p} (\overline{C}_N + a_{Nj}) \end{bmatrix} \qquad (3.3\text{-}2)$$

$$T' = \begin{Bmatrix} T_{1,n+1} \\ T_{2,n+1} \\ \vdots \\ T_{N,n+1} \end{Bmatrix} \ (3.3\text{-}3) \quad ; \quad \{Q\} = \begin{Bmatrix} q_1 + \overline{C}_1\, T_{1,n} + \sum\limits_{j=N+1}^{p} a_{1j}\, T_{j,n} \\ q_2 + \overline{C}_2\, T_{2,n} + \sum\limits_{j=N+1}^{p} a_{2j}\, T_{j,n} \\ \vdots \\ q_N + \overline{C}_N\, T_{N,n} + \sum\limits_{j=N+1}^{p} a_{Nj}\, T_{j,n} \end{Bmatrix} \ (3.3\text{-}4)$$

The matrix represented by equation (3.3-2) <u>appears</u> to be a full matrix (very small number of elements that are zero), but in reality most of the off-diagonal elements are zero. Thus, if equation (3.2-3) was to be solved by a matrix inversion technique, all elements including zeros must be stored. Since the number of elements varies as N^2 (N is the number of nodes), the required number of data locations would vary as N^2 and the computer time required for matrix inversion would be proportional to N^3.

The explicit and iterative implicit numerical methods (refer to Section 5) of solving equation (3.2-1) lend themselves for optimizing the data storage area required and for reducing the solution time. If the conductors are numbered and related to the appropriate adjoining nodes as indicated in Table (3.2-1), retention of adjoining node number for each conductor provides a means of identifying element position in the coefficient matrix. This can be seen by considering the one-dimensional heat conduction example pictured in Figure (3.3-1).

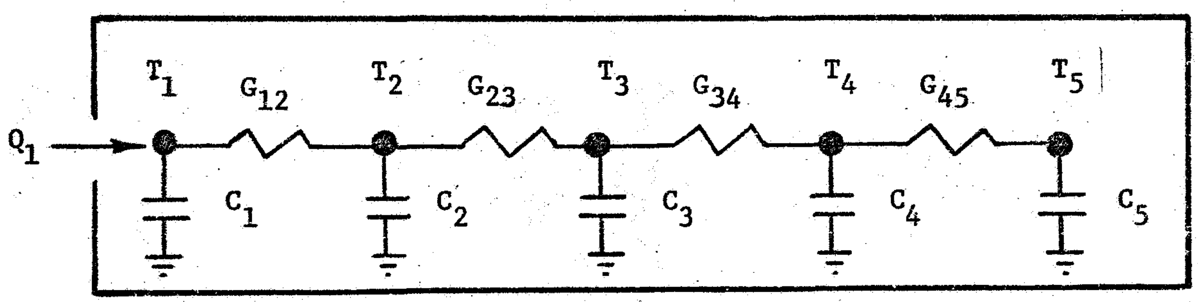


Figure 3.3-1. Thermal Circuit for a One-Dimensional System

The set of equations associated with the problem of Figure (3.3-1) may be readily expressed as,

$$\begin{bmatrix} (\overline{C}_1+G_1), & -G_1, & 0, & 0, & 0 \\ -G_1, & (\overline{C}_2+G_1+G_2), & -G_2, & 0, & 0 \\ 0, & -G_2, & (\overline{C}_3+G_2+G_3), & -G_3, & 0 \\ 0, & 0, & -G_3, & (\overline{C}_4+G_3+G_4), & -G_4 \\ 0, & 0, & 0, & -G_4, & (\overline{C}_5+G_4) \end{bmatrix} \begin{Bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \end{Bmatrix} = \begin{Bmatrix} Q_1 \\ 0 \\ 0 \\ 0 \\ 0 \end{Bmatrix} \qquad (3.3\text{-}5)$$

By comparing the element position of equation (3.3-5) with the tabular identification in Table (3.2-1) it is seen that elements with zero values need not be stored. The main diagonal term is never zero and is a composite of capacitance and off-diagonal conductors.

Table (3.2-1) Tabular Identification of Conductor and Adjoining Node Numbers

Conductance Number G#	ith Node N#	Adjoining Node Number N#	Comment
1	1	2	G1 is conductor #1 between nodes 1 and 2.
1	2	1	G1 is conductor #1 between nodes 2 and 1.
2	2	3	G2 is conductor #2 between nodes 2 and 3.
.	.	.	
.	.	.	
4	5	4	G4 is conductor #4 between nodes 5 and 4.

It is of interest to note that the use of a pseudo-compute sequence is only one of a number of ways to store data efficiently. For example, TRW TAP[5] does not employ a pseudo-compute sequence because of other user requirements. However, from a data storage standpoint, it appears that the use of a pseudo-compute sequence utilizes computer core most efficiently.

More than one pseudo-compute sequence is formed by SINDA. Both a so-called long (LPCS) and a so-called short (SPCS) pseudo-compute sequence as used in CINDA-3G[2] are formed and in addition a second long pseudo-compute (LPCS2) required for thermal network correction is also formed in SINDA. A detailed discussion of these pseudo-compute sequences will be presented in Section 4.6, but is of interest here to indicate the characteristics of these "sequences."

3.3.1 Long Pseudo-Compute Sequence (LPCS)

A long pseudo-compute sequence identifies the position and value of all off-diagonal elements of the coefficient matrix. This is done by operating on adjoining node numbers which have been assigned relative node numbers by the preprocessor. Since nodal temperatures are calculated sequentially in ascending numerical order, the conductor and adjoining node number are searched until node one is found with the conductor number and the other adjoining node number stored in a single core location. In addition, several indicators are stored in this single core location. These indicators are : (1) var C (indicates the input of a capacitor as a variable); (2) var G (indicates the input of a conductor as a variable); (3) rad (indicates the input of a radiation conductance); (4) Q (indicates the input of a source in the source data block); (5) one-way (indicates the input of a one-way conductor); and (6) last G (indicates the last conductor to a particular node). Order of indicator storage is indicated in Table (3.3-2).

Search is continued until all node-one's have been located and characteristics processed. The procedure is repeated for all node-two's and so forth sequentially until all nodes have been processed. The important consideration of a LPCS is the encounter of each conductor of the coefficient matrix twice. Formation of a pseudo-compute sequence for the example shown in Table (3.3-1) is given in Table (3.3-2). A pseudo-compute sequence starts with node one and advances the node number by one each time a last conductor indicator (last G) is passed. The conductor and node numbers identify the position of the conductor value in an array of conductor values and the position of the temperature, capacitor and source values in arrays of temperature, capacitor and source values respectively.

A long pseudo-compute sequence is well-suited for "successive point" iteration (refer to Section 5.2.2 for a discussion of this) of the implicit finite difference equations because all elements of the coefficient matrix are identified. Thus, when a row of the coefficient matrix is processed and a new value of temperature obtained, the new temperature can then be used in the calculation procedure of succeeding rows.

3.3.2 Short Pseudo-Compute Sequence (SPCS)

The short pseudo-compute sequence identifies each conductor only once and since the coefficient matrix (equation 3.3-1) is symmetrical, all sparsity and off-diagonal elements of the coefficient matrix are accounted for. The node being processed and the adjoining node number reveal temperature- and source-value locations. The short pseudo-compute sequence for the example in Table (3.3-1) is formed in Table (3.3-3). By placing a minus sign on the initially encountered other-adjoining nodes, these nodes are not recognized on a second encounter. A short pseudo-compute sequence

Table (3.3-1) Example of Conductor Connections

Conductor No. G#	Adjoining Node Numbers N#	N#
1	1	2
2	1	3
3	1	4
4	2	3
5	2	4
6	3	4

Table (3.3-2) Long Pseudo-Compute Sequence (LPCS)
for the Example of Table (3.3-1)

Node No. Searched	Last G	var C	var G	rad	Q	G#	One-way	Node # Stored
1						1		2
1						2		3
1	1					3		4
2						1		1
2						4		3
2	1					5		4
3						2		1
3						4		2
3	1					6		4
4						5		2
4	1					6		3

Table (3.3-3) Short Pseudo-Compute Sequence (SPCS)
for the Example of Table (3.3-1)

Node No. Searched	Last G	var G	var G	rad	Q	G#	One-way	Node # Stored
1						1		2
1						2		3
1	1					3		4
2						4		3
2	1					5		4
3	1					6		4
4	1					0		0

is well-suited for explicit numerical solutions methods which calculate the energy flow through the conductor, add it to the source location of the node being processed and subtract it from the source location for the adjoining node. The SPCS can be used for implicit methods of solution but the "block" iterative procedure (refer to Section 5.2.2 for a discussion of this) must be used since succeeding rows of conductor and adjoining node numbers do not contain the necessary element information.

3.3.3 Second Long Pseudo-Compute Sequence (LPCS2)

The second long pseudo-compute sequence (LPCS2) as a user input option flags a non-linear conductor between two diffusion nodes twice; LPCS flags the non-linear conductor only one. LPCS2 is required for the thermal network correction of a sparse network by the use of subroutine KAFIL (refer to Reference 3 or 6).

3.3.4 Pseudo-Compute Sequence One (PCS 1) and Pseudo-Compute Sequence Two (PCS 2)

PCS 1 and PCS 2 are not user options but are fixed internally. The contents of PCS 1 and PCS 2 are governed by the user input of LPCS, SPCS or LPCS2). PCS 1 contains two relative addresses (conductor and adjoining node locations), two non-linear type indicators, and an impressed source indicator. Indicators are keyed through a simple counter to a second pseudo-compute sequence (PCS 2) which contains integer addresses or relative constant and array starting locations necessary for evaluation of temperature varying coefficients and time varying coefficients for sources. When the input data contain literal values in SIV type calls, the preprocessor stores the values as extended user constants and supplies the relative constant address to the second pseudo-compute sequence. Detailed discussion on PCS 1 and PCS 2 is presented in Section 4.6.

3.4 Data Logistics

3.4.1 Relative Numbers

Both the long and short pseudo-compute sequences require the storage of only the finite values in the coefficient matrix, thereby taking advantage of matrix sparsity. If the short pseudo-compute sequence is used, the advantage of symmetry is accounted for. Conductors with the same constant value may share the same conductor number and value. The storage efficiency of the pseudo-compute sequences requires the sequential numbering of the nodes and the conductors. Since the numbering of thermal math-models is arbitrary and not sequential, the SINDA program assigns relative numbers (starting from one, sequential and ascending) to the actual numbers of the incoming node data, conductor data, constants data and array data in the order received. Thus, numbers not used in the actual numbering system are neither identified nor required.

3.4.2 Storage Requirements and Dynamic Storage Allocation

All numerical solution subroutines require three locations for each diffusion node data (temperature, capacitance and source), two locations for each arithmetic node data (temperature and source), one location for each boundary data (temperature) and one location for each conductor value. In addition intermediate data storage ranging from zero to three locations per node may be required for the storage of temperatures and temperature differences; acceleration of convergence (refer to Section 6.2.7) used in the implicit and steady state routines (except CINDSS) requires three locations. Storage requirements for conductances depends upon the problem. For example, each internal diffusion and arithmetic node of a three-dimensional conduction system with rectangular-nodalization will be connected with only three being unique; thus, each diffusion node (or arithmetic node) in a three-dimensional conduction system requires from six to nine storage locations for data values (temperature, capacitance, source, three conductors and up to three intermediate locations). Now each of the conductors for the short pseudo-compute sequence requires a single core location that contains two integer values (conductor and adjoining node numbers) and six indicators (refer to Section 3.3.1 for description). Each of the conductors between variable temperatures for the long pseudo-compute sequence requires two core locations since the conductors are used twice during the computational process. This means that each internal node of a three-dimensional conduction system will require six data addressing locations for the long pseudo-compute sequence and, on the average, three data addressing locations for the short pseudo-compute sequence.

Thus for a three-dimensional conduction system (no radiation), the number of required core locations per node can vary from nine (temperature, capacitance, source, three unique conductors and three data addressing locations) to fifteen (temperature, capacitance, source, six conductors and six data addressing locations) exclusive of the second pseudo-compute sequence which is required for variable coefficients, capacitance and sources.

The user must allocate an array of data locations which is to be used for intermediate data storage and initialize the array start and length indicators. Each subroutine that requires intermediate storage area has access to this array and the start and length indicators. During a subroutine execution a check on the sufficiency of space is made and start and length indicator are updated. If a subroutine calls upon another subroutine that requires intermediate storage, the called subroutine repeats the check and update procedure. Whenever any subroutine terminates its operation, the start and length indicators are returned to their entry values. This process is termed "Dynamic Storage Allocation" and allows subroutines to share a common working area.

3.5 Order of Computation

A network data deck consists of four data blocks (node, conductor, constants, and array), one optional data block (source) and four operations blocks which are preprocessed by the preprocessor and passed on to the system FØRTRAN compiler. Non-network problems require no node or conductor data blocks. The operations blocks are named EXECUTIØN, VARIABLES 1, VARIABLES 2 and OUTPUT CALLS: the SINDA preprocessor constructs these blocks into individual subroutines with the entry names EXECTN, VARBL1, VARBL2 and ØUTCAL, respectively. After a successful FØRTRAN compilation, control is passed to the EXECTN subroutine. This means that the order of computation depends on the sequence of subroutine calls placed in the EXECUTIØN block by the program user. No other operations blocks are performed unless called upon by the user either directly by name or indirectly through a subroutine call. The numerical solution subroutines described in Section 6 internally call upon VARBL1, VARBL2 and ØUTCAL; The internal order of computation for these routines is similar with the primary difference being the numerical solution method. A general flow diagram of the numerical solution routines, as well as a detailed description of each is presented in Section 6.

4. PREPROCESSOR

4.1 General Description

The SINDA preprocessor reads and analyzes the user input deck and from this information constructs a program tailored to the user's requirements.

The rationale for a preprocessor is flexibility and speed. Flexibility is achieved by providing the user with a library of routines to solve problems, manipulate data, and print selected values. In addition, the user may insert non-SINDA routines into the constructed program. Speed (defined here as minimal execution time) is achieved by structuring the data in an efficient manner.

The SINDA preprocessor consists of thirty routines with seven overlay links. All of the routines are written in FØRTRAN except for one assembly language routine which writes a "tape" in a format acceptable to the FØRTRAN compiler. These routines provide the user with a number of major options in the type of problem to be solved and the form of the data to be used. Henceforth these major options are designated as "major logic" of the preprocessor. See Figure 4.1-1 for a flow chart of the major logic of the preprocessor and its interface with the user program.

The major logic consists of the five following options: (1) NASA MSC EDIT feature; (2) RECALL option; (3) generation of a THERMAL problem; (4) generation of a GENERAL problem; (5) and PARAMETER RUNS option. The primary features of each item of the major logic is discussed below.

(1) EDIT feature: The first card of the deck is checked for the user request of the EDIT feature. If the EDIT feature is requested the input "tape" is changed from the system input "tape" to the EDIT "tape" and control is transferred to subroutine EDIT for processing. On return a branch is made to the THERMAL or GENERAL section as specified by the data on the EDIT "tape." If the EDIT feature has not been requested, the check for RECALL is made.

(2) RECALL option: The first card of the deck is checked for user request of the RECALL option. If the RECALL option

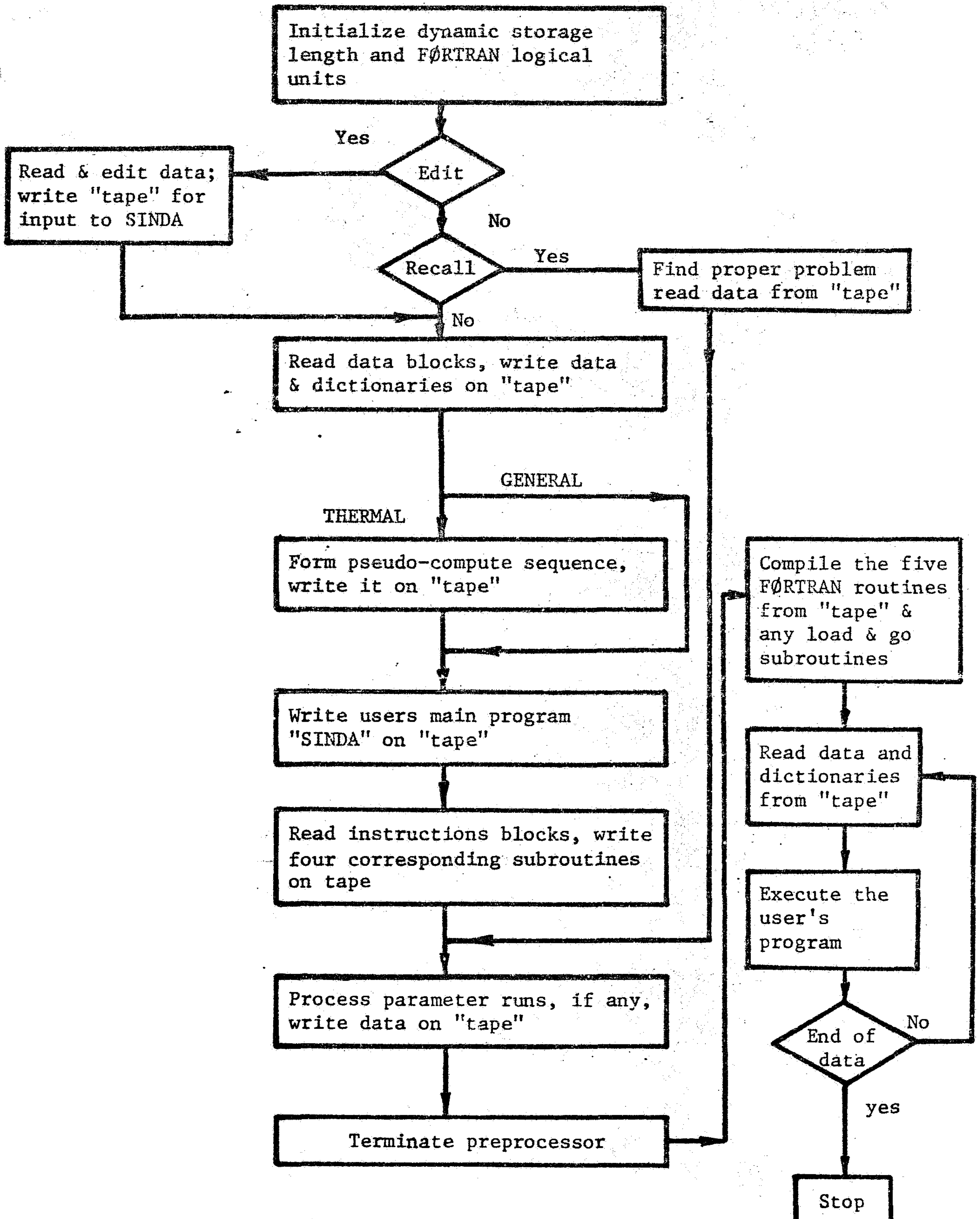


Figure 4.1-1. SINDA Preprocessor - Major Logic and Interface with the User Specified Problem

is requested, control is transferred to subroutine SPLIT for processing. On return a branch is made to the PARAMETER RUNS section. If the RECALL option has not been requested, the second card of the deck is checked for the type of problem, THERMAL or GENERAL.

(3) THERMAL problem: The type of pseudo-compute sequence requested is noted, the title block is read, the data blocks are read and processed, the pseudo-compute sequence is formed, the driver for the user program (SINDA) is written on "tape," the operations blocks are read and processed and their FØRTRAN equivalents are written on "tape," and finally a check is made for the user requests of the PARAMETER RUNS option.

(4) GENERAL problem: This section is identical to a THERMAL problem except that only constants data and array data of the data blocks are read and processed; a pseudo-compute sequence is not formed.

(5) PARAMETER RUNS option: A check is made for the user request of the PARAMETER RUNS option. If the PARAMETER RUNS option is requested, the appropriate data blocks are read and processed. If not, the preprocessor is terminated.

Description of SINDA preprocessor routines is presented in the sections to follow. Terminology used in the description is listed and defined in Table 4.1-1.

Table (4.1-1) Terminology Used in Description of SINDA Preprocessor Routines

(1) DATA BLOCKS: The five user input blocks which contain data rather than instructions; these DATA BLOCKS are NODE DATA, CONDUCTOR DATA, CONSTANTS DATA, ARRAY DATA and the optional blocks SOURCE DATA.

(2) OPERATIONS BLOCKS: The four user input blocks which contain instructions on problem solution, as opposed to data contained in the DATA BLOCKS. These OPERATIONS BLOCKS are EXECUTIØN, VARIABLES 1, VARIABLES 2 and ØUTPUT CALLS.

(3) Non-fatal error: An error that does not terminate the preprocessor immediately. That is, the preprocessor will continue scanning the remaining cards of the input deck for errors. However, the user program will not be executed.

(4) Fatal error: An error that terminates the run immediately.

(5) N/A: Means <u>n</u>ot <u>a</u>pplicable.

(6) "TAPE": The term "tape" in quotes is used to signify any external storage device. That is, any piece of computer hardware, excluding the central processor, on which data can be stored and retrieved. The three most familiar examples are: magnetic tape, drum and disk.

(7) Dictionary: A list of the actual SINDA numbers in relative order. For example, the actual node number corresponding to the kth relative node number is the kth item of the node number dictionary.

(8) Data group: A data group composed of the pertinent information extracted from a particular data block. For example, the two groups derived from the constants data are: the user constants numbers and the user constants values.

(9) Bit manipulation: Terminology that implies the ability to store and access information <u>within</u> a computer word. This capability is also called packing and unpacking.

(10) Routine: A general term used to describe any program element.

(11) Subroutine: A special type of program element that is callable from a routine.

(12) Fixed constants: The term used in the preprocessor for control constants.

4.2 Description of Subroutines

Sections 4.2.1 through 4.2.30 below describe the 30 routines of the SINDA preprocessor. The descriptions are based on the UNIVAC 1108 computer under the EXEC II operating system; it should be understood, however, that much of the information is machine-dependent and is dependent upon the facilities operating system. Note that the element named SINDA (Section 4.2.1) and the element named PREPRØ (Section 4.2.2) are not subroutines in the technical sense of the word; hence, these two elements are referred to by the more general term "routine."

Each element of the preprocessor is described by the following eleven subtitles:

(1) SUBROUTINE NAME - this specifies the name of the element.

(2) PROGRAMMING LANGUAGE - This may be FØRTRAN, ASM, or MAP. FØRTRAN implies FØRTRAN V, ASM stands for assembly language (sometimes called SLEUTH II) and MAP is a special language which defines the overlay structure.

(3) PURPOSE - This gives a brief statement of the functional capabilities of the element.

(4) RESTRICTIONS - This gives an indication of where the input parameters come from, the form of the input parameters and the placement of the output parameters.

(5) "TAPES" USED - This represents a list of each FØRTRAN logical unit referenced within this element.

(6) SPECIAL FEATURES - This specifies programming features that are unique to a particular machine.

(7) OTHER SUBROUTINES CALLED - This represents a list of the external references.

(8) CALLING SEQUENCE - This gives a list of the subroutine arguments, if any, and a brief discussion of their use.

(9) ERROR PROCEDURES - This discusses the steps taken when an error is encountered.

(10) STORAGE REQUIRED - This gives the octal and decimal storage required for this element.

(11) LABELED COMMON - This represents a list of each labeled common name used in this element.

4.2.1 ROUTINE NAME: SINDA

PROGRAMMING LANGUAGE: MAP

PURPOSE: This routine specifies the overlay structure of the preprocessor to the system allocator (loader).

RESTRICTIONS: N/A

"TAPES" USED: N/A

SPECIAL FEATURES: N/A

OTHER SUBROUTINES USED: N/A

CALLING SEQUENCE: N/A

ERROR PROCEDURES: N/A

STORAGE REQUIRED: N/A

LABELED COMMON: N/A

4.2.2 ROUTINE NAME: PREPRØ

PROGRAMMING LANGUAGE: FØRTRAN

PURPOSE: This routine is the main routine (i.e., the driver) for the preprocessor. It initializes the counters and FORTRAN logical units, sets the length of the dynamic storage array, and controls the major logic. The major logic includes: (1) the EDIT feature (NASA MSC only); (2) the RECALL of a stored problem; (3) setup of a new user problem; (4) and preprocessor termination procedures.

RESTRICTIONS: N/A

"TAPES" USED:

System input "tape"	NIN
System output "tape"	NØUT
Problem data "tape"	LB3D
Problem FORTRAN "tape"	LB4P
Dictionary "tape"	LUT1
Parameter runs "tape"	LUT3
Recall "tape"	LUT7
Internal scratch "tape"	INTERN

SPECIAL FEATURES: System error termination - the problem data unit (LB3D) and the problem FORTRAN unit (LB4P) are flagged to stop before the data scan begins in the event that a system error terminates the preprocessor prematurely. The reason the problem data unit is flagged to stop is that for a RECALL problem the problem FORTRAN unit must not be written on.

OTHER SUBROUTINES USED: CØDERD, GENLNK, PRESUB, PSEUDØ, SINDA4, SPLIT and WRTBLK.

CALLING SEQUENCE: N/A

ERROR PROCEDURES: The error termination procedures are controlled by three flags named ERDATA, PRØGRM, and ENDRUN. The three flags are in the labeled common block named DATA. ERDATA is used to flag non-fatal errors encountered while reading the data blocks, while PRØGRM performs the same function for the operations blocks. See Section 4.7.2.

STORAGE REQUIRED: 443 octal words = 291 decimal words. See Section 4.7.1.

LABELED COMMON: BUCKET, CRDBLK, DATA, LØGIC, PLØGIC, and TAPE.

4.2.3 SUBROUTINE NAME: ALPINT

PURPOSE: This subroutine accepts an integer in the alphanumeric format nA1, and converts it to integer format, determines the relative number of this actual number, and converts the relative number back to an alphanumeric format of the form mA1.

RESTRICTIONS: The input and output is transmitted via the labeled common block named CIMAGE (see Section 4.3). The input must consist exclusively of the ten decimal digits.

"TAPES" USED: System output "tape" NØUT

SPECIAL FEATURES: None

OTHER SUBROUTINES USED: SEARCH

CALLING SEQUENCE: ALPINT(KLET,IST,IEND,J)

KLET	is an integer variable that indicates which dictionary is to be used for converting actual to relative.
IST	is the starting location of the alphanumeric integer.
IEND	is the ending location of the alphanumeric integer.
J	points to the last location + 1 of the converted integer.

ERROR PROCEDURES: In the event that a given actual number has no relative number in the dictionary list, an error message will be issued and the relative operations blocks error flag (PRØGRM) will be set to 1.0.

STORAGE REQUIRED: 665 octal words = 437 decimal words. See Section 4.7.1.

LABELED COMMON: BUCKET, CIMAGE, DATA PØINT, and TAPE.

4.2.4 SUBROUTINE NAME: BLKCRD

PROGRAMMING LANGUAGE: FØRTRAN

PURPOSE: This subroutine formats the five generated FØRTRAN routines (SINDA, EXECTN, VARBL1, VARBL2, and ØUTCAL) in 507 word blocks, which are acceptable to the FØRTRAN compiler. This information is stored in labeled common block CRDBLK, array KBLK. A complete discussion of the required tape format is found in UNIVAC 1108, EXEC II, Programmers Reference Manual, UP-4058 C, Appendix D.4 entitled, Program Elements on Magnetic Tape (via CUR).

RESTRICTIONS: The input is Hollerith card images with a 14A6 format. It is transmitted either through the array IMAGE in labeled common CRDBLK, or through "tape" INTERN.

"TAPES" USED: Internal scratch "tape" INTERN

SPECIAL FEATURES: None

OTHER SUBROUTINES USED: STFFB and WRTBLK

CALLING SEQUENCE: BLKCRD

ERROR PROCEDURES: none

STORAGE REQUIRED: 753 octal words = 491 decimal words. See Section 4.7.1.

LABELED COMMON: CRDBLK and TAPE.

4.2.5 SUBROUTINE NAME: CØDERD

PROGRAMMING LANGUAGE: FØRTRAN

PURPOSE: This subroutine reads and checks the block header cards for the data blocks. It also performs the following functions: (1) the second data card of the deck is checked for a thermal or general problem, and if it is a thermal problem the type of pseudo-compute sequence specified is noted; (2) the title block is read and processed; (3) the actual array and constant numbers from the automated options are converted into FØRTRAN addresses; and (4) the parameter run block header cards are read and checked.

RESTRICTIONS: None

"TAPES USED:	System input "tape"	NIN
	System output "tape"	NØUT
	FØRTRAN V reread	30

SPECIAL FEATURES: The FØRTRAN V intrinsic function FLD is used for bit manipulation.

CALLING SEQUENCE: CØDERD

ERROR PROCEDURES: In general, the errors checked for in this subroutine are of the fatal type; for example, data blocks out of order. The result of a fatal error is that the fatal error flag (ENDRUN) is set to 1.0 and control is returned to PREPRO for immediate termination.

STORAGE REQUIRED: 3213 octal words = 1675 words decimal. See Section 4.7.1.

LABELED COMMON: BUCKET, DATA, LØGIC, PLØGIC, and TAPE.

4.2.6 SUBROUTINE NAME: CØNVRT

PROGRAMMING LANGUAGE: FØRTRAN

PURPOSE: This subroutine converts Hollerith data to integer data.

RESTRICTIONS: The Hollerith data must be contained in one word and consist of only the ten decimal digits.

"TAPES" USED: None

SPECIAL FEATURES: The FØRTRAN V intrinsic function FLD is used for bit manipulation.

OTHER SUBROUTINES USED: ERRMES

CALLING SEQUENCE: CØNVRT(IST,IEND,ITEMP,CRDERR)

IST	is the pointer to the first bit of the first character.
IEND	is the pointer to the first bit of the last character.
ITEMP	is the word containing the Hollerith data on entry and the integer number on return.
CRDERR	is a logical error flag which is set true if an error is encountered during the conversion.

ERROR PROCEDURES: If a non-integer is encountered, an error message is printed and CRDERR is set to true.

STORAGE REQUIRED: 150 octal words = 104 decimal words. See Section 4.7.1.

LABELED COMMON: None

4.2.7 SUBROUTINE NAME: DATARD

PROGRAMMING LANGUAGE: FØRTRAN

PURPOSE: This subroutine scans the data block card images under an A format and determines the appropriate format (of the form Fn, In, or An) to reread the card image. The card images are then reread under the generated format. In addition, the constants data block and the array data block are processed.

RESTRICTIONS: None

"TAPES" USED:	System input "tape"	NIN
	System output "tape"	NØUT
	FØRTRAN V reread	30

SPECIAL FEATURES: The FORTRAN V intrinsic function FLD is used for bit manipulation.

OTHER SUBROUTINES USED: ERRMES, FINDRM, GENUK, NØDEDA (and its entry point CØNDDA), SETFMT, SQUEEZ, and TYPCHK.

CALLING SEQUENCE: DATARD

ERROR PROCEDURES: All errors checked for in this subroutine are non-fatal. An error message is printed either internally or from subroutine ERRMES and the data blocks error flag (ERDATA) is set to 1.0.

STORAGE REQUIRED: 5344 octal words = 2788 decimal words. See Section 4.7.1.

LABELED COMMON: BUCKET, CHECKD, DATA, FLAGS, LØGIC, PLØGIC, PØINT, and TAPE.

4.2.8 SUBROUTINE NAME: ERRMES

PROGRAMMING LANGUAGE: FØRTRAN

PURPOSE: This subroutine prints most of the error messages that can be generated within the data blocks.

RESTRICTIONS: None

"TAPES" USED: System output "tape" NØUT

SPECIAL FEATURES: None

OTHER SUBROUTINES USED: System subroutine EXIT

CALLING SEQUENCE: ERRMES(JUMP,I,J,K)

JUMP is an integer that points to the appropriate error message via a computed GØ TØ statement

I, J, K are data words which allow a maximum of three printed data words per error message.

ERROR PROCEDURES: If the number of error messages printed exceeds 199, the preprocessor is terminated by a call to EXIT.

STORAGE REQUIRED: 2166 octal words = 1142 decimal words. See Section 4.7.1.

LABELED COMMON: DATA and TAPE

4.2.9 SUBROUTINE NAME: FINDRM

PROGRAMMING LANGUAGE: FØRTRAN

PURPOSE: This subroutine moves the data in the dynamic storage array either up or down by 100 words. In the process it may delete certain groups of data that are no longer needed.

RESTRICTIONS: None

"TAPES" USED: System output "tape" NØUT

SPECIAL FEATURES: None

OTHER SUBROUTINES USED: SQUEEZ, and system subroutine EXIT.

CALLING SEQUENCE: FINDRM(LØCNØ,M)

LØCNØ is a pointer to a portion of the dynamic storage array where the data group that needs more room resides.

M is the address where the next data value is to be stored.

ERROR PROCEDURES: If the dynamic storage array is full, an error message is printed and the preprocessor is terminated via CALL EXIT.

STORAGE REQUIRED: 407 octal words = 263 decimal words. See Section 4.7.1.

LABELED COMMON: BUCKET LØGIC PØINT, and TAPE.

4.2.10 SUBROUTINE NAME: GENLNK

PROGRAMMING LANGUAGE: FØRTRAN

PURPOSE: This subroutine generates the driver, FORTRAN routine name and SINDA, for the user's program.

RESTRICTIONS: None

"TAPES" USED: Internal Scratch "tape" INTERN

SPECIAL FEATURES: None

OTHER SUBROUTINES USED: BLKCRD

CALLING SEQUENCE: GENLNK

ERROR PROCEDURES: None

LABELED COMMON: CRDBLK, DATA, LØGIC, PLØGIC, and TAPE.

4.2.11 SUBROUTINE NAME: GENUK

PROGRAMMING LANGUAGE: FØRTRAN

PURPOSE: This subroutine is used to generate user constants.

RESTRICTIONS: The input data is taken from array TEMP in labeled common CHECKD and the output data (i.e., the generated user constants) is put into array B in labeled common BUCKET.

"TAPES" USED: None

SPECIAL FEATURES: None

OTHER SUBROUTINES USED: ERRMES, FINDRM, and TYPCHK.

CALLING SEQUENCE: GENUK(IWRDS)

IWRDS is the number of words to be processed in array TEMP.

ERROR PROCEDURES: The input data is checked and if an error is found, control is transferred to subroutine ERRMES.

STORAGE REQUIRED: 451 octal words = 297 decimal words. See Section 4.7.1.

LABELED COMMON: BUCKET, CHECKD, DATA, and PØINT.

4.2.12 SUBROUTINE NAME: INCØRE

PROGRAMMING LANGUAGE: FØRTRAN

PURPOSE: This subroutine reads data into the dynamic storage array for the parameter runs option.

RESTRICTIONS: None

"TAPES" USED: Dictionary "tape" LUT1
Parameter runs "tape" LUT3

SPECIAL FEATURES: None

OTHER SUBROUTINES USED: None

CALLING SEQUENCE: INCØRE(ITEST)

ITEST is an integer flag which determines the data group group to be read.

ERROR PROCEDURES: None

STORAGE REQUIRED: 600 octal words = 384 decimal words. See Section 4.7.1.

LABELED COMMON: BUCKET, DATA, LØGIC, PLØGIC, PØINT, and TAPE.

4.2.13 SUBROUTINE NAME: MXTØFN

PROGRAMMING LANGUAGE: FØRTRAN

PURPOSE: This subroutine processes the data for the "M" option. That is, it converts card images from mixed FØRTRAN/SINDA notation to FØRTRAN notation.

RESTRICTIONS: The input (array IHØLL) and output (array JHØLL) are both in labeled common CIMAGE and they are both in an 80A1 format. The FØRTRAN from array JHØLL is copied to array IMAGE under a 14A6 format for processing by the FØRTRAN compiler.

"TAPES" USED: None

SPECIAL FEATURES: The FORTRAN V intrinsic function FLD is used for bit manipulation.

OTHER SUBROUTINES USED: ALPINT and BLKCRD

CALLING SEQUENCE: MXTØFN

ERROR PROCEDURES: None

STORAGE REQUIRED: 522 octal words = 338 decimal words. See Section 4.7.1.

LABELED COMMON: CIMAGE and CRDBLK

4.2.14 SUBROUTINE NAME: NØDEDA

PROGRAMMING LANGUAGE FØRTRAN

PURPOSE: This subroutine processes the data for the node and conductor data blocks.

RESTRICTIONS: The input is received via labeled common CHECKD: array TEMP and the processed data are stored in the dynamic storage array.

"TAPES" USED: None

SPECIAL FEATURES: This subroutine has a second entry point named CØNDDA. Also, the FØRTRAN V intrinsic function FLD is used for bit manipulation.

OTHER SUBROUTINES USED: ERRMES, FINDRM, RELACT, and TYPCHK

CALLING SEQUENCE: NØDEDA(JUMP,IWRDS)
or CØNDDA(JUMP,IWRDS)

JUMP is a flag which indicates which code option (columns 8, 9, and 10 of the data card) the user has selected.

IWRDS is the number of data values in array TEMP to be processed.

ERROR PROCEDURES: If an error is detected while scanning the input data, control is transferred to subroutine ERRMES.

STORAGE REQUIRED: 7030 octal words 3608 decimal words. See Section 4.7.1.

LABELED COMMON: BUCKET, CHECKD, DATA, FLAGS, and PØINT.

4.2.15 SUBROUTINE NAME: PCS2

PROGRAMMING LANGUAGE: FØRTRAN

PURPOSE: This subroutine packs the FØRTRAN addresses for the array and constants locations required by the pseudo-compute sequence.

RESTRICTIONS: None

"TAPES" USED: None

SPECIAL FEATURES: The FØRTRAN V intrinsic function FLD is used for bit manipulation.

OTHER SUBROUTINES USED: None

CALLING SEQUENCE: PCS2(IB,IPCS,LITA)

IB	is the word in the dynamic storage array where the addresses are found.
IPCS	is the word into which the addresses are packed.
LITA	is a flag that is set to 1 if the array address was input as a literal and therefore has been added to the constants data.

ERROR PROCEDURES: None

STORAGE REQUIRED: 54 octal words = 44 decimal words. See Section 4.7.1.

LABELED COMMON: None

4.2.16 SUBROUTINE NAME: PRESUB

PROGRAMMING LANGUAGE: FØRTRAN

PURPOSE: This subroutine reads and checks the block header cards for the operations blocks and generates the non-executable FØRTRAN cards for each of the operations blocks via a call to BLKCRD.

RESTRICTIONS: None

"TAPES" USED: System input "tape" NIN
System output "tape" NØUT

SPECIAL FEATURES: None

OTHER SUBROUTINES USED: BLKCRD

CALLING SEQUENCE: PRESUB(N)

N is an integer from 1 to 4 which indicates which operations block is being processed.

ERROR PROCEDURES: If the card read is not the correct block header card, an error message is printed and the fatal error flag is set to 1.0.

STORAGE REQUIRED: 200 octal words = 128 decimal words. See Section 4.7.1.

LABELED COMMON: CRDBLK, DATA, LØGIC, and TAPE.

4.2.17 SUBROUTINE NAME: PSEUDØ

PROGRAMMING LANGUAGE: FØRTRAN

PURPOSE: This subroutine forms the first and second pseudo-compute sequence. See Section 4.6.

RESTRICTIONS: The necessary input is extracted from the dynamic storage array and the output (the two pseudo-compute sequences) is placed in the dynamic storage array.

"TAPES" USED: NØRT

SPECIAL FEATURES: The FØRTRAN V intrinsic function FLD is used for bit manipulation.

OTHER SUBROUTINES USED: FINDRM, PCS2 and WRTDTA.

CALLING SEQUENCE: PSEUDØ

ERROR PROCEDURES: If an error is encountered while forming the pseudo-compute sequences, an error message will be printed and the non-fatal error flag (ERDATA) is set to 1.0.

STORAGE REQUIRED: 2244 octal words = 1188 decimal words. See Section 4.7.1.

LABELED COMMON: BUCKET, DATA, LØGIC, PLØGIC, and TAPE.

4.2.18 SUBROUTINE NAME: QDATA

PROGRAMMING LANGUAGE: FØRTRAN

PURPOSE: This subroutine checks and processes all data input in the source data block.

RESTRICTIONS: The input is received from the calling sequence and labeled common CHECKD. The processed data is placed in the dynamic storage array.

"TAPES" USED: None

SPECIAL FEATURES: The FØRTRAN V intrinsic function FLD is used for bit manipulation.

OTHER SUBROUTINES USED: ERRMES, FINDRM, RELACT, and TYPCHK.

CALLING SEQUENCE: QDATA(CØDE,IWRDS)

CØDE is the three letter option from columns 8, 9, and 10 of the data card.

IWRDS is the number of words in array TEMP to be processed.

ERROR PROCEDURES: If an error is encountered, control is transferred to subroutine ERRMES.

STORAGE REQUIRED: 1655 octal words = 941 decimal words. See Section 4.7.1.

LABELED COMMON: BUCKET, CHECKD, and PØINT.

4.2.19 SUBROUTINE NAME: RELACT

PROGRAMMING LANGUAGE: FØRTRAN

PURPOSE: This subroutine finds the relative node number from the actual node number. In addition, it computes the FØRTRAN address for arrays and user constants from the actual number.

RESTRICTIONS: This subroutine is used in conjunction with the data blocks.

"TAPES" USED: None

SPECIAL FEATURES: The FØRTRAN V intrinsic function FLD is used for bit manipulation.

OTHER SUBROUTINE USED: ERRMES

CALLING SEQUENCE: RELACT(K,MM,J,JJ)

K — determines the path through the program via a computed GØ TØ statement.

MM — is the actual number on entry, and the FØRTRAN address on return.

J, JJ — are print variables for subroutine ERRMES.

ERROR PROCEDURES: In the event an error is encountered, control is transferred to subroutine ERRMES.

STORAGE REQUIRED: 311 octal words = 201 decimal words. See Section 4.7.1.

LABELED COMMON: BUCKET, DATA, and PØINT.

4.2.20 SUBROUTINE NAME: SEARCH

PROGRAMMING LANGUAGE: FØRTRAN

PURPOSE: This subroutine returns a relative number for nodes, conductors, user constants, and arrays, given the actual number.

RESTRICTIONS: This subroutine is used in conjunction with the operations blocks.

"TAPES" USED: None

SPECIAL FEATURES: The FØRTRAN V intrinsic function FLD is used for bit manipulation.

OTHER SUBROUTINES USED: None

CALLING SEQUENCE: SEARCH(N,IA,NDIM,LØC)

N	is the actual number.
IA	is the first word of the dictionary of actual numbers to be searched.
NDIM	is the number of words of IA to be searched.
LØC	is the relative number returned to the calling program.

ERROR PROCEDURES: If the input actual number is not found in the dictionary, LØC is set to zero.

STORAGE REQUIRED: 101 octal words = 65 decimal words. See Section 4.7.1.

LABELED COMMON: None

4.2.21 SUBROUTINE NAME: SETFMT

PROGRAMMING LANGUAGE: FØRTRAN

PURPOSE: This subroutine processes the cards for the "new format" option; that is, it sets up the format for data cards as specified by the cards with a N in column one.

RESTRICTIONS: The input/output array is passed through the calling sequence argument.

"TAPES" USED: FØRTRAN V reread 30

SPECIAL FEATURES: None

OTHER SUBROUTINES USED: None

CALLING SEQUENCE: SETFMT(JUMP,B)

JUMP is an integer flag that determines the path through the code.

B is an array which contains the card images.

ERROR PROCEDURES: None

STORAGE REQUIRED: 221 octal words = 145 decimal words. See Section 4.7.1.

LABELED COMMON: None

4.2.22 SUBROUTINE NAME: SINDA4

PROGRAMMING LANGUAGE: FØRTRAN

PURPOSE: This subroutine reads and processes the user input cards from the operations blocks.

RESTRICTIONS: None

"TAPES" USED:	System input "tape"	MIN
	System output "tape"	NØUT
	Internal scratch "tape"	INTERN
	FØRTRAN V reread	30

SPECIAL FEATURES: None

OTHER SUBROUTINES USED: BLKCRD, MXTØFN, and SEARCH.

CALLING SEQUENCE: SINDA4(NAME)

NAME is an integer flag that tells the subroutine which operations block is being processed.

ERROR PROCEDURES: In the event an error is encountered while processing the operations blocks, an error message is printed and the error flag PROGRM is set to 1.0.

STORAGE REQUIRED: 2372 octal words = 1274 decimal words. See Section 4.7.1.

LABELED COMMON: BUCKET, CIMAGE, CRDBLK, DATA, LØGIC, PLØGIC, PØINT, and TAPE.

4.2.23 SUBROUTINE NAME: SKIP

PROGRAMMING LANGUAGE: FØRTRAN

PURPOSE: This subroutine is used when a problem is being RECALLed. It positions the tape to the proper problem as specified on the first card of the data deck.

RESTRICTIONS: The data is read from tape R. There is no output.

"TAPES" USED: RECALL "tape" LUT7

SPECIAL FEATURES: None

OTHER SUBROUTINES USED: None

CALLING SEQUENCE: SKIP

ERROR PROCEDURES: None

STORAGE REQUIRED: 324 octal words = 212 decimal words. See Section 4.7.1.

LABELED COMMON: TAPE

4.2.24 SUBROUTINE NAME: SPLIT

PROGRAMMING LANGUAGE: FØRTRAN

PURPOSE: This subroutine reads the data from the RECALL tape and splits the RECALL information onto the program data "tape" (LB3D) and the dictionary "tape" (LUT1).

RESTRICTIONS: The input is from the RECALL "tape" and the output is placed on the program data "tape," the dictionary "tape," and the parameter runs "tape."

"TAPES" USED:	RECALL "tape"	LUT7
	Program data "tape"	LB3D
	Dictionary "tape"	LUT1
	Parameter runs "tape"	LUT3

SPECIAL FEATURES: None

OTHER SUBROUTINES CALLED: SKIP

CALLING SEQUENCE: SPLIT(ID)

ID is the RECALL name punched in the first card of the data deck.

ERROR PROCEDURES: None

STORAGE REQUIRED: 746 octal words = 486 decimal words. See Section 4.7.1.

LABELED COMMON: BUCKET, DATA, and TAPE.

4.2.25 SUBROUTINE NAME: SQUEEZ

PROGRAMMING LANGUAGE: FØRTRAN

PURPOSE: This subroutine compresses the specified data groups in the dynamic storage array. The compression is accomplished by placing the data groups sequentially in the dynamic storage array.

RESTRICTIONS: None

"TAPES" USED: None

SPECIAL FEATURES: None

OTHER SUBROUTINES USED: None

CALLING SEQUENCE: SQUEEZ(IST,IEND)

IST is the data group number where the compression is to start.

IEND is the last data group number for this compression.

ERROR PROCEDURES: None

STORAGE REQUIRED: 115 octal words = 77 decimal words. See Section 4.7.1.

LABELED COMMON: BUCKET and PØINT

4.2.26 SUBROUTINE NAME: STFFB

PROGRAMMING LANGUAGE: FØRTRAN

PURPOSE: This subroutine fills out a card image in array KBLK with Hollerith blanks.

RESTRICTIONS: The pointers to the words to set to blank are in the calling sequence, and the array containing the card images is in labeled common CRDBLK.

"TAPES" USED: None

SPECIAL FEATURES: None

OTHER SUBROUTINES USED: None

CALLING SEQUENCE: STFFB(I,J)

I is the first work in KBLK to set to blank.

J is the last word in KBLK to set to blank.

ERROR PROCEDURES: None

STORAGE REQUIRED: 41 octal words = 33 decimal words. See Section 4.7.1.

LABELED COMMON: CRDBLK.

4.2.27 SUBROUTINE NAME: TYPCHK

PROGRAMMING LANGUAGE: FØRTRAN

PURPOSE: This subroutine checks the input from the data blocks for the correct type; type means integer, floating point, or alphanumeric. Also, it regulates the conversion of the A's and K's for the automated options via a call to CØNVRT.

RESTRICTIONS: The input and output are transferred through the calling sequence arguments and labeled common CHECKD.

"TAPES" USED: None

SPECIAL FEATURES: The FØRTRAN V intrinsic function FLD is used for bit manipulation.

OTHER SUBROUTINES USED: CØNVRT and ERRMES

CALLING SEQUENCE: TYPCHK(JUMP,IERR,J)

JUMP	indicates what type the word should be.
IERR	tells subroutine ERRMES which error message to print if the word is not of the type indicated by JUMP.
J	is a pointer to the word type in array KFLFX.

ERROR PROCEDURES: If a word is not of the proper type control is transferred to subroutine ERRMES to print an error message and the logical flag CRDERR is set to true.

STORAGE REQUIRED: 233 octal words = 155 decimal words. See Section 4.7.1.

LABELED COMMON: CHECKD

4.2.28 SUBROUTINE NAME: WRTDTA

PROGRAMMING LANGUAGE: FØRTRAN

PURPOSE: This subroutine writes the program data "tape" in the format required by INPUTT or INPUTG.

RESTRICTIONS: The data to be written on "tape" is found in the dynamic storage array.

"TAPES" USED: Program data "tape" LB3D

SPECIAL FEATURES: None

OTHER SUBROUTINES USED: None

CALLING SEQUENCE: WRTDTA(JUMP)

JUMP is an integer flag that indicates which data block to write and what format to use.

ERROR PROCEDURES: None

STORAGE REQUIRED: 645 octal words 421 decimal words. See Section 4.7.1.

LABELED COMMON: BUCKET, DATA, LØGIC, PLØGIC, PØINT, and TAPE

4.2.29 SUBROUTINE NAME: WRTPMT

PROGRAMMING LANGUAGE: FØRTRAN

PURPOSE: This subroutine writes the data that is needed for parameter runs on the parameter runs "tape" and writes the dictionary "tape."

RESTRICTIONS: The information that is written on the "tapes" is found in the dynamic storage array.

"TAPES" USED:	Dictionary "tape"	LUT1
	Parameter runs "tape"	LUT3

SPECIAL FEATURES: None

OTHER SUBROUTINES USED: None

CALLING SEQUENCE: WRTPMT(JUMP)

JUMP is an integer flag that indicates to WRTPMT which set of information to write.

ERROR PROCEDURES: None

STORAGE REQUIRED: 401 octal words = 257 decimal words. See Section 4.7.1.

LABELED COMMON: BUCKET, DATA, LØGIC, PØINT, and TAPE.

4.2.30 SUBROUTINE NAME: WRTBLK

PROGRAMMING LANGUAGE: Assembly Language

PURPOSE: This subroutine writes the 507 word blocks contained in array KBLK on the program FØRTRAN "tape."

RESTRICTIONS: None

"TAPES" USED: Program FØRTRAN "tape" LB4P

SPECIAL FEATURES: None

OTHER SUBROUTINES USED: None

CALLING SEQUENCE: WRTBLK

ERROR PROCEDURES: None

STORAGE REQUIRED: 14 octal words = 12 decimal words. See Section 4.7.1.

LABELED COMMON: CRDBLK

4.3 LABELED CØMMØN VARIABLES

The SINDA preprocessor uses nine labeled common blocks to pass data and flags between the various subroutines. Labeled common names, in alphabetical order, are:

BUCKET	CHECKED	CIMAGE
CRDBLK	DATA	FLAGS
LØGIC	PLØGIC	PØINT

Note that the UNIVAC 1108 version does not utilize blank common. The two sections that follow give: 1) a map of the labeled common usage by subroutine name and by overlay link; 2) a definition of the variables used within each labeled common block; and 3) dynamic storage structure.

4.3.1 Labeled Common Map

The map below gives the labeled common name, a list of the overlay links that use it by link number and a list of the routines that use it.

LABELED CØMMØN NAME	ØVERLAY LINK NAMES	RØUTINE NAMES	
BUCKET	0, 1, 2, 4, 5	ALPINT	CØDERD
		DATARD	FINDRM
		GENUK	INCØRE
		NØDEDA	PREPRØ
		PSEUDØ	QDATA
		RELACT	SINDA4
		SPLIT	SQUEEZ
		WRTDTA	WRTPMT
CHECKD	1	DATARD	GENUK
		NØDEDA	QDATA
		TYPCHK	
CIMAGE	4	ALPINT	MXTØFN
		SINDA4	
CRDBLK	0, 3, 4	BLKCRD	GENLNK
		MXTØFN	PREPRØ
		PRESUB	SINDA4
		STFFB	WRTBLK

LABELED CØMMØN NAME	ØVERLAY LINK NAMES	RØUTINE NAMES	
DATA	0, 1, 2, 3, 4, 5	ALPINT	CØDERD
		DATARD	ERRMES
		GENLINK	GENUK
		INCØRE	NØDEDA
		PREPRØ	PRESUB
		PSEUDØ	RELACT
		SINDA4	SPLIT
		WRTDTA	WRTPMT
FLAGS	1	DATARD	NØDEDA
LØGIC	0, 1, 2	CØDERD	DATARD
		FINDRM	INCØRE
		PREPRØ	PSEUDØ
		WRTDTA	WRTPMT
PLØGIC	0, 1, 3, 4	CØDERD	DATARD
		GENLNK	INCØRE
		PREPRØ	SINDA4
		WRTDTA	
PØINT	0, 1, 2, 4	ALPINT	CØDERD
		DATARD	FINDRM
		GENUK	INCØRE
		NØDEDA	PREPRØ
		PSEUDØ	QDATA
		RELACT	SINDA4
		SQUEEZ	WRTDTA
		WRTPMT	
TAPE	0, 1, 2, 3, 4, 5	ALPINT	BLKCRD
		CØDERD	DATARD
		ERRMES	FINDRM
		GENLNK	INCØRE
		PREPRØ	PRESUB
		PSEUDØ	SINDA4
		SKIP	SPLIT
		WRTDTA	WRTPMT

4.3.2 Definition of Labeled Common Variables

(1) Labeled common name BUCKET.

BUCKET is the dynamic storage array (see Section 4.3.3).

(2) Labeled common name CHECKD

CHECKD is used to temporarily store and check the user's input data.

VARIABLE NAME	DESCRIPTION
TEMP(35) or ITEMP	A temporary storage array, which contains the user's input data as read from the data cards.
XGEN(35) or IGEN	A temporary storage array used to store a copy of TEMP when the user is generating data.
KFLFX(35)	An indicator used to check the data array which contains one of the following numbers: 1 = floating point number 0 = integer number -1 = Hollerith word
CRDERR	A logical flag: Set true if and only if an error was found on the data card now being processed.

(3) Labeled common name CIMAGE

CIMAGE is used to store and manipulate Hollerith card images for the 'M' option.

VARIABLE NAME	DESCRIPTION
IHØLL(80)	The input card image, read under an 80A1 format.
JHØLL(160)	The constructed card image, also an 80A1 format.

(4) Labeled common name CRDBLK

CRDBLK is used to construct the five generated FØRTRAN routines.

VARIABLE NAME	DESCRIPTION
LSTART	A logical flag that signals the start of a new routine if set to true.
LECARD	A logical flag that signals the end of a routine.
LCØPY	A logical flag that tells the program to copy the card image (14A6) found in IMAGE to the next available slot in KBLK.
NW	Is a counter whose value is the next available word in KBLK.
KBLK(507)	An array that contains FØRTRAN card images of the generated routines.
IMAGE(14)	An array that contains one card image to be copied into KBLK.

(5) Labeled common name DATA

DATA is used to store the counters that indicate (to the program) how many of each data type has been encountered. In addition, it contains three error flags.

VARIABLE NAME	DESCRIPTION
NND	The number of diffusion nodes.
NNA	The number of arithmetic nodes.
NNB	The number of boundary nodes.
NNT	The total number of nodes.
NGL	The number of linear conductors.
NGR	The number of radiation conductors.
NGT	The total number of conductors.
NUC	The number of user constants.
NEC1	The number of added constants from automated options in the node data block.

NEC2	The number of added constants from the automated options in the conductor data block.
NCT	The total number of constants.
LENA	The total number of words used in the array data block.
ERDATA	The non-fatal error flag for the data blocks. ERDATA ≠ 0.0 means an error has been found.
PRØGRAM	The non-fatal error flag for the operations blocks. PRØGRAM ≠ 0.0 indicates an error condition.
ENDRUN	The fatal error flag for the preprocessor. ENDRUN ≠ 0.0 signals the program to terminate immediately.
LSEQ1	The length of the first pseudo-compute sequence.
LSEQ2	The length of the second pseudo-compute sequence.
LØNG	A logical flag set to true if the user is requesting the long pseudo-compute sequence.

(6) Labeled common name FLAGS

FLAGS contains three flags that are used to go to the proper block of coding in subroutine NØDEDA.

VARIABLE NAME	DESCRIPTION
LEAP	Used with the GEN option.
NØNLIN	Flags a set of multiply connected conductors as radiation, if set to true.
INDX	Determines path when multiply connected conductors require more than one data card.

(7) Labeled common name LØGIC

LØGIC contains a number of logical flags and the fifty fixed constants.

VARIABLE NAME	DESCRIPTION
LNØDE	Set to true if any node data was processed.
LCØND	Set to true if any conductor data was processed.
LCØNST	Set to true if any user constants were processed
LARRAY	Set to true if any array data was processed.
LPRINT	Debug print flag, set to true if there is an asterisk in column 80 of the BCD 3 THERMAL/GENERAL card.
KBRNCH	An integer that specifies which data block is being processed.
FIXC(50) or IFIXC	The array that contains the fixed (control) constants.
KTPRNT	Optional print flag for a list of relative versus actual user constant numbers. Set time if there is an asterisk in column 80 of the BCD 3CØNSTANTS DATA card.
AYPRNT	Optional print flag for a list of actual array numbers versus FØRTRAN address. Set true if there is an asterisk in column 80 of the BCD 3ARRAY DATA card.
GENERL	Set true for a general problem.
LQ	Set true if any data was processed from the source data block.

(8) Labeled common name PLØGIC

PLØGIC contains a number of logical flags that are used in conjunction with parameter runs.

VARIABLE NAME	DESCRIPTION
PARINT	Set true for initial parameters run.
PARFIN	Set true for final parameters run.
PNØDE	Set true if node data was processed.
PCØND	Set true if conductor data was processed.
PCØNST	Set true if user constants data was processed.

PARRAY	Set true if array data was processed.
PTITLE	Set true if a new title was input.
PCHGID	Contains the alphanumeric word INITIAL or FINAL to be used as the run identification on "tape" LB3D.

(9) Labeled common name PØINT

PØINT is used in conjunction with dynamic storage array BUCKET. See Section 4.3.3.

4.3.3 Dynamic Storage Structure

Dynamic storage represents one of the techniques of maximizing problem size with a computer with finite core. In dynamic storage each data set is placed sequentially into one array end-to-end. This eliminates the wasted core inherent with the traditional system of dimensioning each variable at some fixed length. However, the price paid for the additional core is the extra time required to compute the address of a variable.

The SINDA preprocessor used three arrays to store and address the data sets. The data sets are stored in an array named B, or IB, or BB. This array resides in labeled common BUCKET. The length at which B can be dimensioned depends on the system that the computer facility uses. At NASA MSC approximately 30,000 words are allocated to B. In addition, in labeled common PØINT there are two arrays named LØC and LEN, each dimensioned at 20. LØC (I) contains the starting location in B for the Ith data set and LEN (I) contains the length of the Ith data set.

The information below gives, in detail, the contents of the dynamic storage array for each data block as it exists just after the data block has been processed.

(1) Node data block

data set 1:

bit 1,	automated option flag
bit 2,	Q from SØURCE DATA flag
bits 16-35,	actual node number

data set 2:

bits 0-35,	temperature value

data set 3:

bits 0-35,	capacitance value

data set 4:

bits 0-5	non-linear capacitance type
bit 6,	literal array flag
bits 7-20,	actual array number
bit 21,	literal constant flag
bits 22-35,	actual constant number

data set 5:

bits 0-35,	literals encountered in 4.

(2) Source data block

data set 2 (first word of group):

bits 0-5,	source option type
bits 6-20,	relative node number
bits 21-35,	not used

data set 2 (second word of group):

bits 0-5,	not used
bit 6,	literal array flag
bits 7-20,	actual array number
bit 21,	literal constant flag
bits 22-35,	actual constant number

data set 3:

bits 0-35,	literals encountered in 2

(3) Conductor data block

data set 6:

bits 0-35,	actual conductor number

data set 7:

bit 0,	multi connections flag
bit 1,	radiation flag
bit 2,	automated option flag

bits 3-5,	not used
bit 6,	1 way flag for NA
bits 7, 20,	relative node number NA
bit 21	1 way flag for NB
bits 22-35,	relative node number NB

data set 8:

bits 0-35,	conductance value

data set 9:

bits 0-5,	conductor option type
bit 6,	literal array flag
bits 7-20,	actual array number
bit 21,	literal constant flag
bits 22-35,	actual constant number

data set 10:

bits 0-35,	literals encountered in 9

(4) Constants data block

data set 11:

bits 0-35,	actual constant number

data set 12:

bits 0-35,	constant value

(5) Array data block

data set 13:

bits 0-35,	actual array number

data set 14:

bits 0-35,	array length

data set 15:

bits 0-35,	array value

(6) Pseudo-compute sequences

data set 16 (1st pseudo-compute sequence):

bit 0,	last conductor flag
bit 1,	automated capacitance flag
bit 2,	automated conductance flag

bit 3,	radiation conductance flag
bit 4,	Q from source block flag
bits 5-20,	relative conductor number
bit 21	1 way conductor flag
bits 22-35,	relative adjoining node number

data set 17 (second pseudo-compute sequence):

bits 0-4,	automated option type
bit 5,	not used
bits 6-21,	FØRTRAN address for array
bit 22,	not used
bits 23-35,	relative constant number

The bit numbering convention above conforms to the UNIVAC standard notation, where each 36 bit word is numbered 0 through 35 from left to right. Each of the 1 bit flags above is querried in the following manner: 0 means NO, and 1 means YES. If the literal array flag or the literal constant flag is set to 1, then the bits immediately to the right of the flag do not contain the actual array or constant number. Instead, they contain a pointer to the next data set where the literal value is stored. In those data sets that store information for the automated options it is sometimes necessary to use more than one word per option. When this is the case, the automated option type (bits 0-5) is set to 0.

4.4 SINDA "Tapes" and Their Formats

The SINDA program in its normal operating mode utilizes six "tapes." Five of these "tapes" are assigned by the program and the remaining one contains the program; it is assigned via control cards. The store and recall options require one additional "tape" each and the NASA edit feature requires two additional "tapes." The following paragraphs contain information on the five normal SINDA "tapes."

4.4.1 LB3D - Program Data "Tape"

This "tape" is set up by the preprocessor (WRTDTA) and read by INPUTT, for a thermal problem, or INPUTG, for a general problem, just prior to performing the instructions of the execution block. The contents of this unit are:

(1) Problem identification.

```
WRITE(LB3D)RUNID
```

(2) Title information (20 words).

```
WRITE(LB3D)(TITLE(I),I=1,20)
```

(3) The number of: diffusion nodes, arithmetic nodes, and total nodes; followed by a temperature value for each node; then a capacitance value for each diffusion, if any.

```
WRITE(LB3D)NND,NNA,NNT,(T(I),I=1,NNT)
IF(NND.GT.0)WRITE(LB3D)(C(I),I=1,NND)
```

(4) The total number of conductors followed by a conductor value for each one.

```
WRITE(LB3D)NGT,(G(I),I=1,NGT)
```

(5) The total number of user constants are followed by the 50 control constant values; then the user constants values, if any.

```
WRITE(LB3D)NCT,(FIXC(I),I=1,50)
IF(NCT.GT.0)WRITE(LBJD)(K(I),I=1,NCT)
```

(6) The total number of arrays and the overall length of the array data; then the array values, if any.

```
WRITE(LB3D)NAT,LENA
IF (LENA.GT.0)WRITE(A(I),I=1,LENA)
```

(7) The lengths of the first and second pseudo-compute sequences, followed by the data for the first pseudo-compute sequence; then the data for the second pseudo-compute sequence, if any.

```
WRITE(LB3D)LSEQ1,LSEQ2,(P1(I),I=1,LSEQ1)
IF(LSEQ2.GT.0)WRITE(LB3D)(P2(I),I=1,LSEQ2)
```

Note that (3), (4), and (7) above apply only to a thermal problem.

4.4.2 LB4P - Program FØRTRAN "Tape"

This "tape" is especially formatted in 507 word blocks as required by the FØRTRAN compiler. Where:

WORD 1 on the first block of each routine contains the name of the routine.

WORD 2 contains the integer number of card images in the block.

WORDS 3 - 506 contain the card images

WORD 507 is set to +0 except on the last block of each routine where it is set to -0.

4.4.3 INTERN - Preprocessor Scratch "Tape"

Generally INTERN is used to pass card images to subroutine BLKCRD under a 14A6 format.

4.4.4 LUT1 - Dictionary "Tape"

This "tape" contains a list of the actual SINDA numbers in a relative order. That is, the actual node number corresponding to the kth relative node number is the kth item of the node number dictionary. The format of this "tape" is:

(1) The total number of nodes, followed by an actual node number for each node.

```
WRITE(LUT1)NNT,(NN(I),I=1,NNT)
```

(2) The total number of conductors, followed by the list of actual conductor numbers.

```
WRITE(LUT1)NGT,(NG(I),I=1,NGT)
```

(3) The number of user constants, the total number of constants, followed by a list of the actual constant numbers.

```
WRITE(LUT1)NUC,NCT,(NK(I),I=1,NCT)
```

(4) The total number of arrays followed by a list of the actual array numbers, then the total number of arrays followed by a list of the length of each array.

```
WRITE(LUT1)NAT,(NA(I),I=1,NAT)
WRITE(LUT1)NAT,(LA(I),I=1,NAT)
```

4.4.5 LUT3 - Parameter Runs "Tape"

This "tape" contains some data from the original problem. It is required by the initial parameters capability. The format of "tape" LUT3 is:

(1) The original title.

```
WRITE(LUT3)(TITLE(I),I=1,20)
```

(2) A list of original temperature and capacitance values.

```
WRITE(LUT3)NND,(T(I),I=1,NNT)
IF(NND.GT.0)WRITE(LUT3)(C(I),I=1,NND)
```

(3) A list of the original conductor values.

```
WRITE(LUT3)(G(I),I=1,NGT)
```

(4) Lists of the original fixed and user constants.

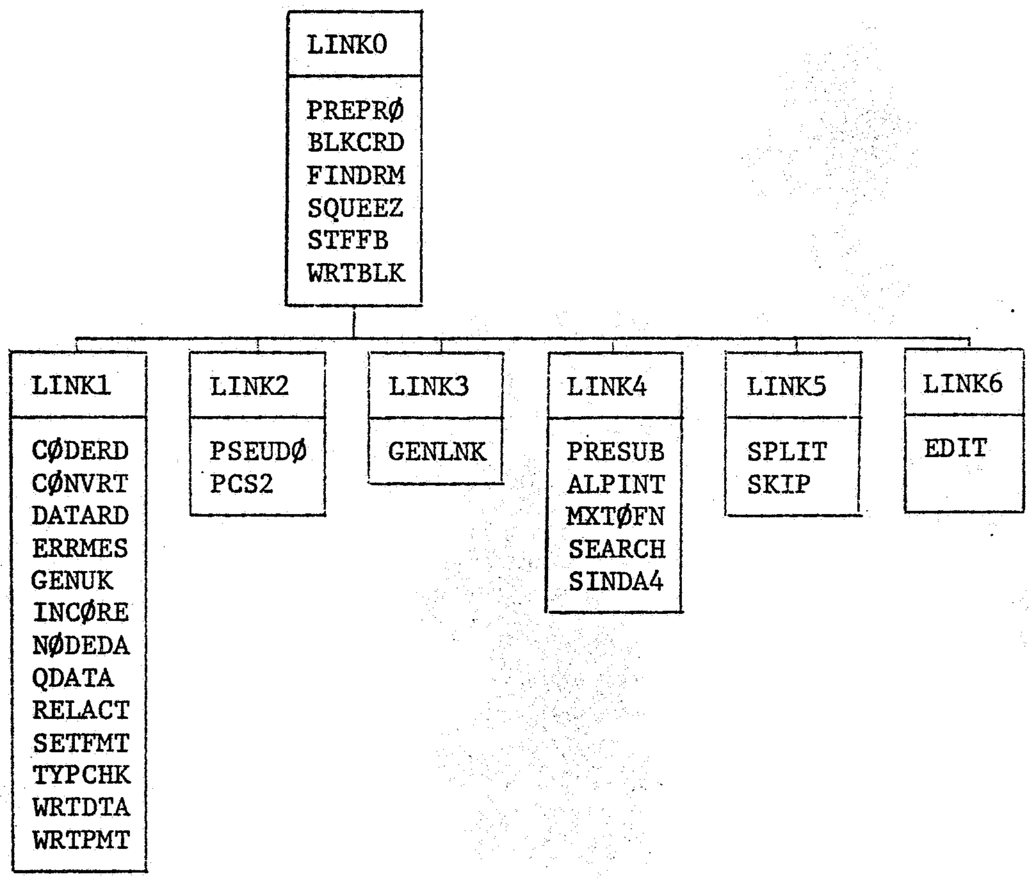
```
WRITE(LUT3)NUC,NCT,(FIXC(I),I=1,50)
IF(NCT.GT.0)WRITE(LUT3)(K(I),I=1,NCT)
```

(5) The original array values.

```
WRITE(LUT3)NAT,LENA
IF(LENA.GT.0)WRITE(LUT3)(A(I),I=1,LENA)
```

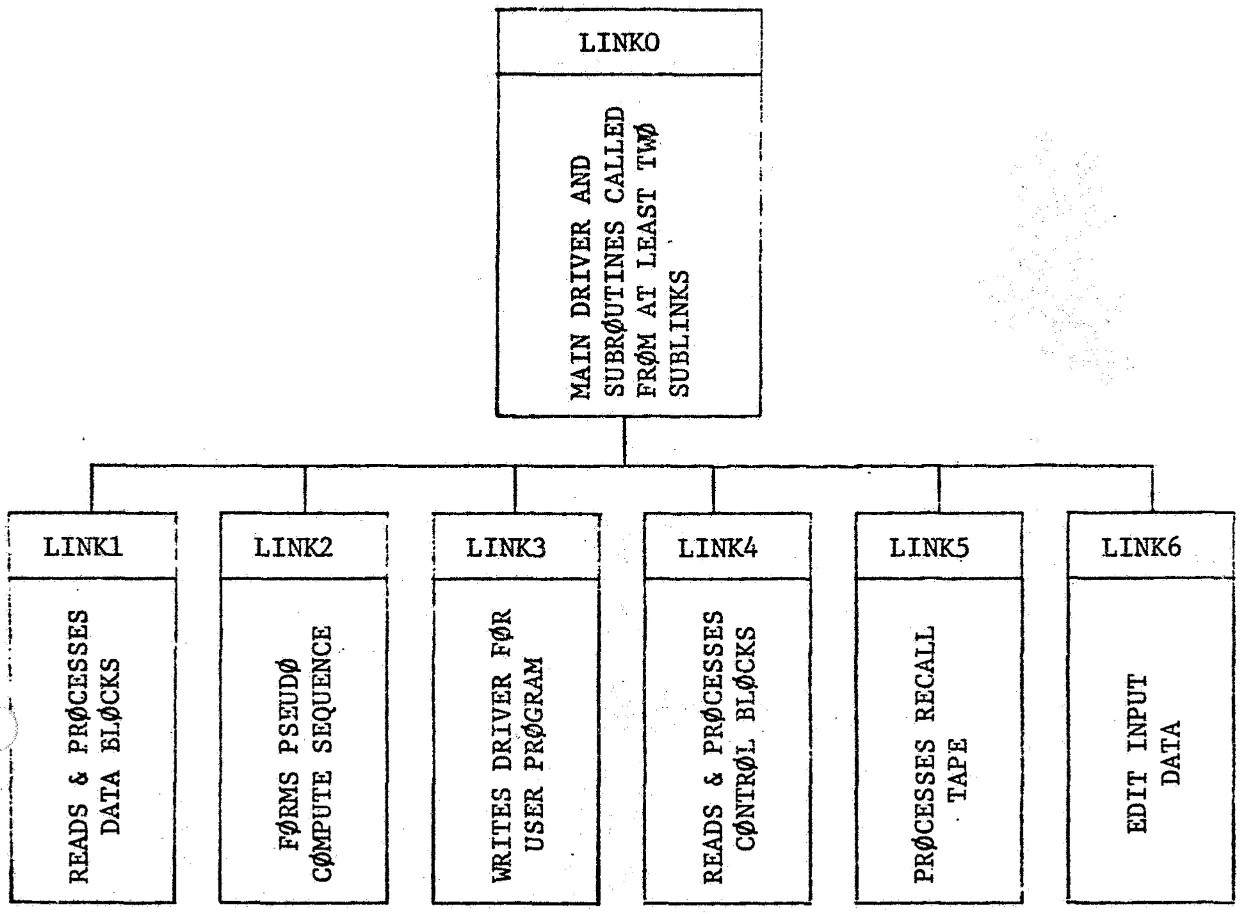
4.5 Overlay Structure

The SINDA preprocessor has an overlay structure composed of a main link (designated LINKO below) which is always in core and five sublinks (designated LINK1, LINK2, LINK3, LINK4, LINK5, and LINK6 below) which overlay one another as they are brought into core.



Note that the first subroutine listed above in each of the sublinks serves as the driver for that sublink and it is also the subroutine called from PREPRØ.

Another approach to overlay specification is to think of each link as a functional unit, hence the graph below.



4.6 Structure of Pseudo-Compute Sequences

4.6.1 Descriptions

The use and structure of the two pseudo-compute sequences generated by the SINDA preprocessor appear to be rather confusing and mysterious. The term "pseudo" itself leads to immediate interpretation difficulties. Suppose that an element G_k between nodes i and j is to be identified for the ith node; by specifying explicitly i,j, and k the element is completely defined. On the other hand, SINDA explicitly specifies j and k but i is implicit in the DØ-LØØP; hence, the description pseudo-compute sequence (PCS) arises. Confusion also arises from the lack of information regarding the need for the (PCS) and the difficulties in reading the packed information. In short, the PCS as used in SINDA is simply two lists of relative numbers which are ordered in a specific manner. The two lists of relative numbers form the heart of the PCS, although other information pertinent to the computation must also be considered.

The PCS is necessary because the data as input by the user does not lend itself efficiently to the computational capabilities of FØRTRAN. As a result, the preprocessor scans the user input data and places the relative numbers (FØRTRAN addresses) into an array in the order in which the data will be used at a later time by the user selected numerical solution routine.

Packing of the data is a technique that conserves computer storage by placing two or more pieces of information in one computer word. This allows the user to execute a larger problem than the one that can be accommodated if the traditional one computer word for each piece of information approach. The penalty for this larger problem capability is an increase in execution time required for the extraction of information each time it is used.

4.6.2 Structure of PCS1

The first PCS, designated PCS1, contains the following information:

bit 0	last G for this node flag
bit 1	automated C option flag
bit 2	automated G option flag
bit 3	radiation G flag
bit 4	Q from source data flag
bits 5-20	relative G number
bit 21	1 way G flag
bits 22-35	relative adjoining node number

The 36 bits of each computer word are numbered 0 through 35 from left to right. All of the 1 bit flags are set such that 0 means NØ and 1 means YES.

PCS1 is stored in an array named NSQ1 and is ordered by relative node number. That is, for relative node number 1 the conductor data is scanned and each time a conductor connected to node number 1 is encountered the PCS1 information is stored in NSQ1. When all of the conductor data has been scanned, the 0 bit of the latest word of PCS1 information is set to 1. This process is repeated for relative node numbers 2, 3, etc., until all diffusion and arithmetic nodes have been processed. PCS1 will be formed as either long or short as specified by the user on the BCD 3THERMAL card. This option is applied to diffusion nodes only since arithmetic nodes are always formed under the long option. The difference between the long PCS and the short PCS is that in the long PCS, each conductor will be listed twice, whereas in the short PCS each conductor will be listed once. This assumes the conductor connects two diffusion nodes. If one or both of the nodes is arithmetic, then the conductor will be listed twice, and if one of the nodes is a boundary the conductor will only be listed once. For example, given conductor number k which connects diffusion nodes i to j, where i < j. The long PCS would contain the k, j information for the processing of node i, and the k,i information for the processing of node j; whereas, the short PCS would only contain the k,j information. The short PCS thus has the advantage of requiring less computer storage than the long PCS, but a block iterative method (refer to Section 5.2.2) must be used; in general, the short PCS requires more iterations to converge than the successive point iterative (refer to Section 5.2.2) method which requires the long PCS.

4.6.3 Structure of PCS2

The second PCS is designated PCS2. The following information is stored whenever bit 1, bit 2, or bit 4 of PCS1 is set to one.

bits 0-4	automated option code
bit 5	not used
bits 6-21	FØRTRAN address of the array or relative constant number.
bit 22	not used
bits 23-35	relative constant number

If the automated option is a doublet type, like DIV, and therefore requires two words to store the information, the automated option code on the second word is set to zero. In the event that more than one of the flag bits (bits 1, 2, or 4) of PCS1 is set to one, then the following order is imposed on PCS2: the capacitance information is stored first, the source block information second and finally the conductor information.

The PCS2 information is stored in array named NSQ2. This array is the same under the long PCS1 or the short PCS1 since automated conductors are only flagged on their first encounter.

4.7 Other Information

This section contains miscellaneous information that may be of interest to the user.

4.7.1 Subroutine Lengths

The storage required by a particular routine will vary depending on the type of computer and the system being used. The routine lengths given in Section 4.2 are based on compiler listings made on 23 January 1971 at Jacobi Computation Center.* The machine is a UNIVAC 1108 with a highly modified system. The numbers represent the sum of the computer storage for computer instructions, constants, and simple variables.

4.7.2 Maximum Thermal Problem Size and Maximum Data Value Size

A short formula for estimating the maximum thermal problem size that can be run on SINDA, and a list of the maximum size of the various data values is given below.

* Now called Computation and Systems Corporation, Los Angeles, California.

Estimation of Maximum Problem Size

NNT + 3*NGT + NCT + 4*NAØ $\leq$ LENBKT

where,

NNT	is the total number of nodes.
NGT	is the total number of conductors.
NCT	is the total number of constants (user constants plus literals from automated options).
NAØ	is the number of automated options specified.
LENBKT	is the length of the dynamic storage array as set in routine PREPRØ.

Maximum Size of Data Values

Actual node number	
Core storage	$2^{33}-1$
Print out	999,999
Relative node number	
Core storage	16,383
Temperature	
Core storage	$\pm 10^{38}$
Capacitance	
Core Storage	$\pm 10^{38}$
Relative conductor number	
Core storage	$2^{35}-1$
Actual user constants number	
Core storage	32,767
Automated options	16,383
Relative user constants number	
Core storage	32,767
Automated options	8,191

User constant values	
Integer	$\pm 2^{35}-1$
Floating point	$\pm 10^{38}$
Alphanumeric	6 characters
Actual array number	
Core storage	$2^{35}-1$
Automated options	16,383
Print out	99,999
Relative array number	
Core storage	$2^{35}-1$
Automated options	65,535
Print out	99,999
Array values	
Integer	$\pm 2^{35}-1$
Floating point	$\pm 10^{38}$
Alphanumeric	6 characters

Note that some of the maxima, such as the relative conductor number of $2^{35}-1$, are strictly academic since the dynamic storage array is considerably smaller than the indicated maximum data value size.

5. REVIEW OF LUMPED PARAMETER EQUATIONS AND BASIC NUMERICAL SOLUTIONS

The use of SINDA as mentioned in a previous section is based on a lumped parameter representation of a physical system.[7] Thus SINDA solves numerically a set of ordinary (in general nonlinear) differential equations that represent the transient behavior of a lumped parameter system or a set of nonlinear algebraic equations representing steady state conditions. Numerous numerical solution techniques are reported in literature; a few of these are listed in the Reference Section.[8-24] These numerical methods are based on finite difference algorithms as opposed to finite element methods which have received considerable attention recently.[25-29] For problems that are generally encountered in spacecraft thermal design, use of the finite element method appears to be inappropriate because of the nonlinearity presented with radiation heat transfer and because of complex geometric configurations.

Variations of the basic finite difference algorithms are numerous because no single numerical solution technique is optimum for all the endless types of thermal problems that can be encountered. Furthermore, because of the nonlinearity of the problems, a specific set of criterions to indicate solution accuracy and stability is not available and does not appear to be forthcoming. As a result, the user is placed in a rather awkward and confused position of not knowing which subroutine to use if a choice is available. Some thermal analyzer-type computer programs allow no choice, as a result, user decision is not necessary. SINDA represents a computer program at the other extreme of user decision flexibility by providing a number of numerical solution methods.

The intent of this Section 5 is to review and formulate the basic numerical solution methods with the presentation (from an engineering standpoint) of the characteristics of each SINDA numerical solution routine deferred to Section 6. In addition to place the use of SINDA in a proper perspective relative to accurate temperature prediction of a physical system, difficulties associated with lumped-parameter representation are discussed here.

5.1 Lumped Parameter Representation

Reduction of a distributive (physical) system to a lumped system which can be represented as an equivalent thermal network is a rather important phase of thermal analysis. From a temperature accuracy standpoint lumping (or nodalization) of the physical system may be far more important than a numerical solution technique that is used in a computer program. The latter is often given undue attention with apparent ignorance of other error sources which may be far more important. A general discussion on lumped parameter representation is not intended for presentation here since the subject material is extensively covered in technical literature, but it is convenient for continuity to indicate basic considerations.

For simple geometries and linear problems, it is rather straightforward to solve the partial differential equations of the type,

$$\frac{\partial T}{\partial t} = \alpha \nabla^2 T + Q \qquad (5.1\text{-}1)$$

where, α = thermal diffusivity (k/C)

T = temperature

Q = source

$\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$ (two dimensional)

Numerous analytical solutions of (5.1-1) for different types of boundary conditions and geometries are available.[30,31] Finite difference algorithms formed directly from the partial differential equations are also abundantly reported in literature.[12,14] These finite difference formulations were generally developed for well-defined geometries and symmetrical discretization. For these problems, the so-called nodal connections or resistances are immediately available and, in general, automatically generated by the computer program. Thus, the need for a lumped parameter representation does not exist. For these types of problems, inaccuracies due to truncation and solution stability are specifically established.

For complex geometries and nonlinear problems such as those that include thermal radiation exchange, analytical solutions of thermal problems are limited.[32,34] As a result, it is a common practice because of practical considerations to nodalize a physical system directly without undue consideration of inaccuracies. Thus, a user merely represents the heat flow between two connecting nodes by using the basic network building block,

$$q_{ij} = (T_i - T_j)/R_{ij} \tag{5.1-2}$$

where, R_{ij} represents an effective resistance between adjoining nodes i and j.

It should be particularly noted here that SINDA employs the concept of conductance in lieu of resistance which is common with most network-type computer programs. Thus the heat flow is represented as:

$$q_{ij} = G_{ij}(T_i - T_j) \tag{5.1-3}$$

where, G_{ij} is the conductance between node i and node j.

The proper value of G_{ij} (or R_{ij}) for an arbitrary nodalization is (and should be) of concern to the user but because of the multitude of variables that must be considered, any discussion here would be incomplete. An excellent article on asymmetrical finite difference networks is presented in Reference 35.

5.1.1 Some Thoughts on Lumped Parameter Errors

Reduction of a physical system to a topological model consisting of a network with resistors and capacitors requires considerable engineering judgment. More often than not, nodal size of a model is governed by budget and schedule constraints. As a result, the discrete areas larger than desired are often used. This does not necessarily mean, however, that the use of a large number of nodes will always yield realistic results since the uncertainties of the input parameters can be appreciable.[36]

Spatial truncation errors are controlled by selecting the grid size so that nonlinear temperature distributions lie within required accuracy by linear interpolation between nodal points, and that variation of temperature-dependent properties over the volume of each node is within required limits of the average values determined for the nodal point. The assumption of linear temperature distribution assumed for the lumped-parameter equations (equation 5.1-6) leads to a spatial truncation error of the order $0(\Delta x^2)$ only if all nodes are symetrically located.[27] If a non-uniform grid is used, the accuracy of computation is only $0(\Delta x)$. Spatial truncation errors are thus inherent in the mathematical model and beyond user control once inputted into

SINDA. For a two-dimensional problem with symmetrical grids, the spatial truncation error can be expressed for typical explicit and implicit methods as,[12,27]

$$E = -\frac{(\Delta x)^2}{12}\frac{\partial^4 T}{\partial x^4} - \frac{(\Delta y)^2}{12}\frac{\partial^4 T}{\partial y^4} + 0(\Delta x^4) \qquad (5.1\text{-}4)$$

Temperature distribution other than linear can also be formulated;[38] however, most thermal analyzer-type computer programs such as TRUMP[39] and including SINDA are based on the linear assumption.

Time truncation errors are directly dependent upon the time-step since the error for the typical explicit and implicit method is,

$$E = -\frac{\Delta t}{2}\frac{\partial^2 T}{\partial t^2} \qquad (5.1\text{-}5)$$

Normally the time step is dependent upon a particular criterion chosen by the user. A more detailed discussion on user control of the time step will be given for each numerical solution routine within the SINDA subroutine library.[3,4]

Another approximation error which is due to discretizing is the assumption of constant radiosity for the discrete areas. Inaccuracies can be expected to affect the level and distribution of temperature. The analysis of thermal radiation exchange has received considerable attention in recent years because of its importance in spacecraft thermal design.[40,52] The influence of non-uniform local heat flux on overall heat transfer between a gray differential area parallel to a gray infinite plane is examined in Reference 43; the assumption of uniform local heat flux appears to be reasonable for this geometry and for the evaluation of the overall heat flux calculations. A method of analysis suitable for engineering applications is developed in Reference 50 for computing local radiant flux and local temperature of opaque surfaces in a space environment. A study evaluating the validity of commonly used simplified methods of radiant heat transfer analysis is reported in Reference 48. A study directed at improving the understanding and prediction of orbiting spacecraft thermal performance is presented in References 46 and 49. A method presented in Reference 51 provides a means of evaluating the uncertainties associated with thermal radiation exchange. For an excellent status review (as of 1969) on radiation exchange between surfaces and in enclosures, the reader should consult Reference 52.

The above discussion merely serves to indicate that considerable care must be given when nodalizing a physical system and that the numerical evaluation of the finite difference equations must be considered from the total temperature error context. This means that user attention to a given numerical solution must be placed in a proper perspective.

5.1.2 Lumped-Parameter Equations

Using the network building block as expressed by equation (5.1-3) the lumped parameter system is identified as a set of ordinary non-linear differential equations by taking a heat balance as an ith node,

$$\frac{dT_i}{dt} = \frac{1}{C_i}\left[q_i + \sum_{j=1}^{p} a_{ij}\,(T_j - T_i) + \sum_{j=1}^{p} \sigma b_{ij}(T_j^4 - T_i^4)\right] \qquad (5.1\text{-}6)$$

$i = 1,2,\ldots,N$ (number of variable temperatures

T_j = constant, $N < j \leq p$

where, C_i = the ith nodal capacity which may be a function of temperature

q_i = the heat into node i and may be a function of time and temperature (impressed)

a_{ij} = the conduction coefficient between nodes i and j; it may be a function of time and temperature

b_{ij} = the radiation coefficient between nodes i and j; it may be a function of time and temperature

σ = Stefan-Boltzmann constant

Coefficients a_{ij} and b_{ij} are SINDA input quantities with the temperature factor of equation (5.1-6) calculated internally by the program. Both a_{ij} and b_{ij} may be variables. Conductance updating is a subject for discussion in a later paragraph. The user requirement to input the coefficients, a_{ij} and b_{ij}, provides considerable program flexibility, but at the same time user generation of these input quantities presents, in some instances, rather difficult engineering judgment decisions.

Radiation coefficient b_{ij} is, in essence, a radiation interchange factor, $\mathcal{F}_{ij}$,[53-55] (also known as script F) between nodes i and j. Generation of this quantity analytically can be quite difficult and inaccurate. A number of methods and computer programs (see, for example, Reference 56) are available for evaluating the shape factors which represent an important part of determining script F. A direct generation of script F is normally through the use of the Monte Carlo technique,[49] but a recent development utilizes a matrix formulation for determining the script F in an enclosure containing surfaces with arbitrary emission and reflection characteristics.[57,58] An experimental technique is reported in Reference 59.

5.2 Basic Finite Difference Formulations

The various numerical solution techniques differ in the finite difference formulations for the time-derivative (refer to equation 5.1-6); since the thermal equation is of the parabolic type, the transient heat transfer problems are of the initial value type. This means that at some time point, $t = n\Delta t$ (n is the number of time steps, Δt) all values of T_i are known. Thus,

$$T_{i,n+1} = T_{i,n} + \left[\frac{dT_{i,n}}{dt}\right]_{t_{n,n+1}} \Delta t \qquad (5.2\text{-}1)$$

$$i = 1,2,\ldots,N$$

where $t_{n,n+1}$ represents the time interval between $t = n\Delta t$ and $t = (n+1)\Delta t$

It is apparent that the selection of the proper value to $(dT_{i,n}/dt)t_{n,n+1}$ cannot be explicit and its selection identifies one numerical method from another. Although many finite difference formulations of the parabolic differential equation are available, two general classifications are commonly denoted as explicit or implicit. These numerical methods are well-documented in literature; the reader should refer to Reference 12 for a comprehensive discussion on various finite difference approximations. Explicit methods also discussed in References 14, 17, 19 and 20, among others, are step-by-step in time and equations.

Explicit methods include:

(1) Forward-difference explicit approximation[12,14]

This is an Euler method that computes temperatures in a step-by-step fashion. The requirement of stability places an upper limit on the time increment. SINDA subroutines CNFRWD, CNFRDL and CNFAST fall within this category. CNFAST is a modified version of CNFRWD which allows the user to specify the minimum time step to be taken. Refer to Sections 6.3.1 and 6.3.2 for details.

(2) Dufort-Frankel approximation[9, 12,17]

The Dufort and Frankel finite difference formulation is a three level formula that appears to be unconditionally stable. SINDA subroutine CNDUFU uses the Dufort-Frankel finite difference algorithm (refer to Section 6.3.4 for a detailed discussion).

(3) Exponential approximation [1, 17]

The exponential approximation is found by integrating the heat balance equations after making linear and constant coefficient assumptions. This method is unconditionally stable for linear systems but may be unstable for some types of nonlinear problems. SINDA subroutine CNEXPN employs this method and is discussed at length in Section 6.3.3.

(4) Alternating direction approximations[17]

This technique employs two formulations, one on odd time levels and the other on even time levels and is unconditionally stable.

The implicit finite difference formulations require a simultaneous computational procedure. In addition to Reference 12, implicit methods are also discussed in References 8, 10, 17, and 20, among others. Implicit methods include:

(1) Backward difference implicit approximation[12]

The backward difference weights only the flux terms at $t = (n+1)\Delta t$. As a result, the method is stable for all values of Δt. SINDA subroutine CNBACK employs this method and is detailed in Section 6.4.1.

(2) Crank-Nicolson approximation[8]

The Crank-Nicolson method uses the arithmetic average of the heat flux at the two time levels, $t = n\Delta t$ and

$t = (n+1)\Delta t$. The method is unconditionally stable. SINDA CNFWBK uses this method and is discussed in Section 6.4.2.

Steady state analysis also requires an implicit method of solution. SINDA steady state subroutines are called CINDSS, CINDSL and CINDSM which are detailed in Sections 6.5.1, 6.5.2, and 6.5.3.

5.2.1 Forward Finite Difference Explicit Method

By replacing the first derivative of temperature with respect to time, dT/dt, with the forward first difference quotient, equation (5.1-6) becomes,

$$C_i \frac{(T_{i,n+1} - T_{i,n})}{\Delta t} = q_i - \sum_{j=1}^{p} a_{ij} (T_{j,n} - T_{i,n}) + \sum_{j=1}^{p} \sigma b_{ij} (T_{j,n}^4 - T_{i,n}^4) \quad (5.2\text{-}1)$$

$$t = n\Delta t$$

$$i = 1,2,\ldots,N$$

$$T_{j,n} = \text{constant}, \; N < j \leq p$$

where, the second subscript on T represents the time level such that

$$T_{i,n} = T_i \,(n\Delta t)$$

Equation (5.1-6) is represented in the form expressed by equation (5.1-3) by letting,

$$G_{ij} = a_{ij} + \sigma b_{ij} \,(T_i^2 + T_j^2)(T_i + T_j) \quad (5.2\text{-}2)$$

It is interesting to note that the finite difference form of (5.1-6) (and thus 5.2-1) represents a second central-difference quotient of $\nabla^2 T$ (refer to (5.1-1).

The computational procedure for the forward difference formulation is rather straightforward since only a single unknown temperature at each time step, $T = n\Delta t$ for each equation is present. Note that the averaging of dT_i/dt) assigns a weighting factor to the heat flux terms only (terms on the right side of equation (5.1-6) at $t = n\Delta t$). Along with the computational simplicity, however, is the stability constraint which places an upper limit on the time increment, Δt, that can be used in the numerical procedure. The stability criterion for the explicit finite difference method is (for the most limiting node),[12, 14]

$$\Delta t < C_i / \sum_{j=1}^{p} G_{ij} \qquad (5.2\text{-}3)$$

$$i = 1,2,\ldots,N$$

A modified stability criterion that allows for a larger time step which results in a conditionally stable temperature for the most limiting node is reported in Reference 23. Since the stability criterion will govern the maximum time step that can be used, it is thus particularly important that a user gives some attention to those factors that compose the condition of stability when nodalizing a physical system.

In the discussion presented so far, arithmetic nodes (nodes with no heat capacity) have not been mentioned. Normally, the computational procedure treats arithmetic nodes separately from the diffusion nodes; arithmetic-node temperatures are solved implicitly. Detailed discussion on the general procedure will be presented in a later paragraph as well as in Section 6 which discusses the various SINDA numerical solution routines.

5.2.2 Implicit Finite Difference Method

The implicit difference equations can be constructed for heat transfer problems in many ways (see, for example, References 12 and 20).

Replacement of equation (5.1-6) with the backward time difference yields,

$$C_i \frac{(T_{i,n+1} - T_{i,n})}{\Delta t} = q_i + \sum_{j=1}^{p} a_{ij} (T_{j,n+1} - T_{i,n+1}) + \sum_{j=1}^{p} \sigma b_{ij} (T^4_{j,n+1} - T^4_{i,n+1}) \qquad (5.2\text{-}4)$$

$$i = 1,2,\ldots,N$$

$$T_{j,n+1} = \text{constant}, \; N < j \leq p$$

$$T_{i,n} = T_i (n\Delta t)$$

The computational procedure for the backward difference formulation must necessarily be re-iterative because of the need to solve a set of simultaneous non-linear equations.

In view of the importance of iteration techniques (such as method of successive approximation), it may be of interest to formulate equation (5.2-4) into an interative form. If we let $C_i/\Delta t \equiv \overline{C}_i$, use equation (5.2-2) in equation (5.2-4) and solve the resultant expression for $T_{i,n+1}$, this yields the recurrent equation for a given time increment, Δt, and time-step, n,

$$T_{i,k+1} = \frac{\overline{C}_{i,k}\, T_{i,k} + \sum_{j=1}^{p} G_{ij,k}\, T_{j,k} + q_{i,k}}{\overline{C}_{i,k} + \sum_{j=1}^{p} G_{ij,k}} \qquad (5.2\text{-}5)$$

$$i = 1,2,\ldots,N$$

where, $\overline{C}_{i,k} = C_{i,k}/\Delta t$

$$G_{ij,k} = a_{ij,k} + \sigma b_{ij,k}\,(T^2_{j,k} + T^2_{i,k})(T_{j,k} + T_{i,k}) \qquad (5.2\text{-}6)$$

$T_{j,k}$ = constant, $N < j \leq p$

k = kth iteration (note that $\overline{C}_{i,k}$, $q_{i,k}$, a_{ij} and b_{ij} are shown to be updated every iteration; SINDA routines update these quantities once each time-step)

The iterative pattern is initiated by assuming "old" temperatures ($T_{i,k}$ and $T_{j,k}$) on the right side of equation (5.2-5) to evaluate a "new" set of temperatures ($T_{i,k+1}$) on the left side of the equation (5.2-5); this single set of calculations represents an iteration. By replacing all of the "old" temperatures ($T_{i,k}$) on the right side of equation (5.2-5) with the just calculated "new" set of temperatures ($T_{i,k+1}$), a second iteration can be made. The iteration procedure is continued until a termination criterion such as the number of iterations or the maximum absolute difference between $T_{i,k}$ and $T_{i,k+1}$ is less than some prespecified value has been satisfied. It should be noted that G_{ij}, C_i and q_i are shown to be updated every iteration. This iterative process is termed "block" iteration since the "old" temperatures on the right side are replaced in a "block" (a set of temperatures) fashion with the "new" temperatures.

Another iterative technique is to utilize on the right side of equation (5.2-5) each "new" temperature as soon as it is calculated. This iterative method is termed "successive point" iteration and appears to yield solutions about 25% faster than the "block" iteration method.

Equation (5.2-5) can be expressed in a "successive point" form as follows:

$$T_{i,k+1} = \frac{\bar{C}_{i,k}\, T_{i,k} + \sum_{j=1}^{i} G_{ij,k}\, T_{j,k+1} + \sum_{j=i+1}^{p} G_{ij,k}\, T_{j,k} + q_{i,k}}{\bar{C}_i + \sum_{j=1}^{p} G_{ij,k}} \qquad (5.2\text{-}7)$$

where, $G_{ij,k} = a_{ij,k} + \sigma b_{ij,k}\, (T^2_{j,\ell} + T^2_{i,k})(T_{j,\ell} + T_{i,k})$

$(\ell = k \text{ if } j \geq i \text{ and } \ell = k+1 \text{ if } j < i)$

$T_{j,k}$ = constant, $N < j \leq p$

k = kth iteration (note that $\bar{C}_{i,k}$ $q_{i,k}$, a_{ij}, and b_{ij} are shown to be updated every iteration)

The iterative method as used in SINDA follows a fixed, predetermined sequence of operations in contrast with a relaxation procedure which is also one of successive approximations but is not processed out in a predetermined sequence. The relaxation procedure seeks and operates on the node with the maximum temperature difference between the "old" and the "new." From a programming standpoint, the search operation requires as much computational time as the temperature calculation itself.

5.2.3 Steady State Method

Standard steady state equations follows directly from equation (5.2-5) for block iteration or from equation (5.2-7) for successive point iteration by letting $\bar{C}_i = 0$ in these equations. The comments made in Section 5.2.2 are equally applicable here.

5.2.4 Some Comments

The finite difference expressions presented in this Section 5.2 represent standard formulations and thus do not show computational techniques and artifices which are used, some more or some less, in all programs. SINDA numerical solution routines contain many computational features (many original with J. D. Gaski) which enhance problem solution. The various computational aspects of the numerical solution methods as used in the SINDA routines are discussed in rather lengthy detail in Section 6.

6. SINDA NUMERICAL SOLUTION ROUTINES

6.1 Objective and Presentation Format

SINDA has available to the user a number of numerical solution routines which employ various numerical methods. A brief description of these routines with the required SINDA input quantities and format are contained in the SINDA users manual;[3,4] a general review of numerical methods was presented in Section 5. Unfortunately, the brief description is not sufficient for a casual SINDA user to make a selection decision from among several routines that are available and for a serious user to fully understand the computational procedure as well as to understand the role of the various control constants that are employed in each routine.

It is the intent of this section to fill wherever possible and practical the description void that presently exists with the numerical solution routines by detailing the characterisitcs of each. It is not the intent here to provide sufficient detailed information for a user to make modifications and/or additions to the existing subprograms, but rather to provide information that will aid the user in assessing the various numerical solution routines and in evaluating the numerical results.

Each of the numerical solution routines is detailed from a theoretical as well as from a computational standpoint. Control constants and their role are described and indicated in a step-by-step verbal flow computational procedure. Details of many of the numerous computational checks have purposely been omitted because of the complex interactions. Minute details of each routine can be obtained only from the individual computer listings; a computer listing of each of the SINDA numerical solution routines is presented in Appendices A, B and C. General computational procedure and features that apply to most, if not all of the SINDA numerical solution routines are assembled in a single section (6.2) in order to eliminate undue repetition. The description of each routine is heavily dependent upon, and coupled to, the general description of Section 6.2. The routines have been categorized as steady state or transient with the latter subcategorized as explicit or implicit in order to allow for an orderly presentation as well as to simplify future additions.

6.2 General Computational Procedure and Features

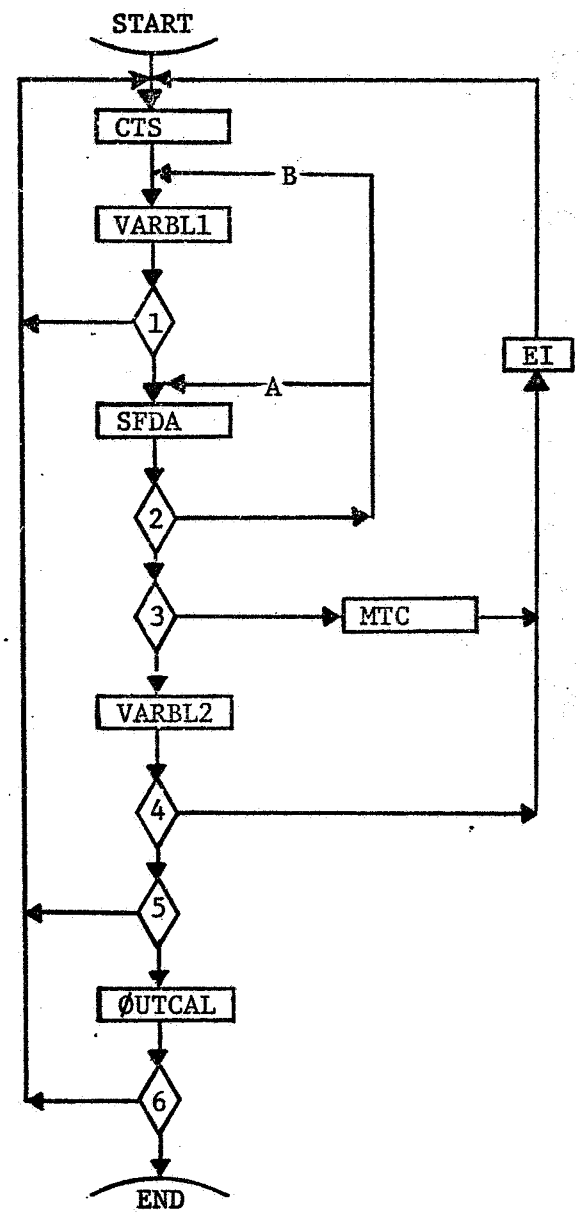
Each of the SINDA numerical solution routines employs a particular finite difference approximation of the lumped parameter heat balance equations. In spite of the uniqueness of each routine, portions of the computational procedure used in each are similar. Also, many of the routines have identical features such as the acceleration of convergence and the use of control constants. As a result, it is convenient to place in this section repetitious material. In some instances material presented here is repeated in the discussion of a particular numerical solution routine.

6.2.1 Order of Computation

It was reported in Section 3.5 that the order of computation depends on the sequence of subroutine calls placed in the EXECUTIØN block by the program user. No other operations block is performed unless called upon by the user either directly by name or indirectly from subroutines which internally call upon them. Numerical solution subroutines internally call upon operations blocks VARIABLES 1, VARIABLES 2, and ØUTPUT CALLS. The internal order of computation for these numerical solution routines is similar with the primary difference between one routine and another being the finite difference approximation employed in a particular routine. A flow diagram indicating the general order of computation for the numerical solution routines is depicted in Figure 6.2-1.

6.2.1.1 Finite Difference Algorithm

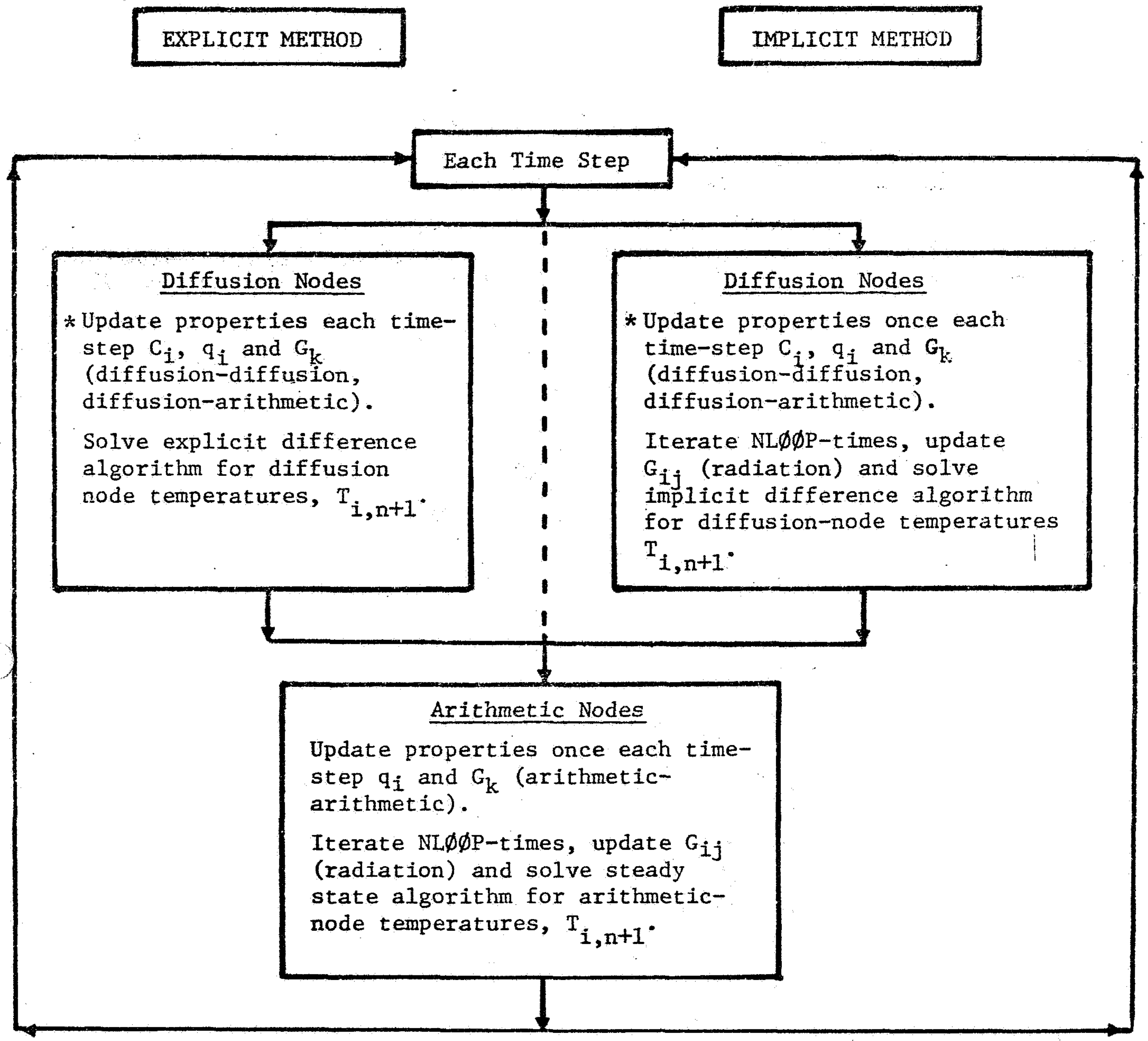
Although each of the SINDA numerical solution routines employs a particular finite difference approximation which is detailed for each numerical solution routine, the computational pattern is similar. Within the box depicted as $\boxed{\text{SFDA}}$ in Figure 6.2-1, solution of the finite difference algorithm occurs. The computational sequence for transient solutions follows one of two patterns: (1) one for explicit finite difference methods; and (2) one for implicit finite difference methods; steady state solutions follow closely the implicit pattern. Both numerical flow pictures are depicted in Figure 6.2-2; details within the flow pictures are different for each routine and are described separately under the individual SINDA numerical solution routines (refer to Sections 6.3 - 6.5).



OPERATION	DESCRIPTION
CTS	Calculate time step
VARBL1	Variables 1 operation
SFDA	Solve finite difference algorithm
VARBL2	Variables 2 operations
ØUTCAL	Output calls operations
MTC	Modify time control
EI	Erase iteration

Check	Reverse direction if
1	Backup nonzero
2	Relaxation criteria not not met
3	Time or temp change too large
4	Backup nonzero
5	Not time to print
6	Problem stop time not reached
A	Implicit routines, iteration loops
B	Implicit routines, once per time-step loop

Figure (6.2-1) General Order of Computation for Numerical Solution Routines



* For CINDA -3G users, it should be noted that the updating of properties occurs within the numerical solution routine <u>after</u> VARIABLES 1 call. CINDA-3G evaluates the variable properties <u>before</u> VARIABLES 1 call.

Figure 6.2-2. Numerical Computational Pattern for Explicit and Finite Difference Algorithms

6.2.1.2 Updating of Optionally Specified Properties

Optionally specified properties are defined here as those items which result in pointers being set in the second pseudo compute sequence (refer to Section 3.3.4). The term optional refers to mnemonic options that are available for different types of variable properties.[3,4] The properties are updated in all SINDA numerical solution routines the same way. This definition is used here in lieu of stating that optionally specified properties are time and/or temperature varying properties since source data may be specified to be constant. The pointers are set by one or more of the following user input quantities:

(1) All capacitances, C_i, specified as f(T) or f(t,T) in NODE DATA BLOCK:

(2) All data, q_i, entered in the SOURCE DATA BLOCK:

(3) All coefficients, G_k, specified as f(T) or f(t,T) in CONDUCTOR DATA BLOCK. It should be noted here that the term coefficient as used here requires amplification. The conductance, G_{ij}, may be for conduction or for radiation; that is,

$$G_{ij} \equiv G_k = a_{ij} \quad \text{(for conduction conductance)}$$

$$G_{ij} = \sigma b_{ij} \; (T_i^2 + T_j^2)(T_i + T_j) \quad \text{(for radiation conductance)}$$

$$= G_k \; (T_i^2 + T_j^2)(T_i + T_j)$$

Thus, note that the calculated conduction conductance G_{ij} is identical to the updated G_k, whereas for the calculated radiation conductance only σb_{ij} is equivalent to the updated G_k.

The type of optional properties is identified by the integer stored in the first six bits of the second pseudo compute sequence which indicates to the program which option is in effect. Optional property types are listed and described for the three categories of input quantities in Table 6.2-1 for capacitance, Table 6.2-2 for impressed source, and Table 6.2-3 for coefficients with the definition of symbols listed in Table 6.2-4.

6.2.2 Operations Blocks

In a previous paragraph, it was mentioned that the sequence of subroutine calls placed in the EXECUTION block by the user determines the

TABLE 6.2-1 OPTIONALLY SPECIFIED CAPACITANCE EXPRESSIONS

Option	Type	Expression
SIV	1	$C_i = F(A^i:T_i)$
DIV	2	$C_i = F1(A_1^i:T_i) + F2(A_2^i:T_i)$
DIV	3	$C_i = F1(L) + F2(A^i:T_i)$
DIV	4	$C_i = F1(A^i:T_i) + F2(L)$
SPV	5	$C_i = F(A^p:T_i)$
DPV	6	$C_i = F1(A_1^p:T_i) + F2(A_2^p:T_i)$
DPV	7	$C_i = F1(L) + F2(A^p:T_i)$
DPV	8	$C_i = F1(A^p:T_i) + F2(L)$
BIV	9	$C_i = F(A^b:T_i, t_m)$

Notation: Refer to Table 6.2-4.

TABLE 6.2-2 OPTIONALLY SPECIFIED IMPRESSED SOURCE EXPRESSIONS

Option	Type	Expression
blank	1	$q_i = q_i + F$
SIV	2	$q_i = q_i + F(A^i:T_i)$
SIT	3	$q_i = q_i + F(A^i:t_m)$
DIT	4	$q_i = q_i + F1(A_1^i:t_m) + F2(A_2^i:t_m)$
DIT	5	$q_i = q_i + F1(L) + F2(A^i:t_m)$
DIT	6	$q_i = q_i + F1(A^i:t_m) + F2(L)$
DTV	7	$q_i = q_i + F1(A_1^i:t_m) + F2(A_2^i:T_i)$
DTV	8	$q_i = q_i + F1(L) + F2(A^i:T_i)$
DTV	9	$q_i = q_i + F1(A^i:t_m) + F2(L)$

Notation: Refer to Table 6.2-4.

Table 6.2-3. Optionally Specified Coefficient Expressions for Conduction and Radiation

Mnemonic Options	Type	Expression
SIV	1	$G_k = F(A^i:T_m)$
SIV	2	$G_k = F(A^i:T_i)$
DIV (conduction)	3	$G_k = 1.0/[1.0/F1(A_1^i:T_i) + 1.0/F2(A_2^i:T_j)]$
(radiation)		$G_k = [F1(A_1^i:T_i)][F2(A_2^i:T_j)]$
DIV (conduction)	4	$G_k = 1.0/[1.0/F1(L) + 1.0/F2(A^i:T_j)]$
(radiation)		$G_k = [F1(L)][F2(A^i:T_j)]$
DIV (conduction)	5	$G_k = 1.0/[1.0/F1(A^i:T_i) + 1.0/F2(L)]$
(radiation)		$G_k = [F1(A^i:T_i)][F2(L)]$
SPV	6	$G_k = F(A^P:T_m)$
SPV	7	$G_k = F(A^P:T_i)$
DPV (conduction)	8	$G_k = 1.0/[1.0/F1(A_1^P:T_i) + 1.0/F2(A_2^P:T_j)]$
(radiation)		$G_k = [F1(A_1^P:T_i)][F2(A_2^P:T_j)]$
DPV (conduction)	9	$G_k = 1.0/[1.0/F1(L) + 1.0/F2(A^P:T_j)]$
(radiation)		$G_k = [F1(L)][F2(A^P:T_j)]$
DPV (conduction)	10	$G_k = 1.0/[1.0/F1(A^P:T_i) + 1.0/F2(L)]$
(radiation)		$G_k = [F1(A^P:T_i)][F2(L)]$
BIV	11	$G_k = F(A^b:T_m, t_m)$
SIV	12	$G_k = F(A^i:T_j)$
SPV	13	$G_k = F(A^P:T_j)$

Notation: Refer to Table 6.2-4; note $G_k \equiv \sigma b_{ij}$ (for radiation)
$\equiv a_{ij}$ (for conduction)

Table 6.2-4. Definition of Symbols for Tables 6.2-1 -- 6.2-3

Symbols	Definition
C_i	Capacitance of ith node.
F, F1, F2	Multiplying factors, either user constants or literal
$G_k(=a_{ij})$	Conduction coefficient.
$G_k(=\sigma b_{ij})$	Radiation coefficient.
L	A literal multiplying factor.
q_i	Heat load into the ith node. (impressed)
Δt	Time-step
t_m	Mean time, (TIMEØ + TIMEN)/2.0
T_m	Mean temperature, $(T_i + T_j)/2.0$
$(A^i:t_m)$	Interpolated value of array A using t_m as the independent variable.
$(A^i:T_i)$	Interpolated value of array A using T_i as the independent variable.
$(A^b:T_i, t_m)$	Interpolated value of the bivariate array A using T_i and t_m as independent variables.
$(A^b:T_m, t_m)$	Interpolated value of the bivariate array A using T_m and t_m as independent variables.

Mnemonic Options

BIV	Bivariate Interpolation Variable
DIT	Double Interpolation with Time as variable
DIV	Double Interpolation Variable
DPV	Double Polynomial Variable
DTV	Double interpolation with Time and Temperature as Variables
SIT	Single Interpolation with Time as variable
SIV	Single Interpolation Variable
SPV	Single Polynomial Variable

Subscripts

i	Indicates the ith node.
j	Indicates the jth node.
2	Indicates two (array).

order of computation. Operations blocks number four, EXECUTION, VARIABLES 1, VARIABLES 2, and OUTPUT CALLS. These operations blocks are described in the SINDA Users Manual[3,4] but their role insofar as the numerical solution routines are concerned may be of particular interest.

6.2.2.1 EXECUTION Operations Block

The EXECUTION operations block provides the user considerable flexibility in the use of SINDA calls and FØRTRAN operations. Combinations of SINDA calls and FØRTRAN operations are innumerable since the user is actually programming. Now all instructions contained in the VARIABLES 1, VARIABLES 2 and ØUTPUT CALLS are performed each iteration or on the output call interval. Thus, if an operation being performed in VARIABLES 1 utilizes and generates non-changing constants, the operation should be placed in the EXECUTION block (prior to the numerical solution call) so that it will be performed only once and thus eliminate repetitious non-changing calculations. Operations of this type are conveniently performed in the EXECUTION operations block. Note, however that a constant impressed source should be placed in the optional source data block for SINDA and VARIABLES 1 block for CINDA-3G.

6.2.2.2 VARIABLES 1 Operations Block

The VARIABLES 1 operations block provides the user with a means of specifying at a point in the computational sequence, as shown in Figure 6.2-1, the evaluation of nonlinear network elements, coefficients and boundary values not considered by the various mnemonic codes utilized for node, conductor and source data. It is seen from Figure 6.2-1 that VARIABLES 1 operations occur just prior to entering the numerical solution phase in order to define the network completely.

6.2.2.3 VARIABLES 2 Operations Block

VARIABLES 2 operations are post-solution operations in contrast to the VARIABLES 1 operations which are pre-solution operations as shown in Figure 6.2-1. VARIABLES 2 provides the user with a means to examine the characteristics of the numerical solution and make corrections. For example, the heat flow from one node to another can be evaluated or a temperature(s) determined without material phase change can be corrected to account for the phase change by using the VARIABLES 2 operations block.

6.2.2.4 OUTPUT CALL Operations Block

The OUTPUT CALL operations block provides the user with a means of calling any desired subroutine with the operation performed on the output interval. In addition to various subroutines for printing output, several plotting subroutines are available.[3,4]

6.2.3 Control Constants

Control constants number forty-nine and have alphanumeric names. Control constant values are communicated through program common to specific subroutines which require them. Whenever possible, control constant values not specified are set internally to acceptable values. If a required control constant value is not specified, an appropriate error message is printed and the program terminated. Each of the SINDA numerical solution routines employs a number of control constants which fall under the categories as: (1) user specified; (2) optionally user specified; (3) internally set by program; and (4) dummy. These control constants are listed alphabetically with a brief description of each in Section 6.2.3.1 followed by a detailed description of user specified control constants in Section 6.2.3.2; nominal values of these control constants that must be specified or are optionally specified for each SINDA numerical solution routine are indicated in Table 6.2-5. Specification of these control constants is detailed under the discussion of each SINDA numerical solution routine.

6.2.3.1 Alphabetical Listing and Brief Description of Control Constants

ARLXCA (control constant 19)

Maximum arithmetic node relaxation temperature change allowed between iterations; this check occurs after each iteration. Specification is required for the implicit and steady state routines (except CINDSM) and if not specified an error message is printed if the number of arithmetic nodes is greater than zero. Specification is not required for explicit routines and if not specified, ARLXCA is set to 1.E+8.

ARLXCC (control constant 30)

Maximum arithmetic node relaxation temperature change calculated by program; ARLXCC $\leq$ ARLXCA check is made.

Table 6.2-5. Characteristics of User Specified Control Constants

Control Name	Constant Number		CINDSS	CINDSL	CINDSM	CNFRWD	CNFRDL	CNFAST	CNEXPN	CNDUFR	CNQUIK	CNBACK	CNFWBK	CNVARB
		SINDA Numerical Solution Routines	Steady State			Explicit						Implicit		
ARLXCA	19	Allowable arithmetic node relaxation temperature change	**	**	-	1.E+8	1.E+8	1.E+8	1.E+8	1.E+8	1.E+8	**	**	**
ATMPCA	11	Allowable arithmetic node temperature change		-	-	"	"	-	"	"	"	1.E+8	1.E+8	1.E+8
BACKUP	12	Backup switch	-	-	-	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
BALENG	33	System energy balance	-	-	*	-	-	-	-	-	-	-	-	-
CSGFAC	4	Time-step factor for explicit routines	-	-	-	1.0	1.0	-	1.0	1.0	1.0	-	-	-
DAMPA	9	Arithmetic-node damping factor	1.0	1.0	-	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
DAMPD	10	Diffusion-node damping factor	1.0	1.0	1.0	-	-	-	-	-	-	1.0	1.0	1.0
DRLXCA	26	Allowable diffusion-node relaxation temperature change	***	***	***	-	-	-	-	-	-	***	***	***
DTIMEH	8	Maximum time-step allowed	-	-	-	1.E+8	1.E+8	1.E+8	1.E+8	1.E+8	1.E+8	1.E+8	1.E+8	1.E+8
DTIMEI	22	Specified time-step for implicit routines	-	-	-	-	-	-	-	-	-	*	*	*
DTIMEL	21	Minimum time-step allowed	-	-	-	0.0	0.0	*	0.0	0.0	0.0	-	-	-
DTMPCA	6	Allowable diffusion-node temperature change	-	-	-	1.E+8	1.E+8	-	1.E+8	1.E+8	1.E+8	1.E+8	1.E+8	1.E+8
LAXFAC	49	Number of iterations for linearized system	-	-	*	-	-	-	-	-	-	-	-	-
NLØØP	5	Number of desired iteration loops	*	*	*	1	1	1	1	1	1	*	*	*
ØUTPUT	18	Time interval for activating ØUTPUT CALLS	-	-	-	*	*	*	*	*	*	*	*	*
TIMEND	3	Problem stop-time	-	-	-	****	****	****	****	****	****	****	****	****
TIMEØ	13	Old time of problem start time	-	-	-	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
PCS		Pseudo Compute Sequence	SPCS	LPCS	LPCS	SPCS	LPCS	SPCS	SPCS	SPCS	SPCS	LPCS	LPCS	LPCS
Dynamic storage requirements			NND	2(NND) +2(NNA)	3(NND) +3(NNA) +NGT	NND +NNA	NND +NNA	NND	NND +NNA	2(NND) +NNA	2(NND) +NNA	3(NND) +NNA +NNB	3(NND) +NNA +NNB	3(NND) +NNA +NNB

Notation:

* Indicates "run" termination with error message printout if not specified.

** " " " " " " " " " " when NNA > 0.

*** " " " " " " " " " " NND > 0.

**** " " " " " " " " TIMEND ≤ TIMEØ for implicit routines, no error message for explicit routines.

NNA is the number of arithmetic-nodes.

NND " " " " diffusion " .

NNB " " " " boundary " .

NGT " " total number of conductors.

\- Indicates that the control constant is not applicable.

ATMPCA (control constant 11)

Maximum arithmetic node temperature change allowed between time steps for transient routines; the check occurs after the specified number of iterations. If not specified or if specified to be $\leq$ 0.0, ATMPCA is set to 1.E+8.

ATMPCC (control constant 15)

Maximum arithmetic temperature change calculated by program; ARMPCC $\leq$ ATMPCA check is made.

BACKUP (control constant 12)

Backup switch that is checked after VARIABLES 1 and VARIABLES 2 calls. Initialized at zero. If specified to be non-zero, the completed time step is erased and repeated.

BALENG (control constant 33)

A user specified system energy balance to be maintained; this control constant is presently used only in CINDSM. If not specified, an error message will be printed.

CSGFAC (control constant 4)

Time step factor for explicit routines except CNFAST. If not specified or if specified to be less than 1.0, CSGFAC is set internally to 1.0.

CSGMAX (control constant 23)

Maximum value of $C_i/\Sigma\ G_{ij}$; this value aids in the checkout of the thermal network and is calculated only by the output subroutines, CSGDMP and RCDUMP.

CSGMIN (control constant 17)

Minimum value of $C_i/\Sigma\ G_{ij}$; this value is used to limit the computational time step for explicit methods of solution. If CSGMIN is calculated to be $\leq$ 0.0, an error message is printed.

CSGRAL (control constant 24)

Allowable range between CSGMIN and CSGMAX: this control constant is not presently used but is included for future considerations.

DAMPA (control constant 9)

Arithmetic-node damping factor for all numerical solution routines; if not specified, or if specified to be ≤ 0.0, DAMPA is set to 1.0. (Refer to equation 6.2-6.)

DAMPD (control constant 10)

Diffusion node damping factor for implicit and steady state routines; if not specified or is specified to be ≤ 0.0, DAMPD is set to 1.0. (Refer to equation 6.2-20.)

DRLXCA (control constant 26)

Maximum diffusion node relaxation temperature change allowed between iterations for implicit and steady state routines; this check occurs after each iteration. If not specified an error message will be printed when the number of diffusion nodes is greater than zero.

DRLXCC (control constant 27)

Maximum diffusion node relaxation temperature change calculated by the program; DRLXCA $\leq$ DRLXCC check is made.

DTIMEH (control constant 8)

Maximum time step allowed; applies to transient routines. If not specified or if specified to be ≤ 0.0, DTIMEH is set to 1.E-8.

DTIMEI (control constant 22)

Specified time step for implicit solutions; if not specified, an error message will be printed and the "run" terminated.

DTIMEL (control constant 21)

Minimum time step allowed for explicit routines. If not specified for CNFAST, an error message will be printed and the "run" terminated. If DTIMEU is less than DTIMEL the routines will terminate with an error message, except for CNFAST which will do a steady state solution on the offending node. For all routines DTIMEL is initially set at 0.0 internally.

DTIMEU (control constant 2)

Contains time step used in computational procedure.

DTMPCA (control constant 6)

Maximum diffusion node temperature change allowed between time steps for transient routines. If not specified or if specified to be $\leq$ 0.0, DTMPCA is set to 1.E+8.

DTMPCC (control constant 15)

Maximum diffusion node temperature change calculated by program; DTMPCA $\leq$ DTMPCC check is made.

ENGBAL (control constant 32)

Calculated energy balance of the system; presently used only in CINDSM.

LAXFAC (control constant 49)

Specified number of iterations to be performed on a linearized system with no updating of elements during a set of LAXFAC iterations for CINDSM only; if not specified, an error message is printed and the "run" terminated.

LINECT (control constant 28)

A line counter location for program output (integer).

LØØPCT (control constant 20)

Contains number of iterations performed (integer).

NØCØPY (control constant 34)

Contains the no copy switch for matrix users.

NLØØP (control constant 5)

Number of specified iteration loops. Must be specified for the steady state and implicit routines; if not specified, an error message is printed and the "run" is terminated. Optional specification for solution of the arithmetic nodes in the explicit routines; if not specified, NLØØP is set to integer 1.

ØPEITR (control constant 7)

Output each iteration if ØPEITR is specified to be non-zero; if not specified, ØPEITR is set at zero. May be switched on and off during a run.

ØUTPUT (control constant 18)

Time interval for activating ØUTPUT CALLS of transient routines; if not specified, error message is printed and the "run" terminated. May be addressed by user and modified during a run in VARIABLES 2. Can be used in steady state routines for a series of steady state solutions.

PAGECT (control constant 29)

A page counter location for program output (integer).

TIMEM (control constant 14)

Mean time for a computation interval; $\text{TIMEM} = \dfrac{\text{TIMEØ} + \text{TIMEN}}{2.0}$.

TIMEN (control constant 1)

New time at the end of the computational interval.
TIMEN = TIMEØ + DTIMEU.

TIMEND (control constant 3)

Problem stop time for transient analysis. Must be > TIMEØ for all routines; if not, an error message is printed and "run" terminated. May be addressed by the user and modified during a run.

TIMEØ (control constant 13)

Old time at the start of the computational interval. Also used as the problem start time and may be negative; if not specified, TIMEØ is set at zero.

ITEST, JTEST, KTEST, LTEST, MTEST (control constants 39, 40, 41, 42 and 43, respectively)

Contain dummy integer constants.

RTEST, STEST, TTEST, UTEST, VTEST (control constants 44, 45, 46, 47, and 48, respectively)

Contain dummy floating point constants.

(Control constant 31)

Problem type indicator, 0 = THERMAL SPCS, 1 = THERMAL LPCS, 2 = GENERAL.

(Control constant 35)

Contains relative node number of CSGMIN.

(Control constant 36)

Contains relative node number of DTMPCC.

(Control constant 37)

Contains relative node number of ARLXCC.

(Control constant 38)

Contains relative node number of ATMPCC.

6.2.3.2 User Specified and Optionally User Specified Control Constants

The availablity of control constants which must be specified or which can optionally be specified provides the user with considerable flexibility to alter the computational criteria and hence the calculated temperatures. On the other hand, this flexibility presents the user with the problem of imputting control constant values if the nominal values are not suitable. An attempt will be made here to provide some guidelines on control constant values based on rather limited data presently available, but it should be recognized that suitable values to be used are dependent on the problem to be solved and often a trade-off must be made between accuracy and computational time. This normally can be obtained only through the use of the numerical solution routines.

ARLXCA (Allowable Arithmetic Node Relaxation Temperature Change)

This control constant must be specified for the implicit routines if any arithmetic node is present and for the steady state routines except CINDSM. For the explicit solution routines, ARLXCA may be optionally specified; if not specified ARLXCA is set to 1.E+8. ARLXCA represents a maximum temperature change convergence criterion for the arithmetic nodes; ARLXCA is checked each iterative step. It is used in conjunction with control constant NLOØP. Satisfaction of either NLØØP or ARLXCA during any iterative step terminates the arithmetic node temperatures calculation for that time-step with computation proceeding on to the next one. Typically, an ARLXCA value is 0.01, but its value is dependent upon the magnitude of expected temperatures. The 0.01 value tries for 5th digit accuracy for temperatures in the hundreds. An ARLXCA value of 0.0001 would try for seventh digit accuracy. Since the computer will not yield 8 digit accuracy, an ARLXCA value < .0001 will always result in NLØØP iterations being performed.

ATMPCA (Allowable Arithmetic Node Temperature Change)

This control constant may be optionally specified by the user for the implicit routines and for the explicit routines except CNFAST. If not specified, ATMPCA is internally set at 1.E+8. ATMPCA represents an allowable arithmetic-node temperature change criterion between one time-step and another with the calculated temperature change stored in control constant ATMPCC. If the maximum arithmetic-node temperature change is greater than ATMPCA, the time-step, Δt, is shortened to,

$$\Delta t = .95 * \Delta t \ (ATMPCA/ATMPCC)$$

and the arithmetic-node and diffusion-node temperatures re-set to former values. The computational procedure is repeated with the smaller time-step. Specification of ATMPCA prevents a rapid temperature change between time-steps with the value to be specified dependent upon the problem. Thus, the user should estimate the number of time-steps and the range of the temperature to arrive at a reasonable value. For typical spacecraft-type thermal problems an ATMPCA of about 10°F is typical.

BACKUP (<u>Backup Switch</u>)

Control constant BACKUP provides the SINDA user with the means to utilize any thermal numerical solution subroutine as a predictor program. All of the numerical solution subroutines set control constant BACKUP to zero, just prior to the call on VARIABLES 2. Then immediately after the return from VARIABLES 2. a nonzero check on BACKUP is made. If BACKUP is nonzero, all temperature calculations for the just completed time-step are eliminated, the old temperatures (temperatures calculated at the previous time-step) are placed in the temperature locations and the control is routed to the start of the computational sequence.

It should be noted that the user must provide the necessary check and criterion in VARIABLES 2 if the iteration is to be repeated. Thus, if the iteration is to be repeated, BACKUP must be nonzero and a criterion that can be met in subsequent passes established. For example, the criterion may require the correction of a parameter used by the network solution. Further, if other calls in VARIABLES 2 are not to be performed FØRTRAN instructions must be generated to bypass these calls.

It should be noted that BACKUP is sometimes checked after VARIABLES 1. However, for the present this use should be ignored since BACKUP check after VARIABLES 1 is planned for future additions of special boundary calculation subroutines.

BALENG (<u>User Specified System Energy Balance</u>)

This control constant is presently used in the steady state routine CINDSM but not in the other SINDA numerical solution routines. BALENG must be specified, otherwise the "run" is terminated with an error message printout; the value of BALENG is a criterion that represents an acceptable net energy balance (energy in minus energy out) of the system in the calculation of steady state temperatures. A value for BALENG depends upon the magnitude of system energy under consideration. As a guideline 1/2% of the total energy into the system (including heat flow from the boundary) is a reasonable value.

CSGFAC (<u>Time Step Factor</u>)

This control constant may be optionally specified by the user for the explicit routines except CNFAST and it provides the user with some control on the compute time-step as indicated in Section 6.2.4. If CSGFAC is not specified or is specified to be less than one by the user, it is internally set at 1.0. For subroutines CNFRWD and CNFRDL which are conditionally stable CSGFAC is a divisor; a value of CSGFAC greater than one is used to obtain higher accuracy. For subroutines CNEXPN, CNDUFR and CNQUIK, which are unconditionally stable, CSGFAC is a multiplier (refer to page 6-24); a value of CSGFAC greater than one is used to decrease the computational time. A question may be raised, why a value of CSGFAC less than one is not allowed for CNEXPN, CNDUFR and CNQUIK? The reason for this is that it is more accurate to use CNFRWD (or CNFRDL) if a smaller time-step than the one associated with CSGFAC equal to one is desired.

DAMPA (Damping Factor for Arithmetic Nodes)

This control constant may be optionally specified for all of the SINDA numerical solution routines; if not specified, of if specified to be $\leq$ 0.0, DAMPA is set to 1.0. In the development of the finite difference expressions as reported in technical literature, little (if any) mention is made about the so-called damping factor. The damping factor does nothing more than to allow a certain fraction (1.0 - DAMPA) of the "old" temperature (temperature at the previous time-step or iteration) to be included as part of the temperature change for the current time-step or iteration. The value to be used is dependent upon the problem and to some extent upon the routine. Typically, a value of 0.6 is used but a value as small as 0.01 has been used with CINDSL for a thermal radiation-dominated problem. In general, a choice for DAMPA becomes a trial and error procedure. DAMPA is used only with arithmetic nodes (refer to equation 6.2-6).

DAMPD (Diffusion Node Damping Factor)

This control constant may be optionally specified for the implicit and steady state routines; if not specified or if specified to be $\leq$ 0.0, DAMPD is set to 1.0. DAMPD serves the same purpose for the diffusion nodes as DAMPA provides for the arithmetic nodes (refer to equation 6.2-21).

DRLXCA (Allowable Diffusion-Node Relaxation Temperature Change)

This control constant must be specified for the implicit routines and for the steady state routines except CINDSM. DRLXCA serves the same purpose for the diffusion-nodes as control constant ARLXCA does for the arithmetic nodes. Thus, the discussion on ARLXCA equally holds true for DRLXCA. It may be asked, why ARLXCA and DRLXCA? The reason for this is that it provides greater computational flexibility.

DTIMEH (Maximum Time-Step Allowed)

This control constant may be optionally specified for the explicit and the implicit routines. DTIMEH represents the maximum time-step allowed during the computational process. One use of DTIMEH is the prevention of a single large and a single small computational time-step during an output interval by specifying DTIMEH as a fraction of the output interval. If DTIMEH is not specified, DTIMEH is set to 1.0E+8.

DTIMEI (Specified Time-Step for Implicit Routines)

This control constant must be specified for the implicit routines and is not used by the other routines. If not specified, the "run" terminates with an error message printout. DTIMEI represents a specified time-step and is arbitrary, but the governing criterion should be minimum computational time with satisfactory temperature accuracy. This means that DTIMEI should be specified in conjunction with control constant NLØØP which represents the maximum number of computational iterations allowed during each time-step. Since each iterative calculation is essentially equivalent to a time-step calculation, DTIMEI should be normally greater than NLØØP*CSGMIN, where CSGMIN is the time-step used in the explicit routines. If savings in computational time cannot be met with the same accuracy by using the implicit routines, it is more reasonable to use the explicit routines.

DTIMEL (Minimum Time-Step Allowed)

This control constant must be specified for subroutine CNFAST and is optional for other explicit solution routines. If not specified for CNFAST, the "run" terminates with an error message printout. DTIMEL represents the minimum time-step allowed; for all the explicit routines except CNFAST, if the calculated time-step is less than DTIMEL, the "run" terminates with an error message printout. For subroutine CNFAST, if the calculated time-step of any node, as expressed by $C_i/\Sigma G_{ij}$ and stored in CSGMIN, is less than DTIMEL, the temperature of the nodes not satisfying DTIMEL are calculated using the steady state equations without computational iterations (refer to Section 6.3.3 for details on the CNFAST routine). The purpose of this control constant for CNFAST is to shorten the computational time; the danger in its use is that with a large DTIMEL a large number of diffusion nodes will receive the steady state equations without iterations. As a result, the temperature inaccuracies can be expected to be large.

DTMPCA (Allowable Diffusion Node Temperature Change)

This control constant may be optionally specified by the user for the implicit routines and for the explicit routines except CNFAST. DTMPCA represents a diffusion-node temperature change criterion between one time-step and another. If the maximum diffusion-node temperature change which is stored in DTMPCC is greater than DTMPCA, the time-step is shortened to,

$$\Delta t = .95 * \Delta t \text{ (DTMPCA/DTMPCC)}$$

and the diffusion-node and arithmetic-node temperatures re-set to former values. The computational procedure is repeated with the smaller time-step. DTMPCA serves the same purpose for the diffusion nodes as control constant DRLXCA provides the arithmetic nodes.

LAXFAC (Number of Iterations for Linearized Lumped Parameter System)

LAXFAC is used only in the steady state routine CINDSM and represents the number of iterations to be performed on a linear lumped parameter system with no updating of elements during a set of LAXFAC iterations. The system elements are re-evaluated for the new set of temperatures and in turn temperatures are recalculated for another set of LAXFAC iterations with a more severe relaxation criterion. The number of iterations will not exceed control constant NLØØP which represents the total number of iterations. NLØØP will not be met only if relaxation criteria are met during an iterative loop and between iterative loops and if the system energy balance as stored in BALENG is satisfied (refer to Section 6.5.3 for details).

NLØØP (Number of Iteration Loops)

This control constant must be specified for the implicit and the steady state routines; if not specified, the "run" terminates with an error message printout. NLØØP may be optionally specified for the explicit routines since it is used for the arithmetic nodes; if not specified, NLØØP is set to 1. The value of NLØØP to be used depends upon the problem to be solved. For a steady state problem it is not unusual to have NLØØP equal to several hundred, whereas for a transient problem the implicit routines NLØØP should be specified as discussed for control constant DTIMEI. In general, a trial and error procedure is required to arrive at a suitable value of NLØØP.

OUTPUT (Time Interval for Activating ØUTPUT CALLS)

This control constant must be specified for all numerical solution routines except steady state routines since the first time-step used is generally set to ØUTPUT. The input value is left to the judgment of the user. Normally, the output interval is gauged by the length of the run and the expected temperature response characteristics. As a "rule-of-thumb" the output interval lies between CSGMIN and CSGMAX, with ØUTPUT being several times larger than CSGMIN. The values of CSGMIN and CSGMAX can be obtained from the output subroutines CSGDMP and RCDUMP.[3,4] Subroutines CSGDMP and RCDUMP are designed to aid in the checkout of thermal problem data decks and should be used before making a transient computer run.

TIMEND (Problem Stop Time)

The use of this control constant is self-explanatory. For the subroutines as they are presently coded, TIMEND must be specified as larger than TIMEØ, otherwise an error message is printed and the "run" terminated. For the explicit routines, if TIMEND is not larger than TIMEØ a time-step of zero will result and the "TIME STEP TOO SMALL" error message will be printed. The implicit routines will print the error message, "TRANSIENT TIME NOT SPECIFIED." If a solution is to be terminated by the use of a criteria, but the run is not to be terminated, this can be accommodated by setting TIMEND=TIMEØ when the criteria is met.

TIMEØ ("Old" Time or Problem Start Time)

This control constant represents the "old" time or the problem start time for the transient routines. If not specified, TIMEØ is set to 0.0. An important consideration in the use of TIMEØ is that TIMEØ may be set to negative.

6.2.4 Time-Step Calculations

Each numerical solution routine requires the use of a time-step that depends upon many considerations, such as the output interval, the end of the problem time, the stability criterion for explicit routines, etc. In spite of the unique solution procedure of each of the numerical solution routines, the overall time-step calculation procedure for the transient routines is essentially identical. The numerous time-step checks, as well as the selection of the time-step, are indicated below (for definition of control constants refer to Section 6.2.3):

(1) Check that elapsed time, t, does not exceed problem end time.

If: TIMEØ + ØUTPUT > TIMEND

Set: ØUTPUT = TIMEND - TIMEØ

TIMEØ is the old time

ØUTPUT is the output time interval

TIMEND is the problem stop time

(2) Set initial time-step, Δt, which is stored in DTIMEU (control constant for time-step). The initial time step for the SINDA numerical routines is as follows:

Numerical Routines		Initial Time-Step
EXPLICIT	CNFRWD	ØUTPUT
EXPLICIT	CNFRDL	ØUTPUT
EXPLICIT	CNEXPN	ØUTPUT
EXPLICIT	CNDUFR	ØUTPUT
EXPLICIT	CNQUIK	ØUTPUT
EXPLICIT	CNFAST	DTIMEL (minimum time-step allowed)
IMPLICIT	CNBACK	DTIMEI (specified time-step)
IMPLICIT	CNFWBK	DTIMEI
IMPLICIT	CNVARB	DTIMEI

(3) Check Δt (stored in DTIMEU) against maximum allowable time-step.

If: DTIMEU > DTIMEH

Set: DTIMEU = DTIMEH

(4) Check sum of elapsed time since last printout, TSUM, and time-step, DTIMEU, against ØUTPUT.

If: TSUM + DTIMEU > ØUTPUT

Set: Δt = ØUTPUT - TSUM

If: TSUM + Δt < ØUTPUT

and if: TSUM + 2(Δt) > ØUTPUT

Set: Δt = 1/2 (OUTPUT - TSUM)

(5) Store

Set: DTIMEU = Δt

(6) Check DTIMEU against minimum allowable time-step.

If: DTIMEU < DTIMEL

Result: An error message is printed and the "run" terminated except for CNFAST, CNBACK, CNFWBK and CNVARB.

(7) Set new time (TIMEN)

Set: TIMEN = TPRINT + TSUM + Δt

TPRINT is the time of the last printout.

TSUM is the time from the last printout.

(8) Set mean time (TIMEM)

Set: TIMEM = 1/2 (TIMEN + TIMEØ)

(9) Calculate (or specify) time-step.

The calculated (or specified) time-step for the SINDA numerical routines is as follows:

Numerical Routines		Calculated Time-Step
EXPLICIT	CNFRWD	0.95 * CSGMIN/CSGFAC
EXPLICIT	CNFRDL	0.95 * CSGMIN/CSGFAC
EXPLICIT	CNEXPN	0.95 * CSGMIN * CSGFAC
EXPLICIT	CNDUFR	0.95 * CSGMIN * CSGFAC
EXPLICIT	CNQUIK	0.95 * CSGMIN * CSGFAC
EXPLICIT	CNFAST	larger of CSGMIN or DTIMEL
IMPLICIT	CNBACK	DTIMEI
IMPLICIT	CNFWBK	DTIMEI
IMPLICIT	CNVARB	DTIMEI

CSGMIN = $C_i/\Sigma G_{ij}$ (minimum value, i = 1,2,...,NND)

where: C_i is the capacitance of the ith node

G_{ij} is the conductance from node i to node j

CSGFAC is the time-step factor (see above).

(10) It should be recognized that individual routines may have slight variations to the time-step calculations.

6.2.5 Computation of Temperatures

The actual calculation of temperatures, be it for diffusion nodes or for arithmetic nodes, represents the end result of a long computational procedure with many checks and criteria. Nevertheless, if one confines the discussion to the DØ loops of nodal types, a rather compact but general computational pattern becomes apparent. More details are presented in the individual sections describing each numerical solution routine. (Sections 6.3 - 6.5)

6.2.5.1 Transient Explicit Routines

For the explicit routines the diffusion and arithmetic nodes are treated separately. Diffusion-node temperatures are calculated explicitly, whereas the arithmetic-node temperatures are computed implicitly. This means that at each time-step an iterative loop is set-up for the arithmetic nodes; none is required for the diffusion nodes.

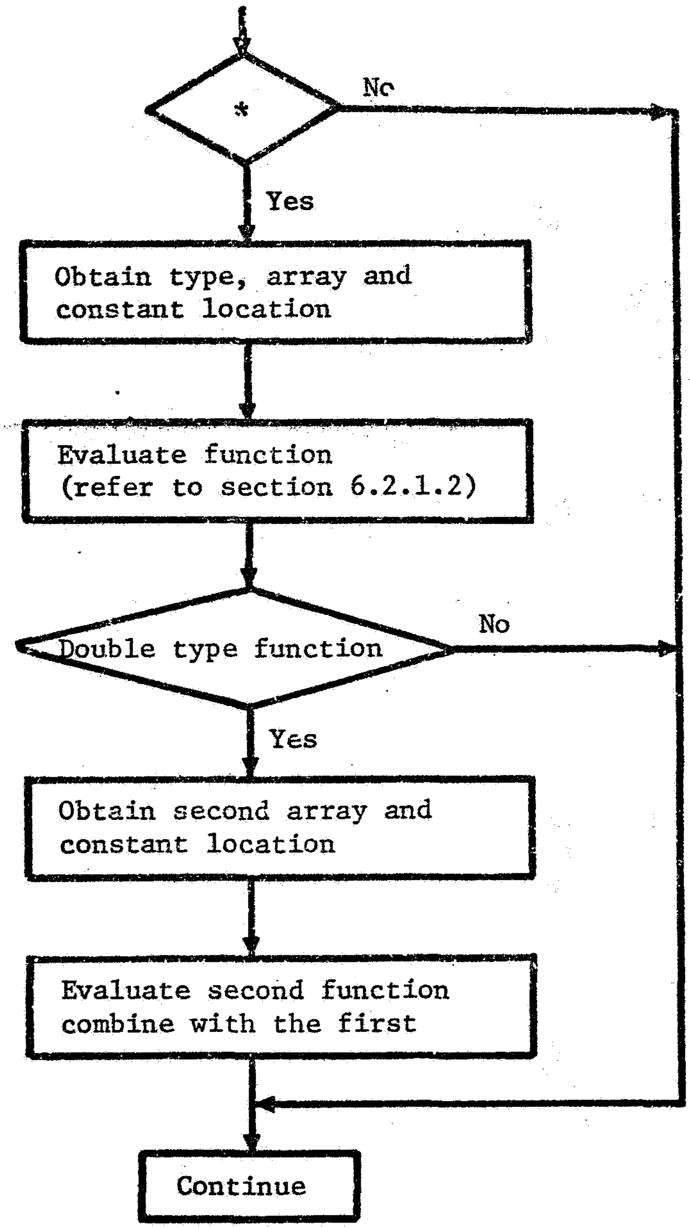
Diffusion-Node Temperatures

Calculation of the diffusion-node temperatures follows the VARIABLES 1 call; the computational pattern is:

DØ-LØØP (I = 1, NND) on the diffusion nodes is established.

The functions associated with the variable capacitance C_i, the variable impressed source q_i, and the variable coefficients G_k (a_{ij} for conduction and σb_{ij} for radiation), between diffusion-diffusion and diffusion-arithmetic nodes are updated at the beginning of each time-step. These functional types are described in Section 6.2.1.2 and the computational pattern is indicated in the flow chart of Figure 6.2-3.

Using the updated C_i, q_i and G_k, the branch heat flow sum, Q_{si}, and conductance sum X_i, are calculated (refer for example to flow chart of Figure 6.3-1).



* Variable capacitance (c_i), impressed source (q_i), or variable coefficient (G_k).

Figure 6.2-3. Evaluation of Nonlinear Capacitance, Source or Conductance

$$Q_{si} = \sum_{j=1}^{p} G_{ij,n} (T_{j,n} - T_{i,n}) + q_{i,n} \qquad (6.2\text{-}1)$$

$$X_i = \sum_{j=1}^{p} G_{ij,n} \qquad (6.2\text{-}2)$$

where, p = total number of nodes; n = time-step old

C_i, q_i, a_{ij}, b_{ij} = optionally specified (refer to Table 6.2-1 - 6.2-4)

$$G_{ij,n} = a_{ij,n} + \sigma b_{ij,n} (T_{j,n}^2 + T_{i,n}^2)(T_{j,n} + T_{i,n})$$

Stability criterion $C_i / \sum_{j=1}^{p} G_{ij,n}$ is computed and the smallest value is stored in control constant CSGMIN. If CSGMIN $\leq$ 0.0, an error message is printed and the "run" terminated.

Diffusion-node temperatures are calculated by using the appropriate finite difference expression associated with each routine. These routines and algorithms are identified as:

CNFRWD and CNFRDL (Section 6.3.1), uses standard forward-difference algorithm.

CNFAST (Section 6.3.2), uses a modified CNFRWD computational procedure to decrease the computational time.

CNEXPN (Section 6.3.3), uses the exponential prediction method.

CNDUFR (section 6.3.4), uses DuFort-Frankel method.

CNQUIK (Section 6.3.5), uses half DuFort-Frankel and half exponential prediction metnod.

Symbolically, the expression for the diffusion-node temperatures may be written as,

$$T_{i,n+1} = T_{i,n} + \frac{\Delta t \; Q_{si}}{C_i} \qquad (6.2\text{-}3)$$

Except for CNFAST the maximum diffusion-node temperature change which is stored in DTMPCC is checked against the allowable diffusion node temperature change which may be specified by the user via the control constant DTMPCA (if not specified DTMPCA = 1.0E+8). If DTMPCA is not satisfied, the time-step is decreased to,

$\Delta t = .95 * \Delta t$ (DTMPCA/DTMPCC) and all temperatures re-set to former values. The computational procedure is repeated with the smaller time-step. CNFAST does not allow for the recalculation of diffusion-node temperatures.

Arithmetic-Node Temperatures

Calculation of the arithmetic-node temperatures always follows the computation of the diffusion-node temperatures and uses "successive point" iteration. The computational pattern is as follows:

Arithmetic-node damping factors DN and DD are established.

DN = DAMPA (optionally specified user constant, if not specified DAMPA = 1.0; factor for the current time-step temperature change)

DD = 1.0 - DN (factor that allows a certain fraction of the "old" temperature to be included as part of the temperature change for the current time-step)

Iterative DØ-LØØP (K=1,NLØØP) is established (NLØØP is the number of iterations specified by the user, if not specified, NLØØP = 1).

DØ-LØØP (I=NND, NND + NNA) for the arithmetic nodes is established.

Impressed source q_i and coefficient G_k (a_{ij} for conduction and σb_{ij} for radiation) are updated once for each time-step.

Using the updated G_k and q_i, the branch heat flow sum Q_{si} and the conductance sum X_i are calculated (refer to flow chart of Figure 6.3-2).

$$Q_{si} = \sum_{j=1}^{p} G_{ij,n} (T_{j,k} - T_{i,k}) \tag{6.2-4}$$

$$X_i = \sum_{j=1}^{p} G_{ij,n} \tag{6.2-5}$$

Arithmetic node temperatures are calculated for each iterative loop by using the following "successive point" expression, which is employed in all of the routines,

$$T_{i,k+1} = DD*T_{i,k} + DN*\left(\frac{q_{i,n} + \sum_{j=1}^{i} G_{ij,n}\, T_{j,k+1} + \sum_{j=i+1}^{p} G_{ij,n}\, T_{j,k}}{\sum_{j=1}^{p} G_{ij,n}}\right) \quad (6.2\text{-}6)$$

where, $i = (NND+1), (NND+2), \ldots, (NND + NNA)$

$T_{j,k}$ = constant, $(NND + NNA) < j \leq p$

p = total number of nodes

$T_{i,k}$ = temperature at kth iteration

$G_{ij,n} = a_{ij,n} + \sigma b_{ij,n}\, (T_{j,\ell}^2 + T_{i,k}^2)(T_{j,\ell} + T_{i,k})$

($\ell = k$ if $j \geq i$ and $\ell = k+1$ if $j < i$)

($a_{ij,n}$ and $b_{ij,n}$ mean updating at time-step, n)

q_i, a_{ij}, b_{ij} = optionally specified (refer to Tables 6.2-1 - 6.2-4)

DN ≡ DAMPA (arithmetic node damping factor)

DD = 1.0 - DN

The maximum arithmetic-node relaxation temperature change is calculated and checked against the allowable arithmetic-node relaxation temperature change which may be specified via the control constant ARLXCA. This relaxation convergence check is made during each iterative step calculation and is used in conjunction with control constant NLØØP. Satisfaction of either ARLXCA or NLØØP during any iterative step terminates the arithmetic-node temperature calculation.

For each time step, except for CNFAST, the maximum arithmetic-node temperature change which is stored in control constant ATMPCC is checked against the allowable arithmetic-node temperature change which may be specified via the control constant ATMPCA (if not specified, ATMPCA = 1.0E+8). If ATMPCA is not satisfied, the time-step is decreased to,

$\Delta t = .95*\Delta t$(ATMPCA/ATMPCC)

and all temperatures re-set to former values. The computational procedure is repeated with the smaller time-step. CNFAST does not allow for recalculation of arithmetic-node temperatures.

6.2.5.2 Transient Implicit Routines

Both diffusion-node and arithmetic-node temperatures are calculated by "successive point" iteration. Although these calculations are performed on the same iterative pass, diffusion node temperatures are evaluated on its own computational loop using a specified algorithm associated with a particular implicit routine. Calculation of the arithmetic-node temperatures is also done on its own computational loop and is identical in all the implicit routines. As a matter of fact, arithmetic-node temperatures are calculated in the same manner in all the SINDA numerical solution routines. Use of a separate computational loop for the diffusion nodes permits the extrapolation of diffusion-node temperatures provided acceleration of convergence criterion is met (refer to Section 6.2.7).

Diffusion-Node Temperatures

In order to facilitate the discussion to follow on the computational procedure, it is convenient to examine the forward-backward finite difference expression.[13]

$$C_i \frac{(T_{i,k+1} - T_{i,n})}{\Delta t} = \beta\, T_{forward} + (1 - \beta)\, T_{backward} \qquad (6.2\text{-}7)$$

where: β = factor with range $0 \leq \beta \leq 1/2$

$$T_{forward} = q_{i,n} + \sum_{j+1}^{p} a_{ij,n}\,(T_{j,n} - T_{i,n}) + \sum_{j=1}^{p} \sigma b_{ij,n}(T^4_{j,n} - T^4_{i,n}) \qquad (6.2\text{-}8)$$

$$T_{backward} = q_{i,n} + \sum_{j+1}^{p} a_{ij,n}(T_{j,k+1} - T_{i,k+1}) + \sum_{j=1}^{p} \sigma b_{ij}(T^4_{j,k+1} - T^4_{i,k+1}) \qquad (6.2\text{-}9)$$

$$i = 1,2,\ldots,N$$

$$T_{j,n};\ T_{j,k+1} = \text{constant},\ N < j \leq p$$

n = nth time-step ; k = kth iteration within a given time-step.

C_i, q_i, a_{ij}, b_{ij} = optionally specified (refer to Tables 6.2-1 -- 6.2-4)

Any value of β less than one yields an implicit set of equations which must be solved simultaneously. For values of β less than or equal to one-half equation (6.2-7) represents an unconditionally stable set of equations, whereas values of β greater than one-half yields a set of equations with conditional stability.

The standard implicit algorithm used in subroutine CNBACK follows directly from equation (6.2-7) by letting $\beta = 0$, whereas the Crank-Nicolson method used in subroutine CNFWBK follows by letting $\beta = 1/2$. Subroutine

CNVARB uses a variable factor which is based upon the ratio of CSGMIN/DTIMEU; this ratio is internally calculated in CNVARB (refer to Section 6.4.3.2). In order to simplify the presentation, the following notation is used.

For CNBACK ($\beta = 0$):

$$Q_i = q_{i,n} + \overline{C}_{i,n} T_{i,n} \tag{6.2-10}$$

$$Q_{sum} = Q_i + \sum_{j=1}^{i} G_{ij,n} T_{j,k+1} + \sum_{j=i+1}^{p} G_{ij,n} T_{j,k} \tag{6.2-11}$$

$$G_{sum} = \overline{C}_{i,n} + \sum_{j=1}^{p} a_{ij,n} \tag{6.2-12}$$

$$G_{ij,n} = a_{ij,n} + \sigma b_{ij,n} T^3_{j,\ell} \tag{6.2-13}$$

($\ell = k$, if $j \geq i$ and $\ell = k+1$, if $j < i$)

$$(q_i)_{ave} = \frac{1}{2} \sum_{j=1}^{p} \sigma b_{ij,n} [(T^4_{i,k}) + (T^4_{i,k})_2], \text{ average heat loss} \tag{6.2-14}$$

from ith node, called radiation damping (refer to Section 6.2.6 for details)

$= 0$, if radiation is not present

For CNFWBK ($\beta = \frac{1}{2}$) (note equation (6.2-7) is multiplied by 2):

$$Q_i = 2q_{i,n} + 2\overline{C}_{i,n} T_{i,n} + \sum_{j=1}^{p} a_{ij,n} (T_{j,n} - T_{i,n})$$

$$+ \sum_{j=1}^{p} \sigma b_{ij,n} (T^4_{j,n} - T^4_{i,n}) \tag{6.2-15}$$

Q_{sum} = same as equation (6.2-11)

$$G_{sum} = 2\overline{C}_{i,n} + \sum_{j=1}^{p} a_{ij,n} \tag{6.2-16}$$

$G_{ij,n}$ = same as equation (6.2-13)

$(q_i)_{ave}$ = same as equation (6.2-14)

For CNVARB (variable β') (note that equation (6.2-7) is multiplied by 2, so that $\beta' = 2\beta$ now ranges, $0 \leq \beta' \leq 1.0$):

$$Q_i = 2q_{i,n} + 2\overline{C}_{i,n}\ T_{i,n} + \beta'\left(\sum_{j=1}^{p} a_{ij,n}(T_{j,n} - T_{i,n}) + \sum_{j=1}^{p} \sigma b_{ij,n}(T^4_{j,n} - T^4_{i,n})\right) \quad (6.2\text{-}17)$$

$$Q_{sum} = Q_i + (2.0 - \beta')\left(\sum_{j=1}^{i} G_{ij,n}\ T_{j,k+1} + \sum_{j=i+1}^{p} G_{ij,n}\ T_{j,k}\right) \quad (6.2\text{-}18)$$

$$G_{sum} = 2\ \overline{C}_{i,n} + (2.0 - \beta') \sum_{j=1}^{p} a_{ij,n} \quad (6.2\text{-}19)$$

$G_{ij,n}$ = same as equation (6.2-13)

$$(q_i)_{ave} = \frac{2.0 - \beta'}{2} \sum_{j=1}^{p} \sigma b_{ij,n}\ [(T^4_{i,k}) + (T^4_{i,k})_2] \quad (6.2\text{-}20)$$

, average heat loss from ith node, called radiation damping (refer to Section 6.2.6 for details)

= 0, if radiation is not present

i = 1,2,...,N

β = 2.0*CSGMIN/DTIMEU (range allowed, $0 \leq \beta' \leq 1.0$)

$T_{j,n}$; $T_{j,k}$ = constant, $N < j \leq p$ (p is the total number of nodes)

n = nth time-step; k = kth iteration

C_i, q_i, a_{ij}, b_{ij} = may be optionally specified (refer to Tables 6.2-1 - 6.2-4)

$\overline{C}_{i,n}$ = $C_{i,n}/\Delta t$

Calculation of the diffusion-node temperatures follows VARIABLES 1 call; the computational pattern is:

Iterative DØ-LØØP (kl=1,NLOOP) for the total nodal system is established.

First Iterative Loop:

DO-LOOP (I=1,NND) on diffusion nodes is established.

The functions associated with the variable capacitance C_i, the variable impressed source q_i, and the variable coefficients G_k (a_{ij} for conduction and σb_{ij} for radiation) between diffusion-diffusion and diffusion-arithmetic nodes are updated once for each time-step. These functional types are described in Section 6.2.1.2 and the computational pattern is indicated in the flow chart of Figure 6.2-3.

All known quantities (those evaluated at time-step n) are summed and are identified by the symbol Q_i (equations 6.2-10, 6.2-15 and 6.2-17).

CSGMIN is evaluated.

Radiation damping is used; average radiation heat loss, $(q_i)_{ave}$, from the ith node is evaluated (refer to Section 6.2.6).

For CNVARB, β'= 2.0*CSGMIN/DTIMEU is calculated.

The diffusion-node temperatures are calculated by "successive point" iteration (actually CNBACK and CNFWBK have slightly different first iterative pattern than CNVARB but the difference is not significant).

$$T_{i,k+1} = DD^*T_{i,k} + DN^* [Q_{sum} - (q_i)_{ave}]/G_{sum} \qquad (6.2\text{-}21)$$

DN = DAMPD (use specified diffusion node damping factor, if not specified, DAMPD = 1.0)

DD = 1.0 - DN

For CNVARB, the diffusion-node relaxation temperature change is calculated; maximum value is stored in DRLXCC.

Second and Succeeding Iterative Loops:

With the iterative loops after the first, those quantities C_i, q_i, and G_k which were updated during the first iteration are held constant.

Diffusion-node temperatures are found by using equation (6.2-21).

The diffusion-node relaxation temperature change is calculated and the maximum value stored in DRLXCC.

Check of DRLXCC against DRLXCA (allowable maximum diffusion-node relaxation temperature change) is made after the arithmetic-node temperature calculations.

Each third iteration, a check on solution convergence is made; if convergence is occurring linear extrapolation to accelerate convergence is made (refer to Section 6.2.7).

Arithmetic-Node Temperatures (if any)

During the first iterative loop the impressed source q_i and coefficient G_k (a_{ij} for conduction and σb_{ij} for radiation) between arithmetic-arithmetic nodes are updated once each time-step. On every loop, arithmetic-node temperatures are calculated using "successive point" iteration. The finite difference algorithm is presented in Section 6.2.5.1 (equation 6.2-6).

The arithmetic-node relaxation temperature change is calculated and the maximum is stored in ARLXCC.

<u>During Each Iterative Loop After the First</u>

Both DRLXCC and ARLXCC are checked against DRLXCA and ARLXCA, respectively. If both DRLXCA and ARLXCA are satisfied, the iteration ceases.

If LØØPCT equals NLØØP the message "RELAXATION CRITERIA NOT MET" is printed.

Both the calculated maximum diffusion-node and arithmetic-node temperature change (stored in DTMPCC and ATMPCC, respectively) are checked against the corresponding allowable temperature change stored in DTMPCA and ATMPCA. If DTMPCA is not satisfied, the time-step is decreased to,

$$\Delta t = .95 * \Delta t (DTMPCA/DTMPCC)$$

and all temperatures re-set to former values. The computational procedure is repeated with the smaller time-step.

If ATMPCA is not satisfied, the time-step is decreased to,

$$\Delta t = .95 * \Delta t (ATMPCA/ATMPCC)$$

and all temperatures re-set to former values. The computational procedure is repeated with the smaller time-step.

6.2.5.3 <u>Steady State Routines</u>

Diffusion nodes and arithmetic nodes are treated separately in CINDSS and CINDSL even though from a physical standpoint a distinction between diffusion nodes (nodes with capacitance) and arithmetic nodes (nodes with no capacitance) doesn't exist. Thus, the set of control constants for the diffusion nodes and another set of control constants for arithmetic nodes are similar to those used in the transient routines. No distinction in the type of nodes is made in CINDSM.

The computational procedure to be discussed applies only to CINDSS and CINDSL: CINDSM is considerably different (refer to Section 6.5.3).

Diffusion-Node Temperatures (nodes specified with capacitance even though the problem is steady state)

An iterative DØ-LØØP (K1=1,NLØØP) is established.

Within this iterative loop a DØ-LØØP (I=1,NND) on the diffusion nodes is made. The functions associated with the impressed source q_i and the variable coefficients G_k (a_{ij} for conduction and σb_{ij} for radiation) between diffusion-diffusion and diffusion-arithmetic nodes are updated each iteration.

Diffusion-node temperatures are calculated using "block" iteration for CINDSS and "successive point" iteration for CINDSL.

"Block" iteration (CINDSS):

$$T_{i,k+1} = DD*T_{i,k} + DN* \frac{\left(q_{i,k} + \sum_{j=1}^{p} G_{ij,k}\, T_{j,k}\right)}{\sum_{j=1}^{p} G_{ij,k}} \tag{6.2-22}$$

$$G_{ij,k} = a_{ij,k} + \sigma b_{ij,k}\,(T_{j,k}^2 + T_{i,k}^2)(T_{j,k} + T_{i,k})$$

DN = DAMPD (diffusion-node damping factor)

DD = 1.0 - DN

i = 1,2,...,NND (number of diffusion nodes)

k = kth iteration; p = total number of nodes

q_i, a_{ij}, b_{ij} = optionally specified to Tables (6.2-1 - 6.2-4)

$T_{j,k}$ = constant, (NND + NNA) $< j \le p$ (NNA is the number of arithmetic nodes

"Successive point" iteration (CINDSL):

$$T_{i,k+1} = DD*T_{i,k} + DN* \frac{\left(q_{i,k} + \sum_{j=1}^{i} G_{ij,k}\, T_{j,k+1} + \sum_{j=i+1}^{p} G_{ij,k}\, T_{j,k}\right)}{\sum_{j=1}^{p} G_{ij,k}} \tag{6.2-23}$$

$$G_{ij,k} = a_{ij,k} + \sigma b_{ij,k}\,(T_{j,\ell}^2 + T_{i,k}^2)(T_{j,\ell} + T_{i,k})$$

($\ell = k$ if $j \geq i$ and $\ell = k+1$ if $j < i$)

DN = DAMPD

DD = 1.0 - DN

i = 1, 2, ..., (NND + NNA)

k = kth iteration; p = total number of nodes

q_i, a_{ij}, b_{ij} = optionally specified to Tables (6.2-1 - 6.2-4)

$T_{j,k}$ = constant, (NND + NNA) $< j \leq p$ (NNA is the total number of arithmetic nodes)

Diffusion-node relaxation temperature change is calculated and the maximum is stored in DRLXCC.

Arithmetic-Node Temperatures (nodes specified with no capacitance)

Within this iterative DØ-LØØP a DØ-LØØP (I=NND+1, NND + NNA) is established.

The functions associated with impressed source q_i and variable coefficients G_k (a_{ij} for conduction and b_{ij} for radiation) between arithmetic-arithmetic nodes are updated each iteration.

Arithmetic-node temperatures are calculated using "successive point" iteration.

$$T_{i,k+1} = AD^*T_{i,k} + AN^* \frac{\left(q_{i,k} + \sum_{j=1}^{p} G_{ij,k}\, T_{j,k+1} + \sum_{j=i+1}^{p} G_{ij,k}\, T_{j,k}\right)}{\sum_{j=1}^{p} G_{ij,k}} \quad (6.2\text{-}24)$$

$$G_{ij,k} = a_{ij,k} + \sigma b_{ij,k} (T_{j,\ell}^2 + T_{i,k}^2)(T_{j,\ell} + T_{i,\ell})$$

$$(\ell = k \text{ if } j \geq i \text{ and } \ell = k{+}1 \text{ if } j < i)$$

AN = DAMPA (arithmetic-node damping factor)

AD = 1.0 - AN

i = (NND+1), (NND+2), ..., (NND + NNA) (number of arithmetic nodes)

k = kth iteration

p = total number of nodes

$T_{j,k}$ = constant, (NND + NNA) $< j \leq p$

The arithmetic-node relaxation temperature change is calculated and the maximum value is stored in ARLXCC.

During Each Iterative Loop

Both DRLXCC and ARLXCC are checked against DRLXCA and ARLXCA, respectively. If both relaxation criteria, DRLXCA and ARLXCA, are satisfied, the iteration ceases.

If both relaxation criteria, DRLXCA and ARLXCA, are not met with NLØØP iterations, the message "ITERATION COUNT EXCEEDED, NLØØP =   " is printed.

Energy balance of the system is calculated and is stored in control constant ENGBAL.

6.2.6 Radiation Damping

Radiation damping denotes an averaging of radiation heat loss technique used to prevent or minimize large temperature oscillations. This method is currently employed in only the implicit routines. The technique which is original with J. D. Gaski is based upon practical and computational considerations. Solution of numerous problems without large temperature oscillations indicates the effectiveness of the approach.

The radiation averaging technique is relatively simple conceptually and rather easily incorporated in the numerical solution routines. The computational pattern is such that the diffusion nodes are encountered sequentially. Let the encountered node be the ith node. A check is made for the presence of a radiation coefficient, $G_k = \sigma b_{ij}$, to the ith node. If one or more radiation connections is present, the radiation heat loss, $(q_i)_{rl}$, from the ith node is calculated based upon the previous temperature $T_{i,k}$.

$$(q_i)_{rl} = \sum_j \sigma b_{ij,n} T_{i,k}^4 \qquad (6.2\text{-}24)$$

where, j = all radiation connections to node i

n = nth time-step

k = kth iteration

Using $(q_i)_{rl}$, a second temperature $(T_{i,k})_2$, is found as follows:

$$(T_{i,k})_2 = [Q_{sum} - (q_i)_{rl}]/G_{sum} \qquad (6.2\text{-}25)$$

where,

$$Q_{sum} = \overline{C}_i \, T_{i,n} + q_{i,n} + \sum_{j=1}^{i} a_{ij,n} \, T_{j,k+1} + \sum_{j=i+1}^{p} a_{ij,n} \, T_{j,k}$$

$$+ \sum_{j=1}^{i} \sigma b_{ij,n} \, T^4_{j,k+1} + \sum_{j=i+1}^{p} \sigma b_{ij,n} \, T^4_{j,k} \qquad (6.2\text{-}26)$$

$$G_{sum} = \overline{C}_i + \sum_{j=1}^{p} a_{ij,n} \qquad (6.2\text{-}27)$$

Note that in the evaluation of $(T_{i,k})_2$, the damping factor DAMPD is not used. Note further that G_{sum} does not contain $\sum_j \sigma b_{ij,n} \, T^3_{i,k}$ since it is accounted for in the radiation loss term, $(q_i)_{r1}$.

Now a second radiation heat loss based on $(T_{i,k})_2$ is found,

$$(q_i)_{r2} = \sum_j \sigma b_{ij,n} \, (T_{i,k})_2^4 \qquad (6.2\text{-}28)$$

Equations (6.2-24) and (6.2-28) are then averaged,

$$(q_i)_{ave} = [(q_i)_{r1} + (q_i)_{r2}]/2.0 \qquad (6.2\text{-}29)$$

This average radiation heat loss from an ith node is used in the diffusion-node finite difference algorithm as follows,

$$T_{i,k+1} = DD^* \, T_{i,k} + DN^* \, \frac{(Q_{sum} - (q_i)_{ave})}{G_{sum}} \qquad (6.2\text{-}30)$$

where, DN = DAMPD

DD = 1.0 - DN

$(q_i)_{ave}$ = average radiation heat loss (equation 6.2-29)

$G_{sum} = \overline{C}_i + \sum_j a_{ij,n}$

Q_{sum} = of the form shown by equation (6.2-26). The actual expression depends upon algorithm. Equation (6.2-26) is for the standard implicit method.

The reason behind the use of $(q_i)_{ave}$ is that if the initial temperature $T_{i,k}$ is too large, the heat loss from the ith node, $(q_i)_{r1}$ would then be too large. As a result the evaluation of $(T_{i,k})_2$ with $(q_i)_{r1}$ would yield a temperature that is too low. Thus, the averaging of of $(q_i)_{r1}$ and $(q_i)_{r2}$ would be much closer to the true heat loss from the ith node. If $T_{i,k}$ is too small then $(T_{i,k})_2$ would be too large; the averaging scheme still holds true.

6.2.7 Acceleration of Convergence by Extrapolation Technique

Several of the SINDA numerical solution routines use an extrapolation technique to accelerate convergence of the iterative procedure. The extrapolation technique is used in the implicit routines CNBACK, CNFWBK, and CNVARB for the iterative temperature solution of the diffusion nodes, but is not used for the iterative temperature solutions of the arithmetic nodes. The extrapolation method is also used in the steady state routines CINDSL and CINDSM for the iterative temperature solution of both the diffusion and the arithmetic nodes.

6.2.7.1 Extrapolation Technique

The extrapolation is based on a zero temperature difference condition which is defined to be a point where the temperature change of a particular node over two successive iterations is zero. The governing equations are developed as follows:

Consider the temperatures of an ith node at three successive iterations as shown in Figure 6.2-4a. Let these temperatures, which are assumed to be successively decreasing (or increasing), be denoted as,

$$T_{i,k-2},\ T_{i,k-1} \text{ and } T_{i,k}$$

where, k is the present iteration

k-1 is the previous iteration

k-2 is two iterations before the kth iteration

By taking the differences,

$$\Delta T_{i,k-1} = T_{i,k-2} - T_{i,k-1}$$

$$\Delta T_{i,k} = T_{i,k-1} - T_{i,k}$$

and plotting these temperature differences as a function of iterations, the iterative point of zero temperature difference can be found by linear extrapolation as shown in Figure 6.2-4b. The corresponding expression for the line is found by using the point, $\Delta T_{i,k}$ at $I = k$ and the slope, $(\Delta T_{i,k} - \Delta T_{i,k-1}) / (k-(k-1))$, to yield,

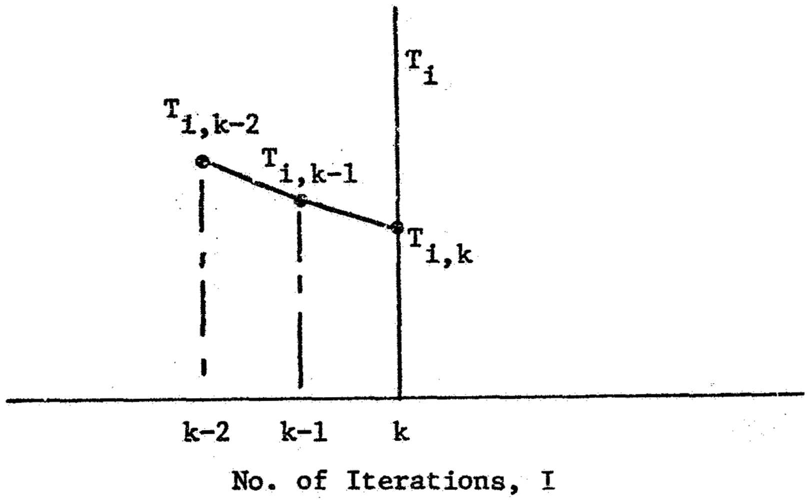


Figure 6.2-4a. Temperature (ith) vs. No. of Iterations

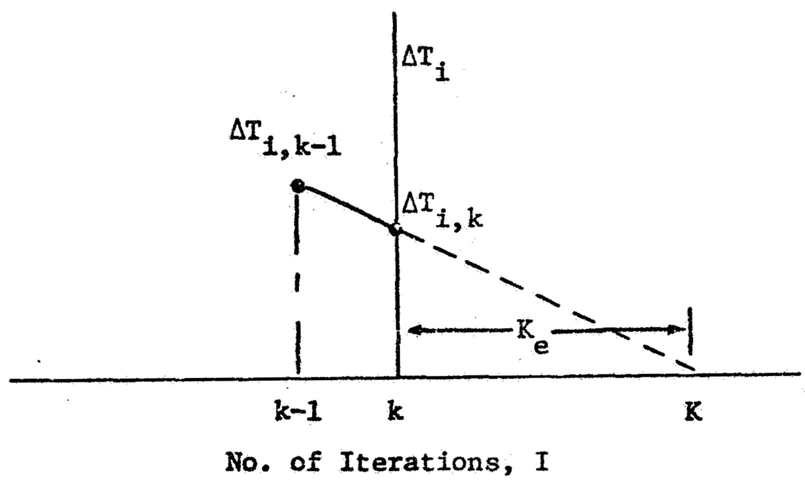


Figure 6.2-4b. Temperature Difference vs. No. of Iterations

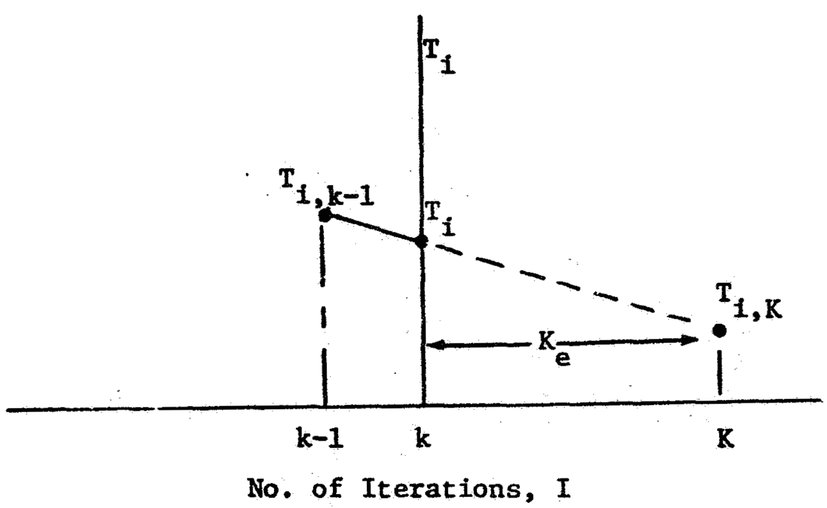


Figure 6.2-4c. Extrapolation of Temperature (ith) to New Value

Figure 6.2-4. Method of Extrapolation to Accelerate Convergence

$$\Delta T_{i,I} = \Delta T_{i,k} + (\Delta T_{i,k} - \Delta T_{i,k-1})(I - k) \qquad (6.2\text{-}31)$$

where, I = iterations

Since at the zero temperature difference condition, $\Delta T_{i,I} = 0$, the expression for the extrapolated iterations, $K_e = (K - k)$, is found to be,

$$K_e = - \Delta T_{i,k}/(\Delta T_{i,k} - \Delta T_{i,k-1}) \qquad (6.2\text{-}32)$$

Now, by extrapolating the line established by the temperatures, $T_{i,k-1}$ and $T_{i,k}$ to the line $I = K$, as shown in Figure 6.2-4c, the extrapolated temperature $T_{i,K}$ is found. The expression is readily found to be,

$$T_{i,I} = T_{i,k} + (T_{i,k} - T_{i,k-1})(I - k) \qquad (6.2\text{-}33)$$

Since $I = K$ and $K - k = K_e$, equation (6.2-33) becomes,

$$T_{i,K} = T_{i,k}\,(1 + K_e) - K_e\, T_{i,k-1} \qquad (6.2\text{-}34)$$

6.2.7.2 Programming Considerations

Each applicable node is tested at the completion of each third iteration to determine if the extrapolation method should be applied. If K_c is calculated to be less than or equal to zero, extrapolation is neglected since the error function is diverging. If K_e is calculated to be greater than zero, a new temperature is calculated based on equation (6.2-34); however, to avoid problems associated with a nearly-zero slope of the line representing the temperature difference vs. number of iterations relationship (Figure 6.2-4b), K_e, is set to a number K_m; otherwise, K_e could be a very large number. For the implicit routines, CNBACK, CNFWBK, and CNVARB, $K_m = 10$. For the steady state routine CINDSL $K_m = 8$ and for steady state routine CINDSM a criterion based upon the maximum temperature is used.

6.2.7.3 Routines Using Acceleration of Convergence

SINDA numerical solution routines that employ the acceleration of convergence features are:

CINDSL, CINDSM	Steady state routines
CNBACK, CNFWBK, CNVARB	Transient implicit routines

6.2.7.4 Comment on Acceleration of Convergence

Neither an extensive study on the value of the acceleration convergence feature has been made, nor has one been reported, but the limited results presently available indicate that for the steady state routine CINDSL, the number of iterations is reduced approximately 20%. Results are not available for the implicit routines.

A study of the acceleration of convergence feature is made difficult because the method is not a user option in the applicable SINDA numerical solution routines. Thus, the user must be sufficiently versed with the routines in order to delete the acceleration of convergence feature.

6.2.8 Other Characteristics of the SINDA Numerical Solution Routines

6.2.8.1 Units

SINDA, as presently coded, requires that the temperatures must be specified in degrees Fahrenheit (°F) since the conversion factor to obtain degrees absolute is internally set at 460.0. This means that the units must be consistent with °F (or °R). The execution routines as presently coded do not permit the use of other units.

6.2.8.2 General Comments on Computational Features

Many of the computational features such as radiation damping are original with J. D. Gaski. No theoretical proofs are offered since a practical "gut-feel" development was often used in lieu of a sophisticated mathematical approach; the features, in general, appear to meet the intended objectives. It should be particularly noted that the numerical solution routines are computationally similar; within a particular numerical solution class explicit, implicit or steady state, the computational similarity is even more pronounced. Yet on the other hand, similarity of patterns are broken for no particular reason other than the programmer's whim.

6.3 Transient Explicit Solution Routines

SINDA explicit solution routine number six. These are identified as follows:

CNFRWD Conditionally stable explicit forward difference.
Requires short pseudo-compute sequence (SPCS).

CNFRDL Identical to CNFRWD except that the long pseudo-compute sequence (LPCS) is required.

CNFAST Modified CNFRWD for accelerated forward differencing.
Requires short pseudo-compute sequence (SPCS).

CNEXPN Unconditionally stable explicit differencing using exponential prediction.
Requires short pseudo-compute sequence (SPCS).

CNDUFR Unconditionally stable explicit differencing using DuFort-Frankel method.
Requires short pseudo-compute sequence (SPCS).

CNQUIK Unconditionally stable explicit differencing using a combination of half CNEXPN and half CNDUFR.
Requires short pseudo-compute sequence (SPCS).

A detailed description of each explicit routine is presented on the pages to follow with heavy reliance upon the general description of Section 6.2. A brief description of these routines is summarized first.

CNFRWD uses an explicit forward differencing algorithm and requires the short pseudo-compute sequence (SPCS). The explicit method is characterized by computational simplicity and stability limitations. Since the allowable time-step is governed by the smallest time constant of the network, care must be given in reducing the physical system to a reasonable lumped-parameter model. Arithmetic-node temperatures are calculated by "successive point" iteration.

CNFRDL is identical to CNFRWD except that CNFRDL requires the long pseudo-compute sequence instead of the short pseudo compute sequence. CNFRDL requires slightly less solution time than CNFRWD but the difference is not significant; CNFRDL does require more core storage, however.

CNFAST represents a modified CNFRWD with the modifications intended to decrease the computational time. A user specified control constant DTIMEL which contains the minimum time-step allowed is used as a criterion for isolating those diffusion nodes that are to receive the steady state calculations. A large pocket of internally converted diffusion nodes can present considerable accuracy problems.

CNEXPN uses an unconditionally stable explicit method with the intent to reduce computational time at the expense of temperature accuracy. If accuracy is an important consideration, another routine such as CNFRWD would be a better choice. As a note of interest, CNEXPN solutions tend to lag in time the true solutions.

CNDUFR uses the unconditionally stable DuFort-Frankel method with the intent to reduce computational time by using time-steps greater than those allowed with the conditionally stable explicit methods. Again accuracy may be compromised. CNDUFR solutions tend to lead in time the true solutions.

CNQUIK uses half CNEXPN and half CNDUFR. Why? Since CNEXPN solutions tend to lag in time and CNDUFR solutions tend to lead in time, a combination may yield better solutions. Preliminary results indicate that CNQUIK solutions are more accurate than either CNEXPN or CNDUFR for the same computational time.

6.3.1 Subroutines: CNFRWD and CNFRDL

6.3.1.1 General Comments

Subroutines CNFRWD and CNFRDL are numerical solution routines that use the forward finite difference explicit approximation[12, 14] of the parabolic differential equation. CNFRWD and CNFRDL are identical except that CNFRDL requires the short pseudo compute sequence (SPCS) whereas CNFRDL requires the long pseudo compute sequence (LPCS). The need for both routines becomes apparent when it is understood that if a steady state numerical solution routine is followed by a transient numerical solution routine, both routines must have consistent PCS (LPCS or SPCS). As a note of interest, each arithmetic node receives the long pseudo compute sequence (LPCS) but this is done internally by the program.

The forward finite difference explicit method as used in CNFRWD and CNFRDL is the conventional Euler method that neither provides a check on the accuracy nor does it provide any scheme of correction once the temperature values are calculated except for the arithmetic nodes which are reiterated NLØØP-times. The explicit method is characterized by computational simplicity and stability limitations with the temperature error at any time point being of the order Δt, $0(\Delta t)$, provided the stability criterion is satisfied. For a rapidly changing boundary condition, such as a heat source, there is no assurance that the calculated temperatures are accurate during the transient period, particularly near the start of the transient, even though the stability criterion is satisfied. Since the allowable time step is governed by the smallest time constant of the network, care must be given in reducing the physical system to a lumped-parameter model. Nonlinearity due to the presence of thermal radiation exchange or temperature-time varying coefficients can lead to numerical solution difficulties; the presence of arithmetic nodes can also present difficulties. These routines offer a number of control constants many of which can be optionally specified by the user to affect the numerical results.

Even with the experience gained through the use of these routines, no realistic criteria can be stated except for the qualitative guidelines indicated above. It is thus recommended that the user becomes familiar with various control constants and their role. The presentation to follow is intended to provide the instructional information.

6.3.1.2 Finite Difference Approximation and Computational Algorithm

The forward finite difference explicit formulation of the lumped parameter heat balance equations was presented in Section 5.2.1. For convenience, the expression is repeated here.

$$C_i \frac{(T_{i,n+1} - T_{i,n})}{\Delta t} = q_{i,n} - \sum_{j=1}^{p} a_{ij} (T_{j,n} - T_{i,n}) + \sum_{j=1}^{p} \sigma b_{ij} (T^4_{j,n} - T^4_{i,n})$$

(From equation 5.2-1 of Section 5.2.1)

where, $i = 1,2,\ldots,N$

$T_{j,n}$ = constant, $N < j \leq p$

p = total number of nodes

Δt = time-step

n = nth time-step

By letting $G_{ij,n} = a_{ij,n} + \sigma b_{ij,n} (T^2_{j,n} + T^2_{i,n})(T_{j,n} + T_{i,n})$, equation (5.2-1) becomes,

$$C_i \frac{(T_{i,n+1} - T_{i,n})}{\Delta t} = q_{i,n} + \sum_{j=1}^{p} G_{ij,n} (T_{j,n} - T_{i,n}) \qquad (6.3\text{-}1)$$

The algorithm as used in the subroutines for the diffusion nodes and for the arithmetic nodes may be expressed as follows.

Diffusion Nodes

$$T_{i,n+1} = T_{i,n} + \frac{\Delta t}{C_i} \left[q_{i,n} + \sum_{j=1}^{p} G_{ij,n} (T_{j,n} - T_{i,n}) \right] \qquad (6.3\text{-}2)$$

where, n = nth time-step

Δt = time-step (refer to Section 6.2.4)

$i = 1,2,\ldots,NND$ (number of diffusion nodes)

$T_{j,n}$ = constant, $(NND + NNA) < j \leq p$ (NNA is the number of arithmetic nodes and p is the total number of nodes)

C_i, q_i, a_{ij}, b_{ij} = may be optionally specified (refer to Tables 6.2-1 through 6.2-4).

Arithmetic Nodes (if any)

$$T_{i,k+1} = DD^* \, T_{i,k} + DN^* \frac{\left(q_{i,n} + \sum_{j=1}^{i} G_{ij,n} \, T_{j,k+1} + \sum_{j=i+1}^{p} G_{ij,n} \, T_{j,k}\right)}{\sum_{j=1}^{p} G_{ij,n}} \qquad (6.3\text{-}3)$$

where, k = kth iteration loop; i = (NND + 1), (NND + 2),...,(NND + NNA)

C_i, q_i, a_{ij}, b_{ij} = optionally specified (refer to Tables 6.2-1 - 6.2-4)

$T_{j,k}$ = constant, (NND + NNA) < $j \leq p$ (NNA is the number of arithmetic nodes and p is the total number of nodes)

DN ≡ DAMPA (arithmetic node damping factor, refer to Section 6.2.3.2)

DD = 1.0 - DN

$$G_{ij,n} = a_{ij,n} + \sigma b_{ij,n} \, (T_{j,\ell}^2 + T_{i,k}^2)(T_{j,\ell} + T_{i,k})$$

$(\ell = k$, if $j \geq i$ and $\ell = k+1$, if $j < i)$

6.3.1.3 Comments on the Computational Procedure

The important steps of the computational procedure used in subroutines CNFRWD and CNFRDL are indicated in Table 6.3-1. For a detailed step-by-step computational description, the user must examine the computer listings for CNFRWD and CNFRDL presented in Appendix A, but some general computational details are given in Section 6.2.5.1. Both CNFRWD and CNFRDL use essentially the same computational steps with the difference occurring in the calculation of the diffusion-node temperatures as shown in the flow chart of Figure 6.3-1; a flow chart for the calculation of the arithmetic-node temperatures is shown in Figure 6.3-2. A functional flow chart of CNFRWD and CNFRDL is shown in Figure 6.3-3. The difference between CNFRWD and CNFRDL is due to the use of the short pseudo-compute sequence (SPCS) by CNFRWD and the use of the long pseudo-compute sequence (LPCS) by CNFRDL.

All diffusion-node temperatures are calculated by a two-pass operation prior to the calculation of the arithmetic node temperatures. On the first pass the pseudo-compute sequence for the diffusion nodes is addressed and the heat flow is calculated and the direction determined for each conductor encountered; the appropriate heat flow and conductance summations are performed. Refer to Section 6.2.5.1 for more details on the computational procedure.

The stability criterion of each diffusion node is calculated and the minimum value is placed in control constant CSGMIN. The time-step used (stored in control constant DTIMEU) is calculated as 95% of CSGMIN divided by control constant CSGFAC which is set at 1.0 unless specified larger by the user. A "look ahead" feature is used when DTIMEU is calculated. If one time-step will pass the output time point the time-step is set to lie on the output time point; if two time-steps will pass the output time point, the time-step is set so that the end of the two time-steps will lie on the output time point. DTIMEU is checked against both DTIMEH and DTIMEL. If DTIMEU exceeds DTIMEH, DTIMEU is set equal to DTIMEH, and if DTIMEU is less than DTIMEL, the "run" is terminated. DTIMEL is internally set to zero if not specified and DTIMEH is set to 1.0E+8 if not specified. The maximum diffusion node temperature change over a time-step is placed in control constant DTMPCC and is checked against the allowable diffusion node temperature change stored in the optionally user specified control constant DTMPCA which is not specified is set to 1.0E+8. If DTMPCC is larger than DTMPCA, DTIMEU is shortened and the calculations repeated. Refer to Section 6.2.4 for detailed procedure on time-step calculation.

The user may iterate the arithmetic node calculations during a time-step by specifying control constant NLØØP and adjust the solution by the use of ARLXCA. The maximum arithmetic node temperature change over an iteration is placed in control constant ARLXCC and is checked against the arithmetic node temperature change criterion stored in ARLXCA. Satisfaction of either NLØØP or ARLXCA terminates the iterative process for that time-step. <u>If the arithmetic node iteration count exceeds NLØØP the results are retained and computation proceeds without user notification</u>. The maximum arithmetic node temperature change over the time-step is stored in control constant ATMPCC and is checked against the allowable temperature change stored in ATMPCA. If larger, the time-step is shortened and the calculation repeated. The user may also specify the control constant DAMPA in order to dampen possible oscillation due to nonlinearities.

6.3.1.4 Control Constants

Control constants ØUTPUT and TIMEND (> TIMEØ) must be specified as indicated in Table 6.2-5 and described in Section 6.2.3.2; otherwise the "run" will terminate with an error message. The function of optionally specified control constants ARLXCA, ATMPCA, BACKUP, CSGFAC, DAMPA, DTIMEH, DTIMEL, DTMPCA, NLØØP, and TIMEØ is described in Section 6.2.3.2. Note particularly that TIMEØ may be set negative and that NLØØP is set to one if not specified.

6.3.1.5 Error and Other Messages

If control constants ØUTPUT and TIMEND are not specified, the following error message will be printed for each,

ØUTPUT	"NØ ØUTPUT INTERVAL"
TIMEND	"TIME STEP TØØ SMALL"

The reason for the TIMEND error message is that a direct check on TIMEND is not made; the resultant error message just happens to be a quirk in the coding.

If the short pseudo-compute sequence SPCS is not specified, the error message will be,

"CNFRWD REQUIRES SHORT PSEUDØ-COMPUTE SEQUENCE"

If the long pseudo-compute LPCS is not specified, the error message will be,

"CNFRDL REQUIRES LONG PSEUDO-COMPUTE SEQUENCE"

If the dynamic storage allocation is not sufficient (NDIM < (NND + NNA)), the message will be,

"_____ LØCATIONS AVAILABLE"

Note that the number printed will be negative indicating the additional storage locations required.

If the time-step used is less than the time-step allowed (DTIMEL) which may be optionally specified by the user, the message will be,

"TIME STEP TØØ SMALL"

If CSGMIN $\leq$ 0, the message printed will be,

"CSGMIN ZERØ or NEGATIVE"

Checks on the control constants, the pseudo-compute sequence and the dynamic storage allocation are made in the following sequence with the "run" terminating if a single check is not satisfied, ØUTPUT, pseudo-compute sequence, dynamic storage locations

It should be particularly noted that no message is printed if ARLXCA is not satisfied with NLØØP iterations; ARLXCA and NLØØP are optionally specified control constants.

Table 6.3-1. Basic Computational Steps for CNFRWD and CNFRDL

1. Specification of control constants (all control constants are pre-set to zero). Control constants ØUTPUT and TIMEND must be specified. SPCS is required for CNFRWD and LPCS for CNFRDL. (Refer to Table 6.2-4 for nominal values and Section 6.2.3.2 for description.)

2. Sufficiency check on dynamic storage. Requirements = NND + NNA (NND = diffusion nodes and NNA = arithmetic nodes).

3. Setting and/or calculation of time-step, Δt. (Refer to Section 6.2.4 for detailed procedure.)

4. Setting of source and diffusion node dynamic storage locations at zero.

5. Calling of VARIABLES 1. (Refer to Section 6.2.2.2.)

6. Checking of BACKUP. (Refer to Section 6.2.3.2.)

7. Calculation of diffusion-node temperatures. (Refer to Section 6.2.5.1 for description and to flow chart of Figure 6.3-1.)

   Diffusion-node temperatures are calculated by using: (refer to Section 6.3.1.2.)

$$T_{i,n+1} = T_{i,n} + \Delta T_{i,n}$$

   where, $\Delta T_{i,n} = \frac{\Delta t}{C_{i,n}} \left[q_{i,n} + \sum_{j=1}^{p} G_{ij,n} (T_{j,n} - T_{i,n}) \right]$

8. Erasure of all temperature calculations for latest time-step if allowable temperature change criterion DTMPCA is not satisfied and recalculation of temperatures with reduced time-step.

9. Calculation of arithmetic-node temperatures; if the number of iterations equals NLØØP the temperatures are retained without user modification. (Refer to Section 6.2.5.1 for description and to flow chart of Figure 6.3.2)

10. Erasure of all temperature calculations for latest time-step if allowable temperature change criterion ATMPCA is not satisfied and recalculation of temperatures with reduced time-step.

11. Setting of BACKUP to 0.0 and the calling of VARIABLES 2.

    If BACKUP is nonzero, temperatures are re-set to former values and the computational procedure repeated.

12. Advancing of time, checking of time to print, and the printing at the output interval.

13. Calling of ØUTPUT CALLS.

14. Checking for problem end-time stored in user specified control constant TIMEND.

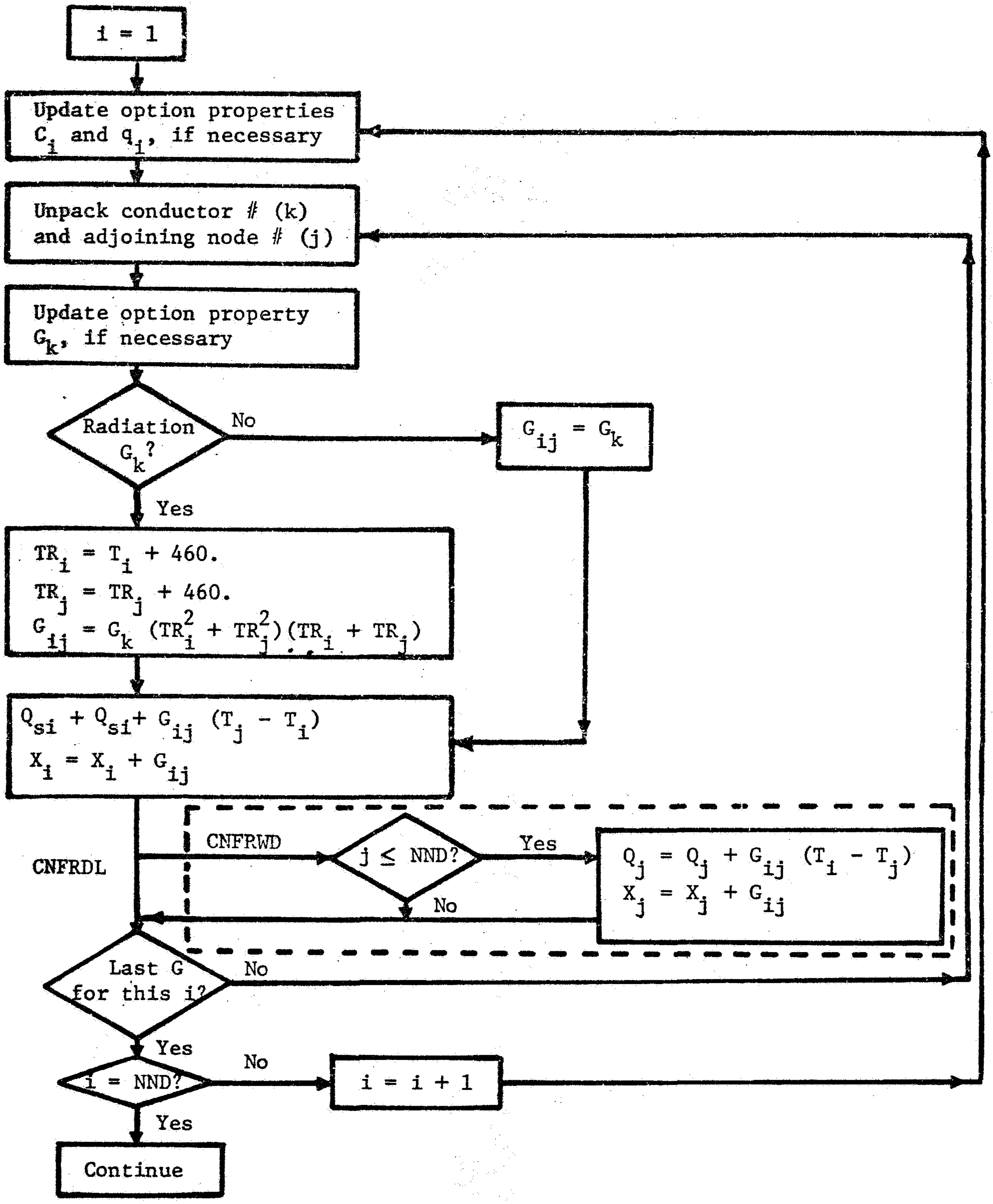


Figure 6.3-1. QSUM and GSUM for "Block" Diffusion-Node Temperature Calculation, CNFRWD and CNFRDL

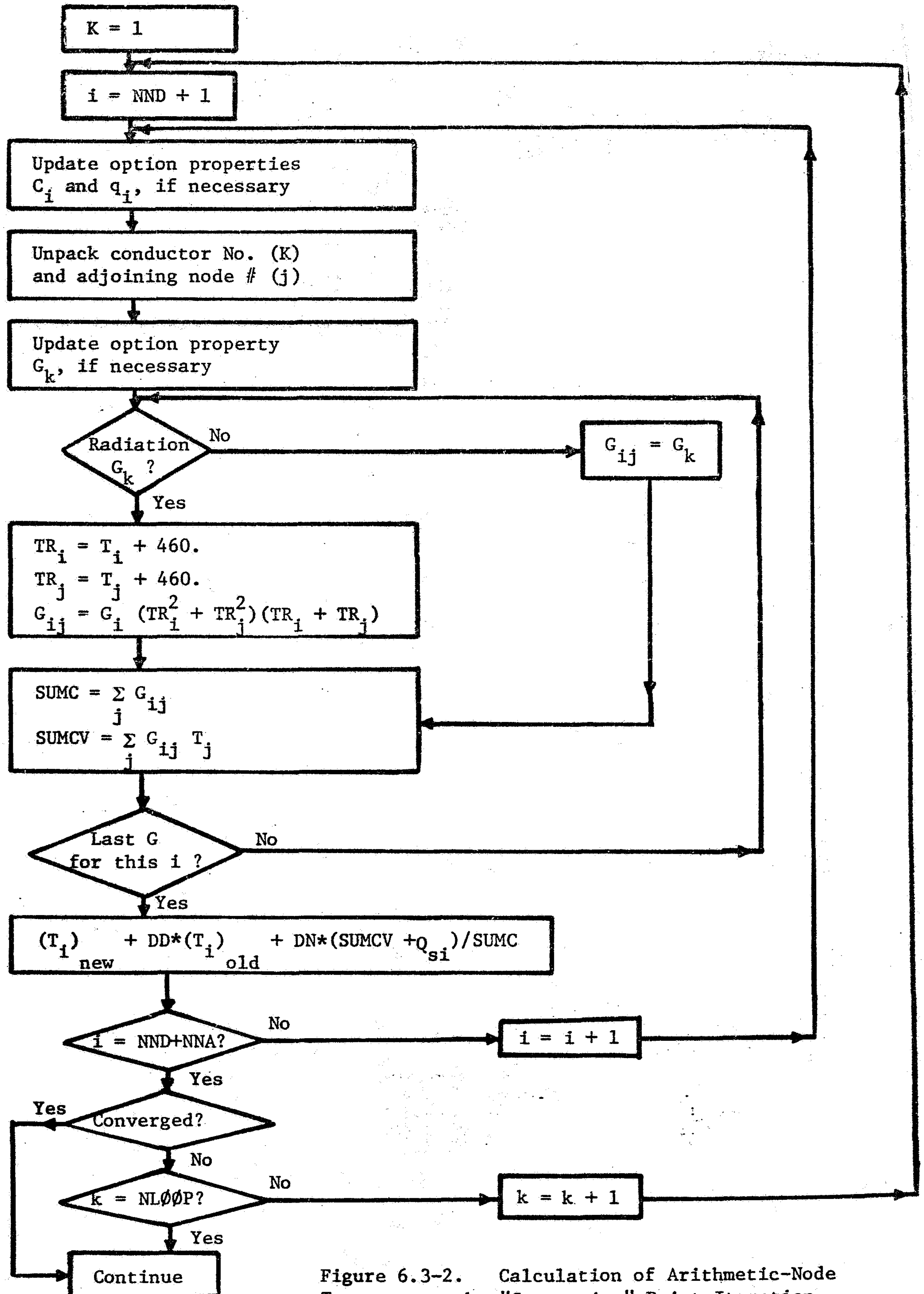


Figure 6.3-2. Calculation of Arithmetic-Node Temperatures by "Successive" Point Iteration

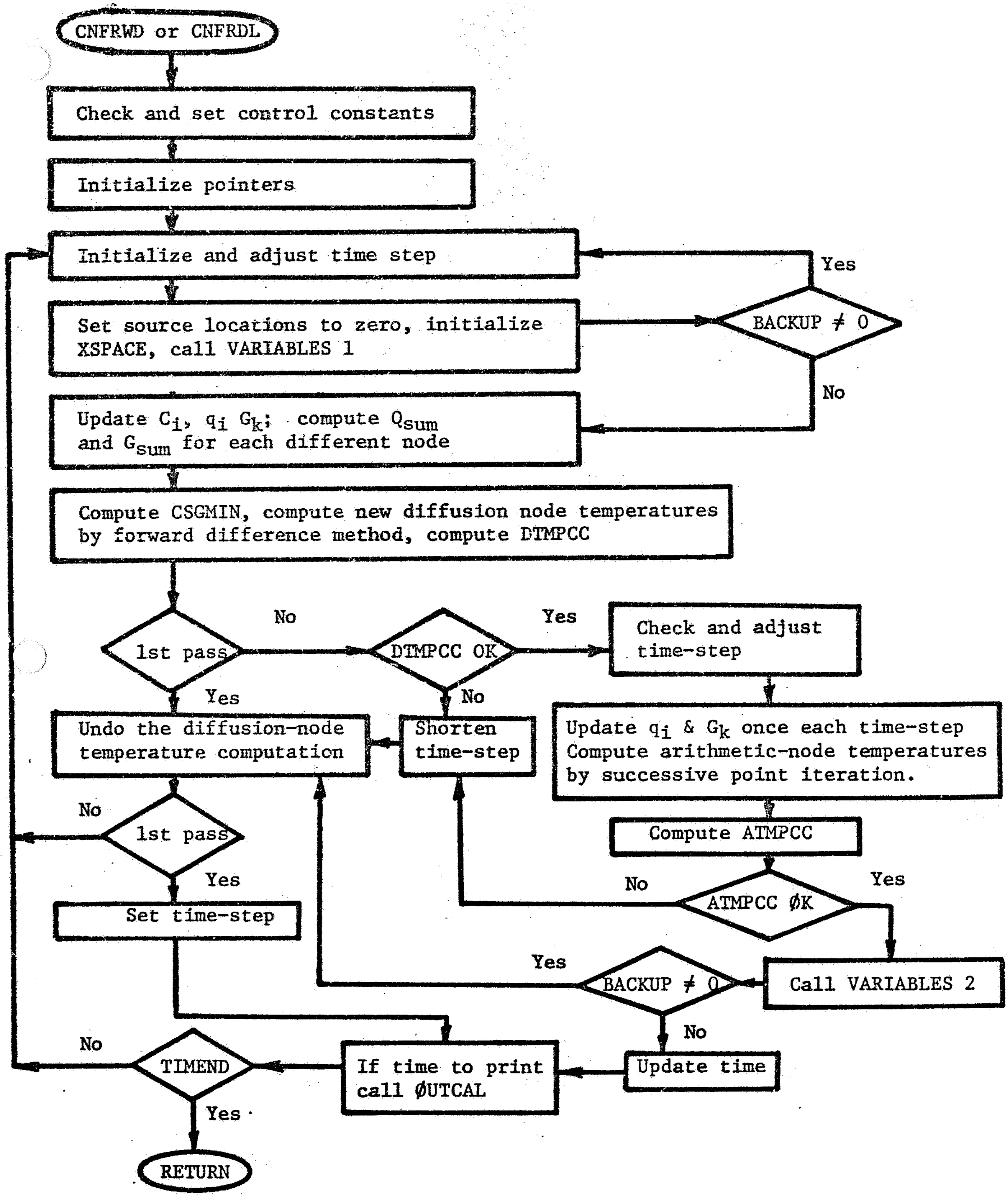


Figure 6.3-3. Functional Flow Chart for CNFRWD and CNFRDL

6.3.2 Subroutine: CNFAST

6.3.2.1 General Comments

Subroutine CNFAST, which requires the short pseudo compute sequence (SPCS) represents a modified CNFRWD with the modifications intended to decrease the computational time. Use of CNFAST requires a user specification of control constant DTIMEL which represents the minimum time-step allowed in addition to control constant ØUTPUT. With minimum computational time and adequate temperature values as the objective, the computational procedure is simplified. A number of checks on control constants are eliminated and temperature nodes with CSGMIN less than the allowable time-step, DTIMEL, are calculated using the steady state equations.

Although experience on the use of CNFAST is rather limited at this time, it is clear that the user specified DTIMEL should be sufficiently small that only a small number of the diffusion nodes should receive the steady state equations. These steady state equations are computed only once during a time-step and thus are not treated computationally the same as the other user-specified arithmetic nodes. A large pocket of internally converted diffusion nodes would lead to large temperature inaccuracies.

6.3.2.2 Finite Difference Approximation and Computational Algorithm

The finite difference expressions for CNFAST are the same as those indicated in Section 6.3.1.2 for subroutines CNFRWD and CNFRDL, but the application of these equations in the computation procedure is different.

Diffusion Nodes

If the user specified control constant, DTIMEL, which represents the maximum time-step allowed as specified by the user is less than or equal to CSGMIN, the diffusion node temperature is calculated as,

$$T_{i,n+1} = T_{i,n} + \frac{\Delta t}{C_i}\left[q_{i,n} + \sum_{j=1}^{p} G_{ij,n}\ (T_{j,n} - T_{i,n})\right] \qquad 6.3\text{-}4)$$

where, Δt = time-step (refer to Section 6.2.4); n = nth time-step

C_i, q_i, a_{ij}, b_{ij} = optionally specified (refer to Tables 6.2-1 - 6.2-4)

i = 1,2,...,NND (of diffusion nodes with DTIMEL $\leq$ CSGMIN)

$T_{j,n}$ = constant, (NND + NNA) $< j \leq p$ (NNA is the number of arithmetic nodes and p is the total number of nodes)

$G_{ij,n} = a_{ij,n} + \sigma b_{ij,n}\ (T_{j,n}^2 + T_{i,n}^2)(T_{j,n} + T_{i,n})$

If DTIMEL > CSGMIN, the time-step is set at DTIMEL and the diffusion node temperature calculated with no iterations as,

$$T_{i,n+1} = \left(\frac{q_{i,n} + \sum_{j=1}^{p} G_{ij,n} \left(T_{j,n} - T_{i,n}\right)}{\sum_{j=1}^{p} G_{ij,n}} \right) \tag{6.3-5}$$

where, n means the nth time-step

$i = 1,2,\ldots,$NND (number of diffusion nodes with DTIMEL > CSGMIN)

Arithmetic Nodes (if any)

The arithmetic-node temperatures are calculated in the same manner as in CNFRWD (Section 6.3.1.2) or refer to Section 5.2.3 for finite difference algorithm.

6.3.2.3 Comments on the Computational Procedure

The important steps of the computational procedure used in subroutine CNFAST are indicated in Table 6.3-2 and a functional flow chart is shown in Figure 6.3-4. For a detailed computational description, the user should examine the computer listing for CNFAST in Appendix A, but some general computational details are presented in Section 6.2.5.1. The computational procedure is similar to the one used in CNFRWD with the major difference being the use of DTIMEL which represents the user specified minimum time-step allowed. The time-step calculations stored in DTIMEU proceed exactly as in CNFRWD until the check with DTIMEL is made. If DTIMEU (CSGMIN of a node) ≥ DTIMEL, the diffusion node temperature calculation is identical to CNFRWD. If DTIMEU (CSGMIN of a node) < DTIMEL, the diffusion node receives the steady state calculation.

Control constants DTMPCA which contains the allowable diffusion-node temperature change and ATMPCA which contains the arithmetic-node temperature change are not checked in CNFAST. Thus time-steps are not shortened and temperature calculations repeated. The remainder of the computational procedure follows those of CNFRWD (Section 6.3.1.3).

6.3.2.4 Control Constants

Control constants DTIMEL, ØUTPUT and TIMEND (>TIMEØ) must be specified as indicated in Table 6.2-5 and described in Section 6.2.3.2;

otherwise the "run" will terminate with an error message. The function of optionally specified control constants ARLXCA, BACKUP, DAMPA, DTIMEH, NLØØP and TIMEØ is described in Section 6.2.3.2. As mentioned before in a previous paragraph, the user should take considerable amount of caution in specifying DTIMEL in order to prevent large pockets of nodes that receive the steady state equation without reiteration. Note also that TIMEØ may be set negative and that NLØØP is set to one if not specified.

6.3.2.5 Error and Other Messages

If control constants DTIMEL, ØUTPUT and TIMEND are not specified, the following error message will be printed for each,

DTIMEL	"NØ DTIMEL"
ØUTPUT	"NØ ØUTPUT INTERVAL"
TIMEND	no message

A direct check on TIMEND is not made; an indirect message is printed for the other explicit routines but is not output for CNFAST.

If the short pseudo-compute sequence SPCS is not specified, the error message will be,

"CNFAST REQUIRES SHORT PSEUDO-COMPUTE SEQUENCE"

If the dynamic storage allocation is not sufficient (NDIM < NND), the message will be,

"_____ LOCATIONS AVAILABLE"

Note that the number printed will be negative indicating the additional storage locations required.

If CSGMIN $\leq$ 0, the message printed will be,

"C/SK ZERØ or NEGATIVE"

Checks on the control constants, the pseudo-compute sequence and the dynamic storage allocation are made in the following sequence, with the run terminating if a single check is not satisfied,

ØUTPUT, DTIMEL, pseudo-compute sequence, and dynamic storage locations.

It should be particularly noted that no message is printed if ARLXCA is not satisfied with NLØØP iterations; ARLXCA and NLØØP are optionally specified control constants.

Table 6.3-2. Basic Computational Steps for CNFAST

1. Specification of control constants. Control constants DTIMEL, ØUTPUT and TIMEND must be specified. SPCS is required for CNFAST. (Refer to Table 6.2-5 for values and Section 6.2.3.2 for descriptions.)

2. Sufficiency check on dynamic storage. Requirements = NND (NND = diffusion nodes).

3. Setting and/or calculation of time-step, Δt. (Refer to Section 6.2.4 for detailed procedure.)

   Note that initial time-step equal DTIMEL and subsequent time-step is the larger of CSGMIN or DTIMEL.

4. Setting of source and diffusion node dynamic storage locations to zero.

5. Calling of VARIABLES 1. (Refer to Section 6.2.2.2 for description.)

6. Checking of BACKUP. (Refer to Section 6.2.3.2 for description.)

7. Calculation of diffusion-node temperatures. (Refer to Section 6.2.5.1 for description.) Calculation differs from the other explicit routines, since diffusion nodes with CSGMIN less than DTIMEL receive steady state calculation (refer to Section 6.3.2.2.)

   If DTIMEL $\leq$ CSGMIN, the node temperature is calculated as,

$$T_{i,n+1} = T_{i,n} + \frac{\Delta t}{C_i}\left[\sum_{j=1}^{p} G_{ij,n}\,(T_{j,n} - T_{i,n}) + q_{i,n}\right]$$

   If DTIMEL > CSGMIN, the node temperature is calculated using the steady state expression,

$$T_{i,n+1} = \left(\frac{q_{i,n} + \sum_{j=1}^{p} G_{ij,n}\,(T_{j,n} - T_{i,n})}{\sum_{j=1}^{p} G_{ij,n}}\right)$$

8. Calculation of arithmetic-node temperatures if the number of iterations equals NLØØP the temperatures are retained without user notification. (Refer to Section 6.2.5.1 for details.)

9. Calling of VARIABLES 2. (Refer to Section 6.2.2.3 for description.)

10. Advancing of time, checking of time to print, and the printing at the the output interval.

11. Calling of ØUTPUT CALLS.

12. Checking for problem end-time stored in user specified control constant TIMEND

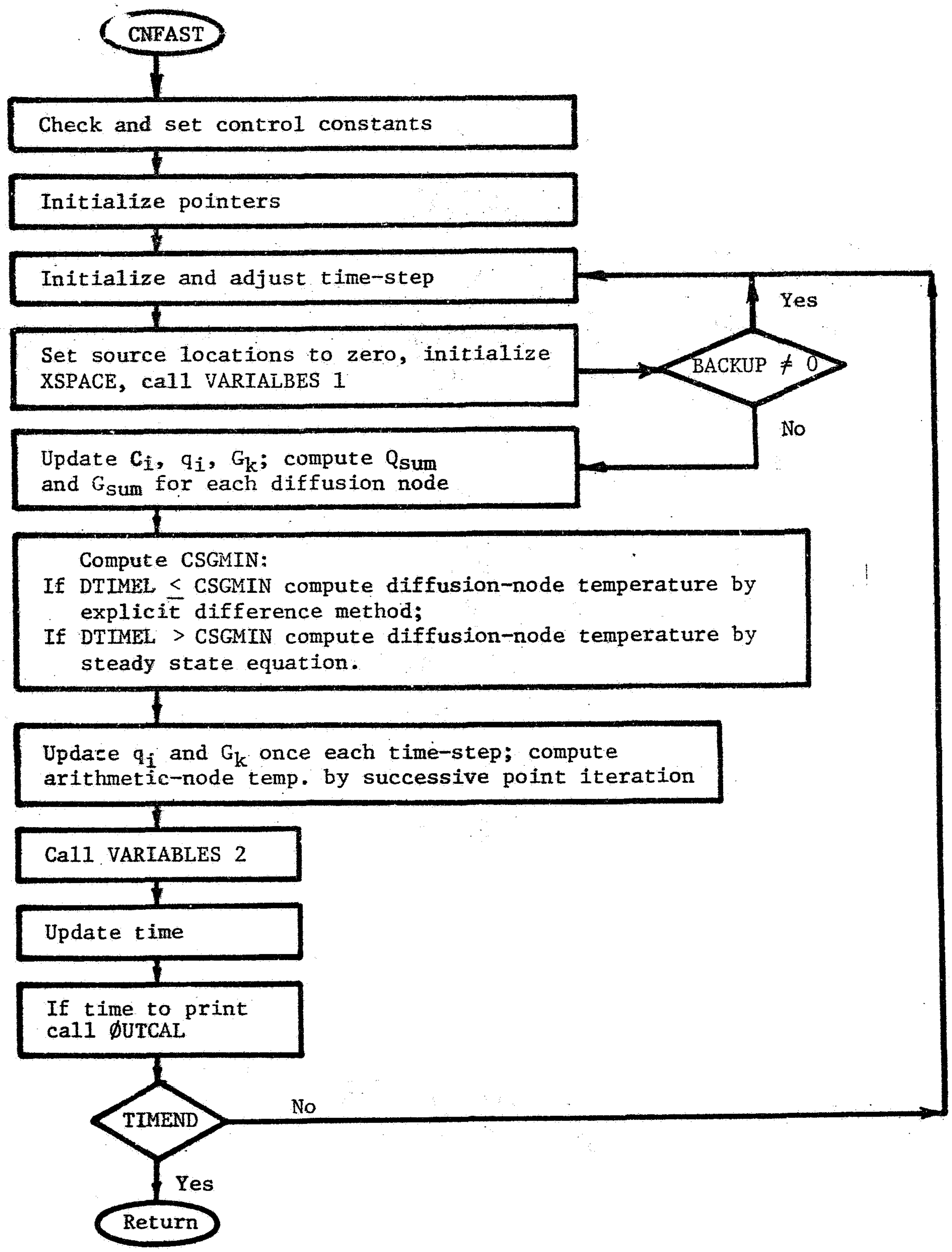


Figure 6.3-4. Functional Flow Chart for CNFAST

6.3.3 Subroutine: CNEXPN

6.3.3.1 General Comments

Subroutine CNEXPN is an explicit routine based upon the exponential prediction method;[1, 17] the method being unconditionally stable permits any size time-steps and requires the short pseudo-compute sequence (SPCS). An infinite time-step reduces the transient equation to a steady state one. Although the method is unconditionally stable, stability should not be confused with accuracy. Comparison of several numerical methods, including the exponential approximation, is given in Reference 17.

If accuracy is an important consideration, time-steps should not be larger than those taken with the standard explicit method such as used in CNFRWD. If high accuracy is not an important consideration, considerable savings in computational time can be affected with the use of a large time-step. It should be noted that the same savings in computational time may be possible with the implicit routines. As another note of interest, CNEXPN solutions have a tendency to lag in time the true temperatures.

6.3.3.2 Finite Difference Approximation and Computational Algorithm

Diffusion Nodes

The expression for the numerical method used in subroutine CNEXPN for solving the diffusion-node temperatures may be derived from the heat balance equation (5.1-6).

$$\frac{dT_i}{dt} = \frac{1}{C_i}\left[q_i + \sum_{j=1}^{p} a_{ij}\ (T_j - T_i) + \sum_{j=1}^{p} \sigma b_{ij}\ (T_j^4 - T_i^4)\right] \qquad \text{(equation 5.1-6 of Section 5)}$$

$$i = 1,2,\ldots,N$$

$$T_j = \text{constant},\ N < j \leq p$$

If $G_{ij} = a_{ij} + \sigma b_{ij}\ (T_j^2 + T_i^2)(T_j + T_i)$ equation (5.1-6) becomes,

$$\frac{dT_i}{dt} = \frac{1}{C_i}\left[q_i + \sum_{j=1}^{p} G_{ij}\ (T_j - T_i)\right] \qquad (6.3\text{-}6)$$

$$i = 1,2,\ldots,N$$

$$T_j = \text{constant},\ N < j \leq p$$

If we further let G_{ij}, q_i and T_j be invariant with time and temperature, equation (6.3-6) may be integrated rather easily to yield,

$$T_{i,n+1} = T_{i,n} e^{-\alpha_n \Delta t} + \frac{q_{i,n} + \sum_{j=1}^{p} G_{ij,n} T_{j,n}}{\sum_{j=1}^{p} G_{ij,n}} \left(1 - e^{-\alpha_n \Delta t}\right) \quad (6.3\text{-}7)$$

where, n = nth time-step; $i = 1, 2, \ldots,$ NND (number of diffusion nodes)

C_i, q_i, a_{ij}, b_{ij} = may be optionally specified (refer to Tables 6.2-1 - 6.2-4)

$T_{j,n}$ = constant, (NND + NNA) $< j \leq p$ (NNA is the number of arithmetic nodes and p is the total number of nodes)

$$\alpha_n = \frac{\sum_{j=1}^{p} G_{ij,n}}{C_{i,n}}$$

Δt = time-step (refer to Section 6.2.4)

$$G_{ij,n} = a_{ij,n} + \sigma b_{ij,n} (T_{i,n}^2 + T_{j,n}^2)(T_{i,n} + T_{j,n})$$

Computationally equation (6.3-7) is applied to the diffusion nodes. It should be noted that the form of equation (6.3-7) represents a "block" change in temperatures since the evaluation of $T_{i,n+1}$ is based upon $T_{i,n}$.

Arithmetic Nodes (if any)

Arithmetic-node temperatures are calculated in the same manner as in CNFRWD (Section 6.3.1.2) or refer to Section 5.2.3 for finite difference algorithm.

6.3.3.3 Comments on the Computational Procedure

The important steps of the computational procedure used in subroutine CNEXPN are indicated in Table 6.3-3 and a functional flow chart is shown in Figure 6.3-5. A detailed computational procedure requires the examination of the CNEXPN computer listing which is presented in Appendix A but some general computational details are given in Section 6.2.5.1. The computational process of subroutine CNEXPN is essentially identical to CNFRWD with the difference being the finite difference expression used for the calculation of the diffusion nodes and the time-step which is calculated as CSGMIN*CSGFAC in lieu of CSGMIN/CSGFAC. The "look ahead" feature for time-step calculation as well as a check with DTIMEH, DTIMEL and DTMPCA is identical to CNFRWD. Temperatures of arithmetic nodes are calculated after the diffusion nodes and utilize NLØØP, ARLXCA, and DAMPA in exactly the same way as CNFRWD. The verbal flow description of CNFRWD (Section 6.3.1.3) applies here except for the differences indicated above.

6.3.3.4 Control Constants

Control constants ØUTPUT and TIMEND (> TIMEØ) must be specified as indicated in Table 6.2-5 and described in Section 6.2.3.2; otherwise the "run" will terminate with an error message. The function of optionally specified control constants ARLXCA, ATMPCA, BACKUP, CSGFAC, DAMPA, DTIMEH, DTIMEL, DTMPCA, NLØØP, and TIMEØ is described in Section 6.2.3.2. The user should take particular care in the selection of CSGFAC since too large of a time-step would lead to grossly inaccurate temperatures even though the solution is stable. Note also that TIMEØ may be set negative and that NLØØP is set to one if not specified.

6.3.3.5 Error and Other Messages

If control constants ØUTPUT and TIMEND are not specified, the following error message will be printed for each,

ØUTPUT	"NØ ØUTPUT INTERVAL"
TIMEND	"TIME STEP TØØ SMALL"

The reason for the TIMEND error message is that a direct check on TIMEND is not made; the resultant error message just happens to be a quirk in the coding.

If the short pseudo-compute sequence SPCS is not specified, the error message will be,

"CNEXPN REQUIRES SHORT PSEUDO-COMPUTE SEQUENCE"

If the dynamic storage allocation is not sufficient NDIM < (NND + NNA)), the message will be,

"_____ LOCATIONS AVAILABLE"

Note that the number printed will be negative indicating the additional storage locations required.

If the time-step used is less than the time-step allowed (DTIMEL) which may be optionally specified by the user, the message will be,

"TIME STEP TØØ SMALL"

If CSGMIN $\leq$ 0, the message printed will be,

"CSGMIN ZERØ or NEGATIVE"

Checks on the control constants, the pseudo-compute sequence and the dynamic storage allocation are made in the following sequence with the run terminating if a single check is not satisfied,

ØUTPUT, pseudo-compute sequence, dynamic storage locations.

It should be particularly noted that <u>no message</u> is printed if ARLXCA is not satisfied with NLØØP iterations; ARLXCA and NLØØP are optionally specified control constants.

Table 6.3-3. Basic Computational Steps for CNEXPN

1. Specification of control constants. Control constants ØUTPUT and TIMEND must be specified. SPCS is required for CNEXPN. (Refer to Table 6.2-5 for values and Section 6.2.3.2 for description.)

2. Sufficiency check on dynamic storage. Requirements = NND + NNA (NND = diffusion nodes and NNA = arithmetic nodes).

3. Setting and/or calculation of time-step, Δt. (Refer to Section 6.2.4 for detailed procedure.) Calculated time-step = 0.95 * CSGMIN * CSGFAC.

4. Setting of source and diffusion node dynamic storage locations to zero.

5. Calling of VARIABLES 1. (Refer to Section 6.2.2.2 for description.)

6. Checking of BACKUP. (Refer to Section 6.2.3.2 for description.)

7. Calculation of diffusion-node temperatures. (Refer to Section 6.2.5.1 for description.)

   Diffusion-node temperatures are calculated by using (refer to Section 6.3.3.2),

$$T_{i,n+1} = T_{i,n} e^{-\alpha_n \Delta t} + \frac{q_{i,n} + \sum_{j=1}^{p} G_{ij,n} T_{j,n}}{\sum_{j=1}^{p} G_{ij,n}} \left(1 - e^{-\alpha_n \Delta t}\right)$$

   where,

$$\alpha_n = \frac{\sum_{j=1}^{p} G_{ij,n}}{C_{i,n}}$$

8. Erasure of all temperature calculations for latest time-step if allowable temperature change criterion DTMPCA is not satisfied and recalculation of temperatures with reduced time-step.

9. Calculation of arithmetic-node temperatures. If the number of iterations equal NLØØP, the temperatures are retained without user notification

10. Erasure of all temperature calculations for latest time-step if allowable temperature change criterion ATMPCA is not satisfied and recalculation of temperatures with reduced time-step.

11. Calling of VARIABLES 2 and checking of BACKUP. (Refer to Section 6.2.2.3 and 6.2.3.2 for description.)

12. Advancing of time, checking of time to print, and the printing at the output interval.

13. Calling of ØUTPUT CALLS.

14. Checking for problem end time stored in user specified control constant TIMEND.

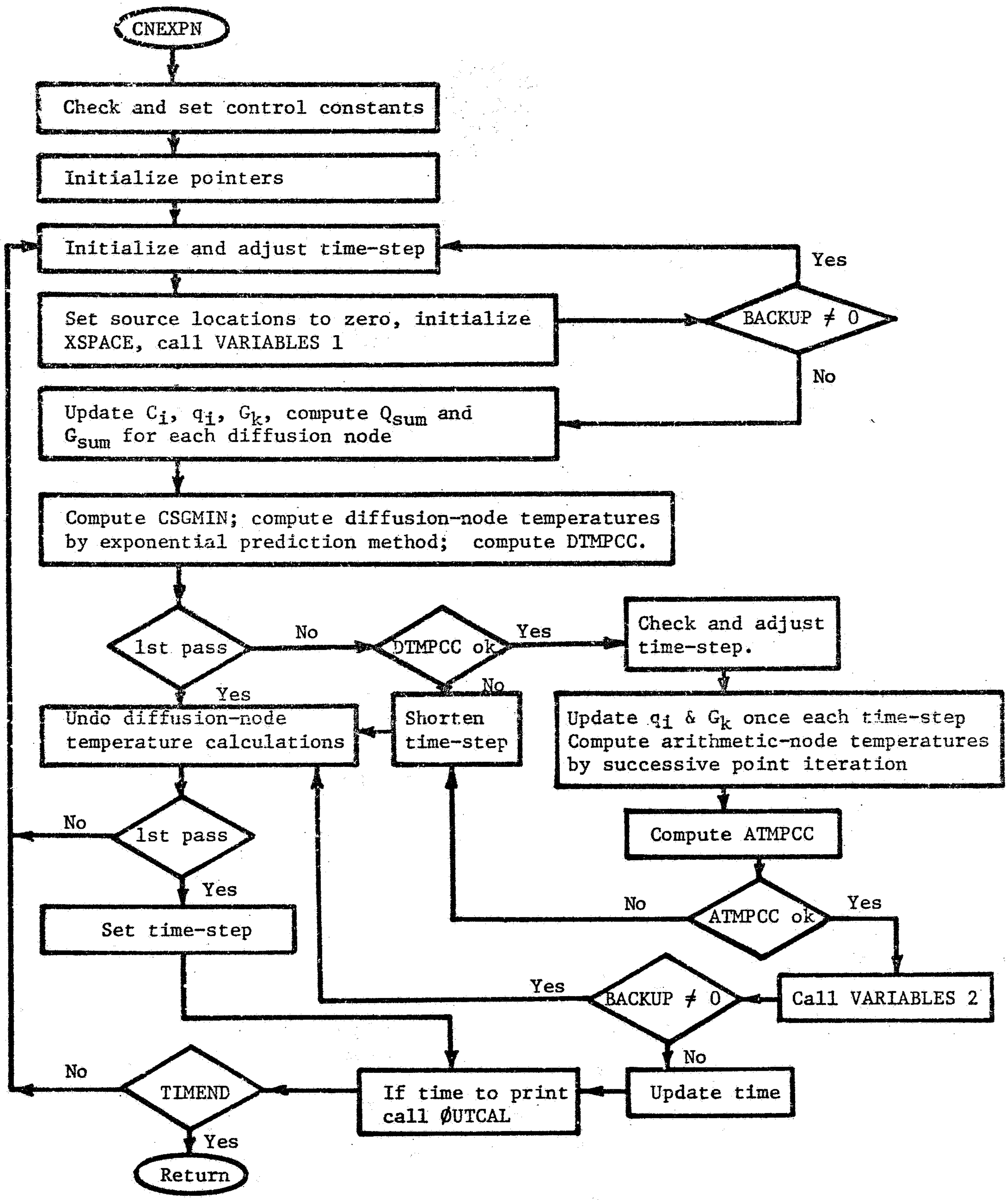


Figure 6.3-5. Functional Flow Chart for CNEXPN

6.3.4 Subroutine: CNDUFR

6.3.4.1 General Comments

Subroutine CNDUFR is an explicit numerical solution routine that uses an unconditionally stable DuFort-Frankel method.[9, 12, 17] The DuFort-Frankel method replaces the present temperature of the node being operated on by the average of future and past temperatures in the forward differencing equation. In subroutine CNDUFR the present temperature of the node being operated on is replaced by a time-weighted average of future and past temperatures. CNDUFR requires the short pseudo-compute sequence (SPCS).

The intent of an unconditionally stable routine such as CNDUFR is the reduction of computational time by using time-steps greater than those allowed with the conditionally stable explicit methods as constrained by the stability criterion. However, less accuracy can be expected with a lengthened time-step. The time-step controlled with control constant CSGFAC represents a user decision that is difficult and must be aided by a trial and error procedure.

Examination of several CNDUFR solutions reveals a tendency to lead in time the true temperatures.

6.3.4.2 Finite Difference Approximation and Computational Algorithm

Diffusion Nodes

The DuFort-Frankel explicit finite difference expression[9, 12, 17] for calculating the diffusion-node temperatures may be readily determined as follows:

Using the standard explicit finite difference expression,

$$C_i \frac{(T_{i,n+1} - T_{i,n})}{\Delta t} = q_{i,n} + \sum_{j=1}^{p} G_{ij,n} \, (T_{j,n} - T_{i,n}) \tag{6.3-8}$$

$$i = 1,2,\ldots,N$$

$$T_{j,n} = \text{constant}, N < j \leq p$$

letting the present temperature, $T_{i,n}$, be replaced by the average of future temperature, $T_{i,n+1}$, and past temperature, $T_{i,n-1}$,

$$T_{i,n} = \frac{T_{i,n+1} + T_{i,n-1}}{2} \tag{6.3-9}$$

where, $\Delta t_i = \Delta t_{i+1}$, $i = 1,2,\ldots,M$ (equal time-steps)

and defining,

$$\bar{C}_i = C_i/\Delta t \text{ (refer to Section 6.2.4 for discussion on } \Delta t) \qquad (6.3\text{-}10)$$

equation (6.3-8) can be expressed as,

$$T_{i,n+1} = \frac{\bar{C}_{i,n}\, T_{i,n-1} + 2\, q_{i,n} + \sum_{j=1}^{p} G_{ij,n}\,(2\, T_{j,n} - T_{i,n-1})}{\bar{C}_{i,n} + \sum_{j=1}^{p} G_{ij,n}} \qquad (6.3\text{-}11)$$

$$i = 1,2,\ldots,N$$

In CNDUFR the present temperature, $T_{i,n}$, of equation (6.3-8) is replaced by a weighted average of future temperature, $T_{i,n+1}$, and past temperature, $T_{i,n-1}$. The weighting is based on unequal time-steps.

$$T_{i,n} = \frac{(\Delta t_{n-1}\, T_{i,n+1} + \Delta t_n\, T_{i,n-1})}{\Delta t_{n-1} + \Delta t_n} \qquad (6.3\text{-}12)$$

where, $\Delta t_{n-1} = t_n - t_{n-1}$ (past time-step)

$\Delta t_n = t_{n+1} - t_n$ (present time-step)

Let

$$\tau_{n-1} = \frac{\Delta t_{n-1}}{\Delta t_{n-1} + \Delta t_n} \qquad (6.3\text{-}13)$$

$$\tau_n = \frac{\Delta t_n}{\Delta t_{n-1} + \Delta t_n} \qquad (6.3\text{-}14)$$

Equation (6.3-8) becomes,

$$T_{i,n+1} = \frac{\tau_n\, T_{i,n-1}\left(\bar{C}_{i,n} - \sum_{j=1}^{p} G_{ij,n}\right) + \sum_{j=1}^{p} G_{ij,n}\, T_{j,n} + q_{i,n}}{\bar{C}_{i,n} - \tau_{n-1}\left(\bar{C}_{i,n} - \sum_{j=1}^{p} G_{ij,n}\right)} \qquad (6.3\text{-}15)$$

where, $i = 1,2,\ldots,NND$ (number of diffusion nodes)

$T_{j,n}$ = constant, $(NND + NNA) < j \leq p$ (NNA is the number of arithmetic nodes and p is the total number of nodes)

$$G_{ij,n} = a_{ij,n} + \sigma b_{ij,n}\,(T_{j,n}^2 + T_{i,n}^2)(T_{j,n} + T_{i,n})$$

C_i, q_i, a_{ij}, b_{ij} = may be optionally specified (refer to Tables 6.2-1 - 6.2-4)

In CNDUFR, equation (6.3-15) is applied to the diffusion nodes with the computational procedure being a "block" change in temperature from one time-step to another.

Arithmetic Nodes (if any)

Arithmetic-node temperatures are calculated in the same manner as in CNFRWD (Section 6.3.1.2) or refer to Section 5.2.3 for the finite difference algorithm.

6.3.4.3 Comments on the Computational Procedure

The important steps of the computation procedure used in subroutine CNDUFR are indicated in Table 6.3-4 and a functional flow chart is shown in Figure 6.3-6. A computer listing of CNDUFR is found in Appendix A but some general computational details are given in Section 6.2.5.1. The computational procedure for CNDUFR follows the CNEXPN computational pattern, but with the temperatures of the diffusion nodes calculated by the DuFort-Frankel method of the exponential prediction method. Another significant difference is that CNDUFR must provide for two sets of past temperatures which are required for DuFort-Frankel algorithm; two time-steps for consecutive time-step calculations are also required. Otherwise, checks and control constant use are identical to CNEXPN. Thus, the verbal flow description of Section 6.3.1.3 applies directly except for the differences indicated above.

6.3.4.4 Control Constants

Control constants ØUTPUT and TIMEND (> TIMEØ) must be specified as indicated in Table 6.2-5 and described in Section 6.2.3.2; otherwise the "run" will terminate with an error message. The function of optionally specified control constants ARLXCA, ATMPCA, BACKUP, CSGFAC, DAMPA, DTIMEH, DTIMEL, DTMPCA, NLØØP, and TIMEØ is described in Section 6.2.3.2. The user should take particular care in the selection of CSGFAC since too large of a time-step would lead to grossly inaccurate temperatures even though the solution is stable. Note also that TIMEØ may be set negative and that NLØØP is set to one if not specified.

6.3.4.5 Error and Other Messages

If control constants ØUTPUT and TIMEND are not specified, the following error message will be printed for each,

ØUTPUT	"NØ ØUTPUT INTERVAL"
TIMEND	"TIME STEP TØØ SMALL"

The reason for the TIMEND error message is that a direct check on TIMEND is not made; the resultant error message just happens to be a quirk in the coding.

If the short pseudo-compute sequence SPCS is not specified, the error message will be,

"CNDUFR REQUIRES SHORT PSEUDO-COMPUTE SEQUENCE"

If the dynamic storage allocation is not sufficient (NDIM < (2*NND + NNA)), the message will be,

"_____ LOCATIONS AVAILABLE"

Note that the number printed will be negative indicating the additional storage locations required.

If the time-step used is less than the time-step allowed (DTIMEL), which may be optionally specified by the user, the message will be,

"TIME STEP TØØ SMALL"

If CSGMIN $\leq$ 0, the message printed will be,

"CSGMIN ZERØ or NEGATIVE"

Checks on the control constants, the pseudo-compute sequence and the dynamic storage allocation are made in the following sequence with the run terminating if a single check is not satisfied,

ØUTPUT, pseudo-compute sequence, dynamic storage locations

It should be particularly noted that no message is printed if ARLXCA is not satisfied with NLØØP iterations; ARLXCA and NLØØP are optionally specified control constants.

Table 6.3-4. Basic Computational Steps for CNDUFR

1. Setting of control constants to nominal values. Control constants ØUTPUT and TIMEND must be specified. SPCS is required for CNDUFR. (Refer to Table 6.2-5 for values and Section 6.2.3.2 for description.)

2. Sufficiency check on dynamic storage. Requirements = 2*NND + NNA (NND = diffusion nodes and NNA = arithmetic nodes).

3. Setting and/or calculation of time-step, Δt. (Refer to Section 6.2.4 for detailed procedure.) Calculated time-step = 0.95*CSGMIN*CSGFAC.

4. Setting of source and diffusion node dynamic storage locations to zero.

5. Calling of VARIABLES 1. (Refer to Section 6.2.2.2 for description.)

6. Checking of BACKUP. (Refer to Section 6.2.3.2 for description.)

7. Calculation of diffusion-node temperatures. (Refer to Section 6.2.5.1 for description.)

   Diffusion-node temperatures are calculated by using (refer to Section 6.3.4.2),

$$T_{i,n+1} = \frac{\tau_n \, T_{i,n-1} \left(\overline{C}_{i,n} - \sum_{j=1}^{p} G_{ij,n}\right) + \sum_{j=1}^{p} G_{ij,n} \, T_{j,n} + q_{i,n}}{\overline{C}_{i,n} - \tau_{n-1} \left(\overline{C}_{i,n} - \sum_{j=1}^{p} G_{ij,n}\right)}$$

   where,

$$\tau_{n-1} = \frac{\Delta t_{n-1}}{\Delta t_{n-1} + \Delta t_n}$$

$$\tau_n = \frac{\Delta t_n}{\Delta t_{n-1} + \Delta t_n}$$

8. Erasure of all temperature calculations for latest time-step if allowable temperature change criterion DTMPCA is not satisfied and temperature recalculation with reduced time-step.

9. Calculation of arithmetic-node temperatures; if the number of iterations equal NLØØP, the temperatures are retained without user notification (refer to Section 6.2.5.1 for details).

10. Erasure of arithmetic-node temperatures for latest time-step if allowable temperature change criterion ATMPCA is not satisfied and temperature recalculation with reduced time-step.

11. Calling of VARIABLES 2 and checking of BACKUP. (Refer to Section 6.2.2.3 and 6.2.3.2 for description.)

12. Advancing of time, checking of time to print, and the printing at the output interval.

13. Calling of ØUTPUT CALLS.

14. Checking for problem end time stored in user specified control constant TIMEND.

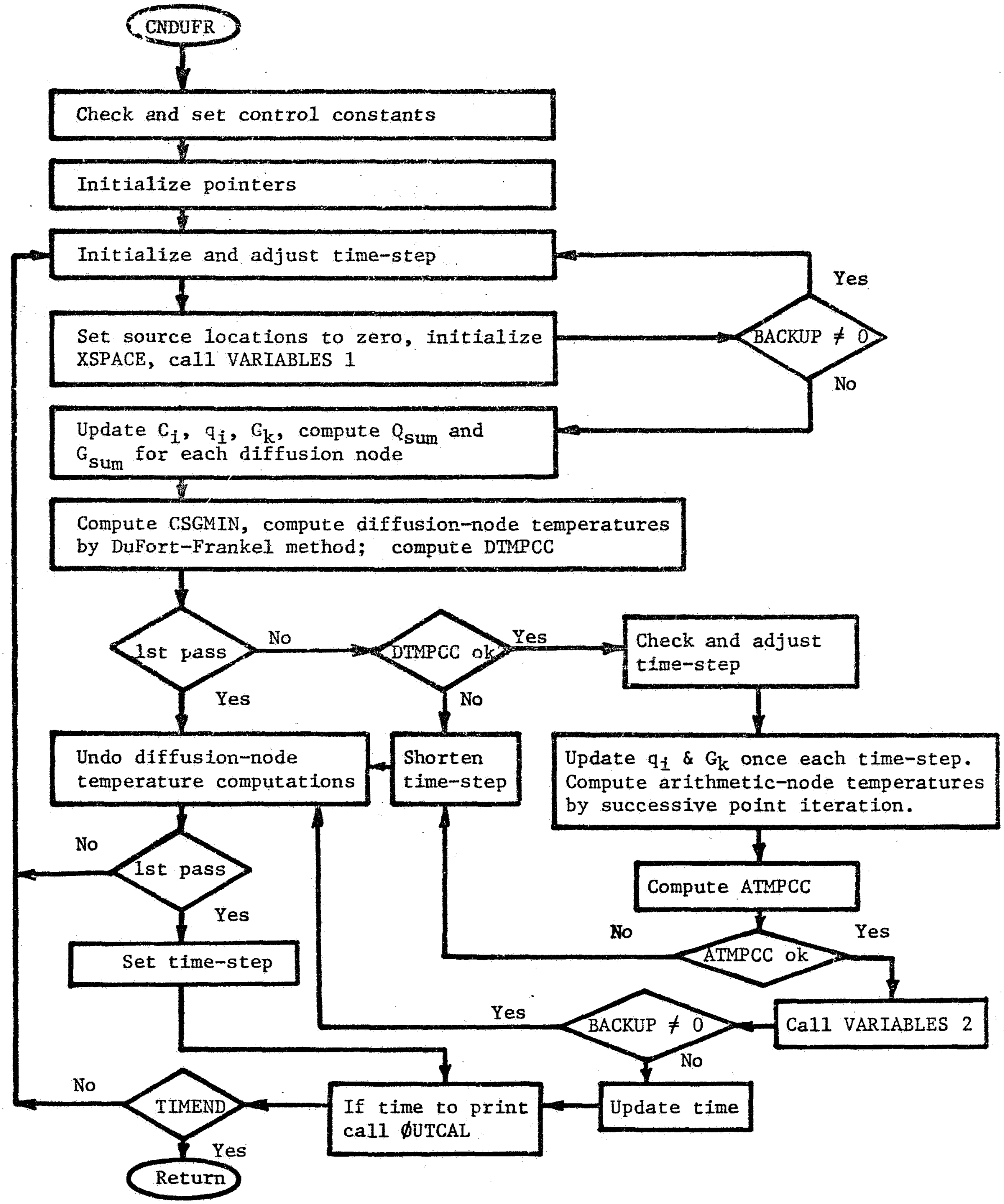


Figure 6.3-6. Functional Flow Chart for CNDUFR

6.3.5 Subroutine: CNQUIK

6.3.5.1 General Comments

Subroutine CNQUIK is a numerical solution routine that uses an algorithm composed of half DuFort-Frakel method[9, 12, 17] and half exponential prediction method.[1, 17] CNQUIK requires the short pseudo-compute sequence (SPCS); characteristics of subroutines CNDUFR and CNEXPN, as described in Section 6.3.3 and 6.3.4, also apply to CNQUIK.

Why CNQUIK? Examination of CNDUFR and CNEXPN solutions reveals that CNDUFR has a tendency to yield temperatures which lead the true temperatures, whereas CNEXPN has a tendency to lag the true temperatures. Thus, it was theorized that a combination of CNDUFR and CNEXPN should yield a more accurate solution than either one. Preliminary results indicate that CNQUIK is more accurate than either CNDUFR or CNEXPN with approximately the same solution time. It can also be theorized that a more accurate combination of the DuFort-Frankel and exponential prediction is probably possible than the half and half used in CNQUIK. However, a detailed study will be required before a realistic evaluation of CNQUIK can be made.

6.3.5.2 Finite Difference Approximation and Computational Algorithm

Diffusion Nodes

Subroutine CNQUIK uses a numerical solution algorithm composed of half DuFort-Frankel and half exponential prediction. That is the temperature of the diffusion nodes is calculated by using,

$$T_{i,n+1} = \left(T_{CNDUFR} + T_{CNEXPN}\right)/2.0 \qquad (6.3\text{-}16)$$

$$T_{CNDUFR} = \frac{\tau_n T_{i,n-1}\left(\overline{C}_{i,n} - \sum_{j=1}^{p} G_{ij,n}\right) + \sum_{j=1}^{p} G_{ij,n} T_{j,n} + q_{i,n}}{\overline{C}_{i,n}(1 - \tau_{n-1}) + \sum_{j=1}^{p} G_{ij,n}}$$

(equation 6.3-15 of Section 6.3.4.2)

$$T_{CNEXPN} = T_{i,n} e^{-\alpha_n \Delta t} + \frac{q_{i,n} + \sum_{j=1}^{p} G_{ij,n} T_{j,n}}{\sum_{j=1}^{p} G_{ij,n}}\left(1 - e^{-\alpha_n \Delta t}\right)$$

(equation 6.3-7 of Section 6.3.3.2)

n = nth time-step

i = 1,2,...,NND (number of diffusion nodes)

$T_{j,n}$ = constant, (NND + NNA) < $j \leq p$ (NNA is the number of arithmetic nodes and p is the total number of nodes)

$$\alpha_n = \frac{\sum_{j=1}^{p} G_{ij,n}}{C_{i,n}}$$

$$G_{ij,n} = a_{ij,n} + \sigma b_{ij,n} (T^2_{i,n} + T^2_{j,n})(T_{i,n} + T_{j,n})$$

$$\tau_n = \frac{\Delta t_n}{\Delta t_{n-1} + \Delta t_n} \; ; \quad \tau_{n-1} = \frac{\Delta t_{n-1}}{\Delta t_{n-1} + \Delta t_n}$$

C_i, q_i, a_{ij}, b_{ij} = optionally specified (refer to Tables 6.2-1 - 6.2-4)

$\bar{C}_i = C_i/\Delta t$ (refer to Section 6.2.4 for discussion of Δt)

Arithmetic Nodes (if any)

Temperatures of arithmetic nodes are calculated in the same manner as in CNFRWD (Section 6.3.1.2) or refer to Section 5.2.3 for the finite difference algorithm.

6.3.5.3 Comments on the Computational Procedure

The important steps of the computational procedure used in subroutine CNQUIK are indicated in Table 6.3-5 and a functional flow chart is shown in Figure 6.3-7. A computer listing of CNQUIK is found in Appendix A. General computational details are given in Section 6.2. The computational procedure for CNQUIK follows CNEXPN or CNDUFR with the diffusion-node temperatures calculated with the half DuFort-Frankel and half exponential prediction algorithm being the only difference. Arithmetic-node temperatures are calculated in the same manner as the other SINDA explicit routines. Note that the time-step is calculated as CSGMIN*CSGFAC and checks are the same as CNEXPN or CNDUFR. Thus, the verbal flow description of Section 6.3.1.3 applies directly except for the differences indicated above.

6.3.5.4 Control Constants

Control constants ØUTPUT and TIMEND (> TIMEØ) must be specified as indicated in Table 6.2-5 and described in Section 6.2.3.2; otherwise the "run" will terminate with an error message. The function of optionally specified control constants ARLXCA, ATMPCA, BACKUP, CSGFAC, DAMPA, DTIMEH, DTIMEL, DTMPCA, NLØØP, and TIMEØ is described in Section 6.2.3.2. Again, caution must be exercised in the selection of CSGFAC since too large of a time-step would lead to grossly inaccurate temperatures even though the solution is stable. Note also that TIMEØ may be set negative and that NLØØP is set to one if not specified.

6.3.5.5 Error and Other Messages

If control constants ØUTPUT and TIMEND are not specified, the following error message will be printed for each,

ØUTPUT	"NØ ØUTPUT INTERVAL"
TIMEND	"TIME STEP TØØ SMALL"

The reason for the TIMEND error message is that a direct check on TIMEND is not made; the resultant error message just happens to be a quirk in the coding.

If the short pseudo-compute SPCS is not specified, the error message will be,

"CNQUIK REQUIRES SHORT PSEUDO-COMPUTE SEQUENCE"

If the dynamic storage allocation is not sufficient (NDIM < (2*NND + NNA), the message will be,

"_____ LOCATIONS AVAILABLE"

Note that the number printed will be negative indicating the additional storage locations required.

If the time-step used is less than the time-step allowed (DTIMEL), which may be optionally specified by the user, the message will be,

"TIME STEP TØØ SMALL"

If CSGMIN $\leq$ 0, the message printed will be,

"CSGMIN ZERØ or NEGATIVE"

Checks on the control constants, the pseudo-compute sequence and the dynamic storage allocation are made in the following sequence with the run terminating if a single check is not satisfied,

ØUTPUT, pseudo-compute sequence, dynamic storage locations.

It should be particularly noted that no message is printed if ARLXCA is not satisfied with NLØØP iterations; ARLXCA and NLØØP are optionally specified control constants.

Table 6.3-5. Basic Computational Steps for CNQUIK

1. Specification of control constants. Control constants ØUTPUT and TIMEND must be specified. SPCS is required for CNEXPN. (Refer to Table 6.2-5 for values and Section 6.2.3.2 for description.)
2. Sufficiency check on dynamic storage. Requirements = 2(NND) + NNA (NND = diffusion nodes and NNA = arithmetic nodes).
3. Setting and/or calculation of time-step, Δt. (Refer to Section 6.2.4 for detailed procedure.) Calculated time-step = 0.95*CSGMIN*CSGFAC.
4. Setting of source and diffusion node dynamic storage locations to zero.
5. Calling of VARIABLES 1. (Refer to Section 6.2.2.2 for description.)
6. Checking of BACKUP. (Refer to Section 6.2.3.2 for description.)
7. Calculation of diffusion-node temperatures. (Refer to Section 6.2.5.1 for description.)

   Diffusion-node temperatures are calculated by using (refer to Section 6.3.5.2),

   $$T_{i,n+1} = (T_{CNDUFR} + T_{CNEXPN})/2.0$$

   (Refer to equation 6.3-17, Section 6.3.5.2.)
8. Erasure of all temperature calculations for latest time-step if allowable temperature change criterion DTMPCA is not satisfied and temperature recalculation with reduced time-step.
9. Calculation of arithmetic-node temperatures; if the number of iterations equal NLØØP, the temperatures are retained without user notification. (Refer to Section 6.2.5.1 for details.)
10. Erasure of all temperature calculations for latest time-step if allowable temperature change criterion ATMPCA is not satisfied and temperature recalculation with reduced time-step.
11. Calling of VARIABLES 2 and checking of BACKUP. (Refer to Section 6.2.2.3 and 6.2.3.2 for description.)
12. Advancing of time, checking of time to print, and the printing at the output interval.
13. Calling of ØUTPUT CALLS.
14. Checking for problem end time stored in user specified control constant TIMEND.

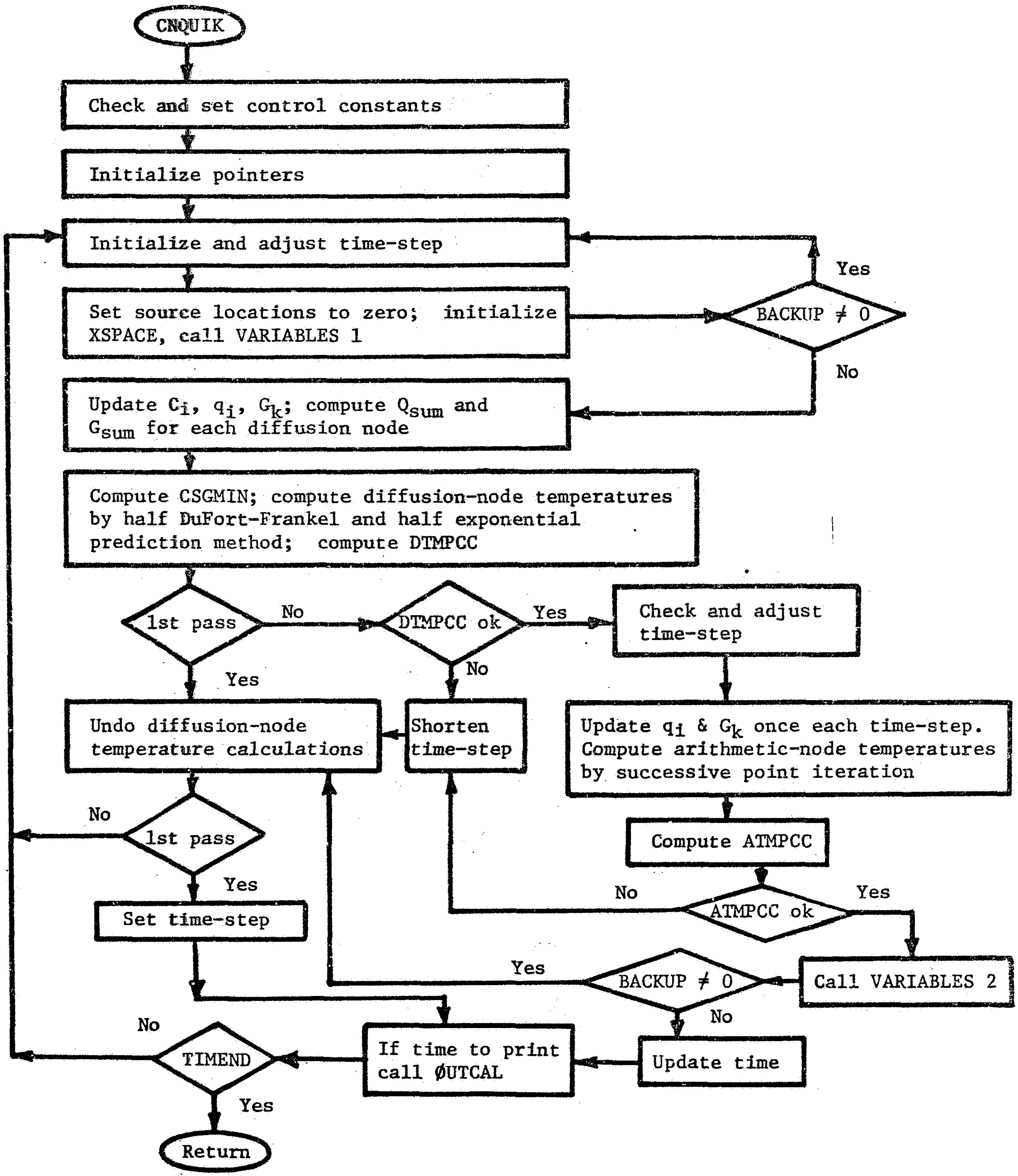


Figure 6.3-7. Functional Flow Chart for CNQUIK

6.4 Transient Implicit Solution Routines

SINDA implicit solution routines number three; these routines are identified as follows:

CNBACK Implicit backward difference method.
Requires long pseudo-compute sequence (LPCS).

CNFWBK Implicit forward-backward differencing, using Crank-Nicolson method.
Requires long pseudo-compute sequence (LPCS).

CNVARB Combination of CNBACK and CNFWBK.
Requires long pseudo-compute sequence (LPCS).

Implicit methods generally tend to be more accurate than explicit methods and are unconditionally stable as are some explicit methods. With implicit methods the time-step is specified in contrast to the calculated time-steps of explicit methods with their stability criterion. An important consideration in the use of implicit methods is that the time-step DTIMEI should be specified in conjunction with control constant NLØØP which represents the maximum number of computational iterations during each time-step. Since each iterative calculation is essentially equivalent to a time-step calculation for an explicit method, the combination of DTIMEI and NLØØP for a given time period should be set less than the total number of time-steps used by the explicit method for the same time period, if computational time is to be reduced; this of course assumes that during each time-step the maximum number of iterations is required. If the NLØØP iterations are required during a time-step, the temperature accuracy is affected but the magnitude would depend upon the value used for the maximum allowable relaxation temperature change criteria, ARLXCA and DRLXCA. It should be noted if NLØØP iterations are required during a time-step, the message "RELAXATION CRITERIA NOT MET" is printed.

A detailed description of each implicit routine, as presented on the pages to follow, relies on the general description of Section 6.2. A brief description of these routines is summarized first.

CNBACK uses the standard backward differencing algorithm and requires the long pseudo-compute sequence (LPCS). The time-step must be specified via control constant DTIMEI and used in conjunction with the control constant NLØØP. CNBACK uses the acceleration of convergence feature.

CNFWBK uses the Crank-Nicolson algorithm which is composed of half forward differencing and half backward differencing. CNFWBK solutions tend to be more accurate than CNBACK solutions with approximately 25% less iterations; however CNFWBK solutions have "blown" on occasions.

CNVARB uses a combination of forward differencing and backward differencing. Unlike CNFWBK which is half and half, CNVARB uses a variable beta factor which ranges from 0 to 1. Thus CNVARB uses a method that is somewhere between forward differencing and backward differencing.

6.4.1 Subroutine: CNBACK

6.4.1.1 General Comments

Subroutine CNBACK is an implicit routine that uses the standard backward difference expression and requires the long pseudo-compute (LPCS). Time-step must be specified via DTIMEI otherwise the "run" will terminate with an error message printout. The time-step value is arbitrary but the user should consider DTIMEI in conjunction with the control constant NLØØP which represents the maximum number of computational iterations during each time-step (refer to Section 6.2.3.2 for description).

Implicit methods tend to be more accurate than explicit methods and are unconditionally stable, but implicit solutions often oscillate at start up or boundary step changes when heat transfer by radiation is present. CNBACK internally controls sudden radiation heat transfer changes by an averaging technique which is termed "radiation damping" (refer to Section 6.2.6 for details). This automatic damping has been very effective in many solutions that have been examined and lessens the need for the use of DAMPD and DAMPA.

6.4.1.2 Finite Difference Approximation and Computational Algorithm

The numerical solution algorithm used in subroutine CNBACK is the standard backward-difference expression[12,13,17] which may be expressed as:

$$C_i \frac{(T_{i,n+1} - T_{i,n})}{\Delta t} = q_{i,n} + \sum_{j=1}^{p} a_{ij} (T_{j,n+1} - T_{i,n+1}) + \sum_{j=1}^{p} \sigma b_{ij} (T^4_{j,n+1} - T^4_{i,n+1})$$

(equation 5.2-5 of Section 5.2.2)

$i = 1,2,\ldots,N$

$T_{j,n+1} = \text{constant}, \; N < j \le p$

$T_{i,n} \equiv T_i(n\Delta t)$

The computational procedure for the backward difference formulation must necessarily be re-iterative because of the need to solve a set of simultaneous nonlinear equations.

Diffusion Nodes

Diffusion node temperatures are solved by "successive point" iteration but differs from the arithmetic-node temperature calculation because of the capacitance term and the use of "radiation damping" (refer to Section 6.2.5.2).

$$T_{i,k+1} = DD^* \, T_{i,k}$$

$$+ \, DN^* \frac{\overline{C}_{i,n} \, T_{i,n} + q_{i,n} + \sum_{j=1}^{i} G_{ij,n} \, T_{j,k+1} + \sum_{j=i+1}^{p} G_{ij,n} \, T_{j,k} - (q_i)_{ave}}{\overline{C}_{i,n} + \sum_{j=1}^{p} a_{ij,n}} \tag{6.4-1}$$

where, $i = 1,2,\ldots,NND$

n = nth time-step

k = kth iteration

DN = DAMPD (diffusion-node damping factor)

$DD = 1.0 - DN$

$G_{ij,n} = a_{ij,n} + \sigma b_{ij,n} \, T^3_{j,\ell}$ ($\ell = k$ if $j \geq i$ and $\ell = k+1$ if $j < i$)

C_i, q_i, a_{ij}, b_{ij} = optionally specified (refer to Tables 6.2-1 - 6.2-4)

$\overline{C}_{i,n} = C_{i,n}/\Delta t$ (Δt = time step, refer to Section 6.2.4)

$(q_i)_{ave} = \sum_{j=1}^{p} \sigma b_{ij,n} \, [(T^4_{i,k}) + (T^4_{i,k})_2]/2.0$, average heat loss from the ith node (refer to Section 6.2.6 on radiation damping for details)

Details on the computational procedure for implicit routines are presented in Sections 5.2.2 and 6.2.5.2.

Arithmetic Nodes

Arithmetic-node temperatures are calculated identically the same in all the SINDA numerical solution routines. Thus, refer to either Section 6.3.1.2 or Section 6.2.5.2 for the finite difference algorithm.

6.4.1.3 Comments on the Computational Procedure

The important steps of the computational procedure used in subroutine CNBACK are indicated in Table 6.4-1. For a detailed step-by-step computational description, the user must examine the computer listing for

CNBACK in Appendix B, but some general computational details are given in Section 6.2.5.2. A functional flow chart of CNBACK is shown in Figure 6.4-1.

Both diffusion-node temperatures and arithmetic-node temperatures are calculated by "successive point" iteration. Each third iteration, diffusion-node temperatures which are decreasing over two time-steps are extrapolated in an attempt to accelerate convergence (refer to Section 6.2.7). Temperature convergence is examined during each time-step by checking DRLXCC and ARLXCC against the user control constants DRLXCA (for diffusion nodes) and ARLXCA (for arithmetic nodes), respectively. If temperatures have not converged with NLØØP iterations, the message "RELAXATION CRITERIA NOT MET" is printed. Control constant NLØØP is used to specify the maximum number of iterations allowed during each time-step.

VARIABLES 1 and VARIABLES 2 are performed only once for each time-step. Since this subroutine is implicit, the user must specify the time-step to be used through the control constant DTIMEI in addition to control constant TIMEND and ØUTPUT. The look ahead feature for the time-step calculation used in CNFRWD is also employed in CNBACK as are checks for maximum allowable time-step DTIMEH, maximum allowable temperature change between time-steps, DTMPCA (diffusion nodes) and ATMPCA (arithmetic nodes). The minimum time-step DTIMEL is not checked however. Damping of solutions can be achieved through the use of the control constants DAMPD and DAMPA but "radiation damping" (refer to Section 6.2.6) used by CNBACK lessens the need for the damping factors DAMPD and DAMPA.

6.4.1.4 Control Constants

Control constants ARLXCA, DRLXCA, DTIMEI, NLØØP, ØUTPUT, and TIMEND must be specified as indicated in Table 6.2-5 and as described in Section 6.2.3.2; otherwise "run" will terminate with an appropriate error message. The function of optionally specified control constants ATMPCA, BACKUP, DAMPA, DAMPD, DTIMEH, DTMPCA, and TIMEØ is described in Section 6.2.3.2.

Specification of time-step DTIMEI should be done in conjunction with control constant NLØØP which represents the maximum number of computational iterations during each time-step. Since each iterative calculation is essentially equivalent to a time-step calculation for an explicit method, the combination of DTIMEI and NLØØP for a given time period should be less than the total number of time-steps by the explicit method for the same period. Note also that TIMEØ may be set negative. Specification of ARLXCA and DRLXCA depends upon the problem but a typical value is 0.1.

6.4.1.5 Error and Other Messages

If control constants ARLXCA, DRLXCA, DTIMEI, NLØØP, ØUTPUT and TIMEND are not specified, the following error message will be printed for each,

ARLXCA	"NØ ARLXCA"
DRLXCA	"NØ DRLXCA"
DTIMEI	"NØ DTIMEI"
NLØØP	"NØ NLØØP"
ØUTPUT	"NØ ØUTPUT INTERVAL"
TIMEND	"TRANSIENT TIME NØT SPECIFIED"

If the long pseudo-compute sequence LPCS is not specified, the error message will be,

"CNBACK REQUIRES LONG PSEUDO-COMPUTE SEQUENCE"

If the dynamic storage allocation is not sufficient (NDIM < (3*NND + NNA + NNB)), the message will be,

"_____ LOCATIONS AVAILABLE"

Note that the number printed will be negative indicating the additional storage locations required.

If CSGMIN $\leq$ 0, the following message will be printed,

"CSGMIN ZERØ or NEGATIVE"

If either ARLXCA or DRLXCA is not satisfied with NLØØP iterations, the following message will be printed,

"RELAXATIØN CRITERIA NØT MET"

Checks on the control constants, the pseudo-compute sequence and the dynamic storage allocation are made in the following sequence with the "run" terminating if a single check is not satisfied,

NLØØP, TIMEND, ØUTPUT, ARLXCA, DTIMEI, DRLXCA, LPCS and dynamic storage allocation.

Table 6.4-1. Basic Computational Steps for CNBACK

1. Specification of control constants. Control constants ARLXCA (if NNA > 0), DRLXCA (if NND > 0), DTIMEI, NLØØP, ØUTPUT and TIMEND (TIMEND $\geq$ TIMEØ) must be specified. LPCS is required. (Refer to Table 6.2-5 for values and Section 6.2.3.2 for description.)

2. Sufficiency check on dynamic storage. Requirements = 3*NND + NNA + NNB (NND = diffusion nodes, NNA = arithmetic nodes and NNB = boundary nodes).

3. Setting and/or calculation of time-step, Δt. (Refer to Section 6.2.4 for detailed procedure.) Time-step = DTIMEI.

4. Setting of iterative DØ loop, 1 to NLØØP.

5. Setting of source locations to zero.

6. Calling of Variables 1. (Refer to Section 6.2.2.2 for description.)

7. Checking of BACKUP. (Refer to Section 6.2.3.2 for description.)

8. Diffusion-node temperature calculations, first iteration only.

   Evaluation of q_i, C_i and G_k.

   Damping of radiation heat transfer. (Refer to Section 6.2.5.2.)

   Calculation of diffusion-node temperature.

   The computational algorithm depends upon the presence of radiation heat transfer, but the method of solution is the standard implicit algorithm (refer to Section 6.2.5.2).

9. Conversion of $T_{i,k+1}$ to degrees Rankine.

10. Diffusion-node temperature calculations, successive iterations after first.

    Repeating of step 8, except that q_i, C_i and G_k are not updated.

    Calculation of DRLXCC.

11. Acceleration of convergence every third iteration if linear extrapolation is met (refer to Section 6.2.7).

12. Conversion of $T_{i,k+1}$ to degrees Fahrenheit.

13. Calculation of arithmetic-node temperatures, second and succeeding iterations; arithmetic-node temperatures are not calculated on the first iteration (refer to Section 6.2.5.2 for details).

14. Conversion of temperatures to degrees Rankine.

15. Checking of ARLXCA and DRLXCA for convergence and ØPEITR for output. If both ARLXCA and DRLXCA are satisfied, iterations during a time-step ceases, otherwise NLØØP iterations are performed.

16. Checking of ATMPCA and DTMPCA. If either one is not satisfied time-step is shortened, previous temperatures erased, and temperatures recalculated for shortened time-steps (refer to Section 6.2.5.2).

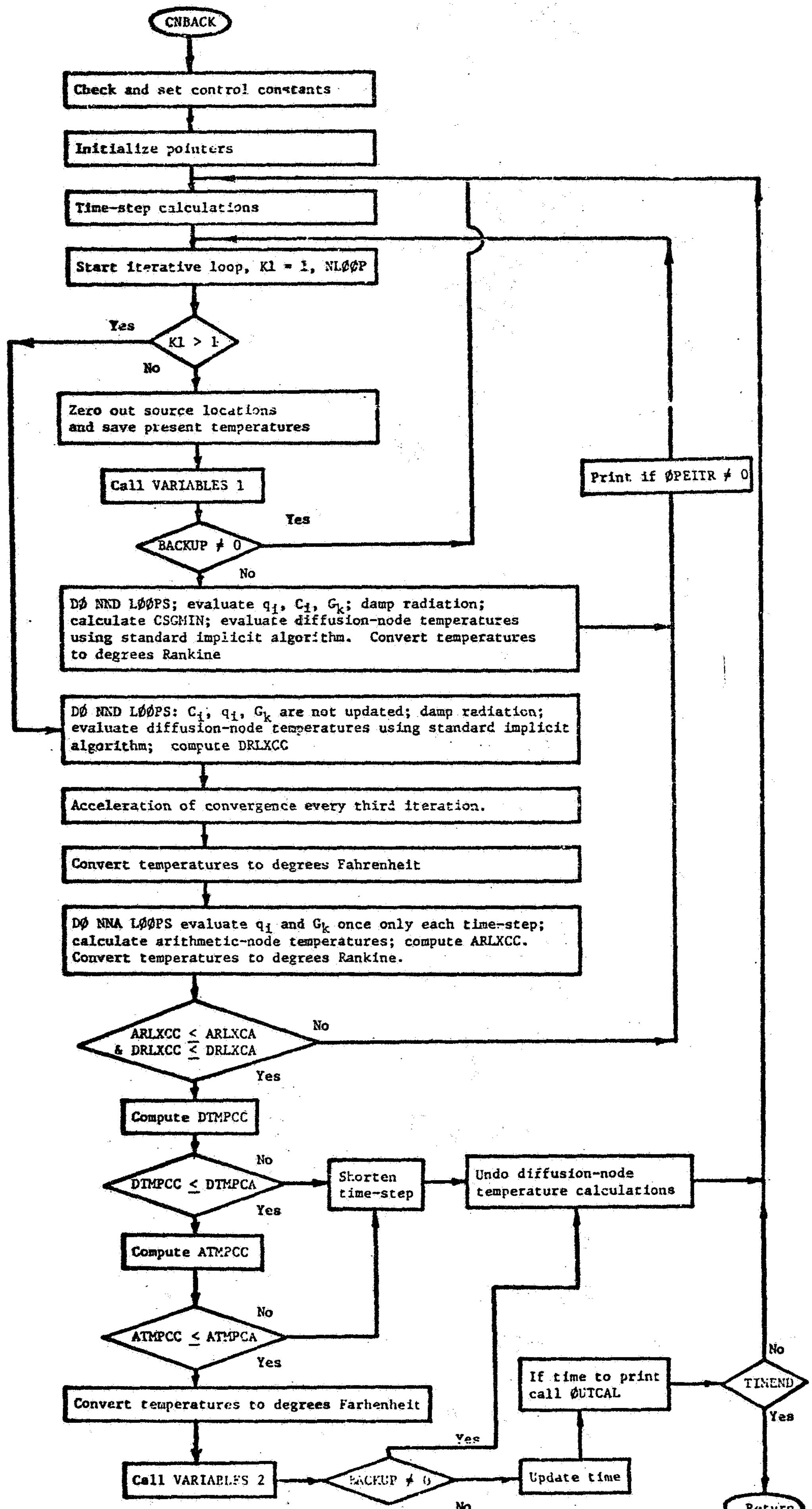
17. Conversion of temperatures back to degrees Fahrenheit.

18. Calling of VARIABLES 2 and checking of BACKUP (refer to Section 6.2.2.3 and 6.2.3.2).

19. Advancing of time, checking of time to print, and the printing of the output interval.

20. Calling of ØUTPUT CALLS.

21. Checking for problem end time stored in control constant TIMEND.



6.4-1. Functional Flow Chart for CNBACK

6.4.2 Subroutine: CNFWBK

6.4.2.1 General Comments

Subroutine CNFWBK is an implicit numerical solution routine that uses the Crank-Nocolson algorithm.[7, 8, 12] The long pseudo-compute sequence (LPCS) is required and the nodal temperatures (both diffusion and arithmetic) are solved by "successive point" iterations. The iteration looping, convergence criteria and other control constant checks are identical to CNBACK. Time-step must be specified via control constant DTIMEI. Diffusion and arithmetic temperature calculations may be damped through the use of DAMPD and DAMPA, respectively. Thermal radiation heat transfer is uniquely "handled" via a so-called "radiation damping" (refer to Section 6.2.6), and acceleration of convergence (refer to Section 6.2.7) is also available in CNFWBK.

CNFWBK solutions which are based on a half forward differencing and a half backward differencing method tend to be more accurate than CNBACK solutions with approximately the same solution time.

6.4.2.2 Finite Difference Approximation and Computational Algorithm

The numerical solution algorithm used in subroutine CNFWBK is the Crank-Nicolson method, which is half forward differencing and half backward differencing, and may be expressed as:

$$C_i \frac{(T_{i,n+1} - T_{i,n})}{\Delta t} = \frac{1}{2} (T_{forward} + T_{backward}) \qquad (6.4\text{-}2)$$

$$T_{forward} = q_{i,n} + \sum_{j=1}^{p} a_{ij,n}(T_{j,n} - T_{i,n}) + \sum_{j=1}^{p} \sigma b_{ij,n}(T^4_{j,n} - T^4_{i,n}) \qquad (6.4\text{-}3)$$

$$T_{backward} = q_{i,n} + \sum_{j=1}^{p} a_{ij,n} (T_{j,n+1} - T_{i,n+1}) + \sum_{j=1}^{p} \sigma b_{ij,n} (T^4_{j,n+1} - T^4_{i,n+1}) \qquad (6.4\text{-}4)$$

n = nth time-step

$i = 1,2,\ldots,N$

p = total number of nodes

$T_{j,n};\ T_{j,n+1}$ = constant, $N < j \leq p$

The computational procedure for the forward-backward difference formulation must be re-iterative because of the need to solve a set of simultaneous nonlinear equations. The pattern of computation is very similar to that used in CNBACK.

Diffusion-Nodes

Diffusion node temperatures are solved by "successive point" iteration but the algorithm differs from the algorithm used in CNBACK because of the additional terms arising from the forward difference portion of the expression.

$$T_{i,k+1} = DD^* \; T_{i,k} + DN^* \; [Q_{sum} - (q_i)_{ave}]/G_{sum} \tag{6.4-5}$$

where,

$$Q_{sum} = Q_i + \sum_{j=1}^{i} a_{ij,n} \; T_{j,k+1} + \sum_{j=i+1}^{p} a_{ij,n} \; T_{j,k} + \sum_{j=1}^{i} \sigma b_{ij,n} \; T^4_{j,k+1} + \sum_{j=i+1}^{p} \sigma b_{ij,n} \; T^4_{j,k} \tag{6.4-6}$$

$$Q_i = 2 \; q_{i,n} + 2 \; \overline{C}_{i,n} \; T_{i,n} + \sum_{j=1}^{p} a_{ij,n} \; (T_{j,n} - T_{i,n}) \tag{6.4-7}$$

$$G_{sum} = 2 \; \overline{C}_{i,n} + \sum_{j=1}^{p} a_{ij,n} \tag{6.4-8}$$

n = nth time-step; k = kth iteration

C_i, q_i, a_{ij}, b_{ij} = optionally specified (refer to Tables 6.2-1 - 6.2-4)

DN = DAMPD (diffusion-node damping factor)

DD = 1.0 - DN

$\overline{C}_{i,n} = C_{i,n}/\Delta t$ (Δt = time-step)

$(q_i)_{ave} = \Sigma \; \sigma b_{ij,n} \; [(T^4_{i,k}) + (T^4_{i,k})_2]/2.0$, average heat loss from ith node (refer to Section 6.2.6 on radiation damping for details)

(Note that the known quantities at time-step, n, are indicated by Q_i, equation 6.4-7.)

Arithmetic Nodes

Arithmetic-node temperatures are calculated identically the same in all the SINDA numerical solution routines. Thus, refer to Section 6.3.1.2 or Section 6.2.5.2 for the finite difference algorithm.

6.4.2.3 Comments on the Computational Procedure

The important steps of the computational procedure used in subroutine CNFWBK are indicated in Table 6.4-2. For a detailed step-by-step computational description, the user must examine the computer listing for CNFWBK in Appendix B, but some general computational details are given in Section 6.2.5.2. A functional flow chart of CNFWBK is shown in Figure 6.4-2.

The computational flow pattern for CNFWBK is identical to CNBACK with the only difference between the routines being the diffusion-node temperature finite-difference algorithm. On the first iteration only the source locations zeroed out and the present temperatures stored, VARIABLES 1 is called and variable C_i, impressed source q_i and variable coefficients G_i (diffusion-diffusion and diffusion-arithmetic) evaluated. All quantities which are evaluated at time, t_n, are summed in accordance with equations (6.4-6) and (6.4-8). CSGMIN is evaluated and the diffusion-node temperatures calculated; note the arithmetic-node temperatures are not calculated on the first iteration.

On the second and succeeding iterations the quantities C_i, q_i and G_k (diffusion-diffusion and diffusion-arithmetic) are not updated. Diffusion-node temperatures are calculated and DRLXCC determined. Every third iteration, if a diffusion-node temperature is converging, a linear extrapolation to accelerate convergence is performed (refer to Section 6.2.7). If arithmetic nodes are encountered, the appropriate q_i and G_k (for arithmetic nodes) are evaluated once per time-step. Arithmetic-node temperatures are calculated and ARLXCC determined.

Control constants DRLXCC and ARLXCC are checked against DRLXCA and ARLXCA, respectively each time-step; if both criteria are satisfied the iterations cease, otherwise the iterations continue NLØØP times and the message "RELAXATION CRITERIA NOT MET" is printed.

Diffusion-node and arithmetic-node temperature changes between time-steps are calculated and stored in DTMPCC and ATMPCC, respectively.

If DTMPCC > DTMPCA or if ATMPCC > ATMPCA, the just completed calculations are erased and the time-step shortened (refer to Section 6.2.5.2).

6.4.2.4 Control Constants

The control constants for CNFWBK are used in exactly the same way as used in CNBACK. Control constants ARLXCA, DRLXCA, DTIMEI, NLØØP, ØUTPUT, and TIMEND must be specified as indicated in Table 6.2-5 and as described in Section 6.2.3.2; otherwise "run" will terminate with an appropriate error message. The function of optionally specified control constants ATMPCA, BACKUP, DAMPA, DAMPD, DTIMEH, DTMPCA, and TIMEØ is described in Section 6.2.3.2.

Specification of time-step DTIMEI should be done in conjunction with control constant NLØØP which represents the maximum number of computational iterations during each time-step. Since each iterative calculation is essentially equivalent to a time-step calculation for an explicit method, the combination of DTIMEI and NLØØP for a given time period should be less than the total number of time-steps by the explicit method for the same time period. Note also that TIMEØ may be set negative. Specification of ARLXCA and DRLXCA depends upon the problem but a typical value is 0.1.

6.4.2.5 Error and Other Messages

If control constants ARLXCA, DRLXCA, DTIMEI, NLØØP, ØUTPUT and TIMEND are not specified the following error message will be printed for each,

ARLXCA	"NØ ARLXCA"
DRLXCA	"NØ DRLXCA"
DTIMEI	"NØ DTIMEI"
NLØØP	"NØ NLØØP"
ØUTPUT	"NØ ØUTPUT INTERVAL"
TIMEND	"TRANSIENT TIME NØT SPECIFIED"

If the long pseudo-compute sequence LPCS is not specified, the error message will be,

"CNFWBK REQUIRES LØNG PSEUDØ-CØMPUTE SEQUENCE"

If the dynamic storage allocation is not sufficient (NDIM < (3* NND + NNA + NNB)), the message will be,

"_____ LOCATIONS AVAILABLE"

Note that the <u>number presented will be negative</u> indicating the additional storage locations required.

If CSGMIN $\leq$ 0, the following message will be printed,

"CSGMIN ZERØ OR NEGATIVE"

If either ARLXCA or DRLXCA is not satisfied with NLØØP iterations, the following message will be printed,

"RELAXATIØN CRITERIA NØT MET"

Checks on the control constants, the pseudo-compute sequence and the dynamic storage allocation are made in the following sequence with the "run" terminating if a single check is not satisfied,

NLOOP, TIMEND, OUTPUT, ARLXCA, DTIMEI, DRLXCA, LPCS and dynamic storage allocation.

Table 6.4-2. Basic Computational Steps for CNFWBK

1. Specification of control constants. Control constants ARLXCA (if NNA > 0), DRLXCA (if NND > 0), DTIMEI, NLØØP, ØUTPUT and TIMEND (TIMEND $\geq$ TIMEO) must be specified. LPCS is required. (Refer to Table 6.2-5 for values and Section 6.2.3.2 for description.

2. Sufficiency check on dynamic storage.
Requirements = 3*NND + NNA + NNB (NND = diffusion nodes, NNA = arithmetic nodes and NNB = boundary nodes)

3. Setting and/or calculation of time-step, Δt. (Refer to Section 6.2.4 for detailed procedure.) Time-step = DTIMEI.

4. Setting of iterative DØ loop, 1 to NLØØP.

5. Setting of source locations to zero.

6. Calling of Variables 1. (Refer to Section 6.2.2.2 for description.)

7. Checking of BACKUP. (Refer to Section 6.2.3.2 for description.)

8. Diffusion-node temperature calculations, first iteration only. Evaluation of q_i, C_i and G_k. Damping of radiation heat transfer. (Refer to Section 6.2.5.2.) Calculation of diffusion-node temperature. The computational algorithm depends upon the presence of radiation heat transfer, but the method of solution is the Crank-Nicolson algorithm (half forward and half backward, refer to Section 6.2.5.2).

9. Conversion of $T_{i,k+1}$ to °R (Rankine).

10. Diffusion-node temperature calculation, successive iterations after first. Repeating of step 8 except that q_i,C_i and G_k are not updated. Calculation of DRLXCC.

11. Acceleration of convergence every third iteration if linear extrapolation is met (refer to Section 6.2.7).

12. Conversion of $T_{i,k+1}$ to degrees Fahrenheit.

13. Calculation of arithmetic-node temperatures, second and succeeding iterations; arithmetic-node temperatures are not calculated on the first iteration (refer to Section 6.2.5.2 for details).

14. Conversion of temperatures to degrees Rankine.

15. Checking of ARLXCA and DRLXCA for convergence and ØPEITR for output. If both ARLXCA and DRLXCA are satisfied, iterations during a time-step cease, otherwise NLØØP iterations are performed.

16. Checking of ATMPCA and DTMPCA. If either one is not satisfied time-step is shortened, previous temperatures erased, and temperatures recalculated for shortened time-steps (refer to Section 6.2.5.2).

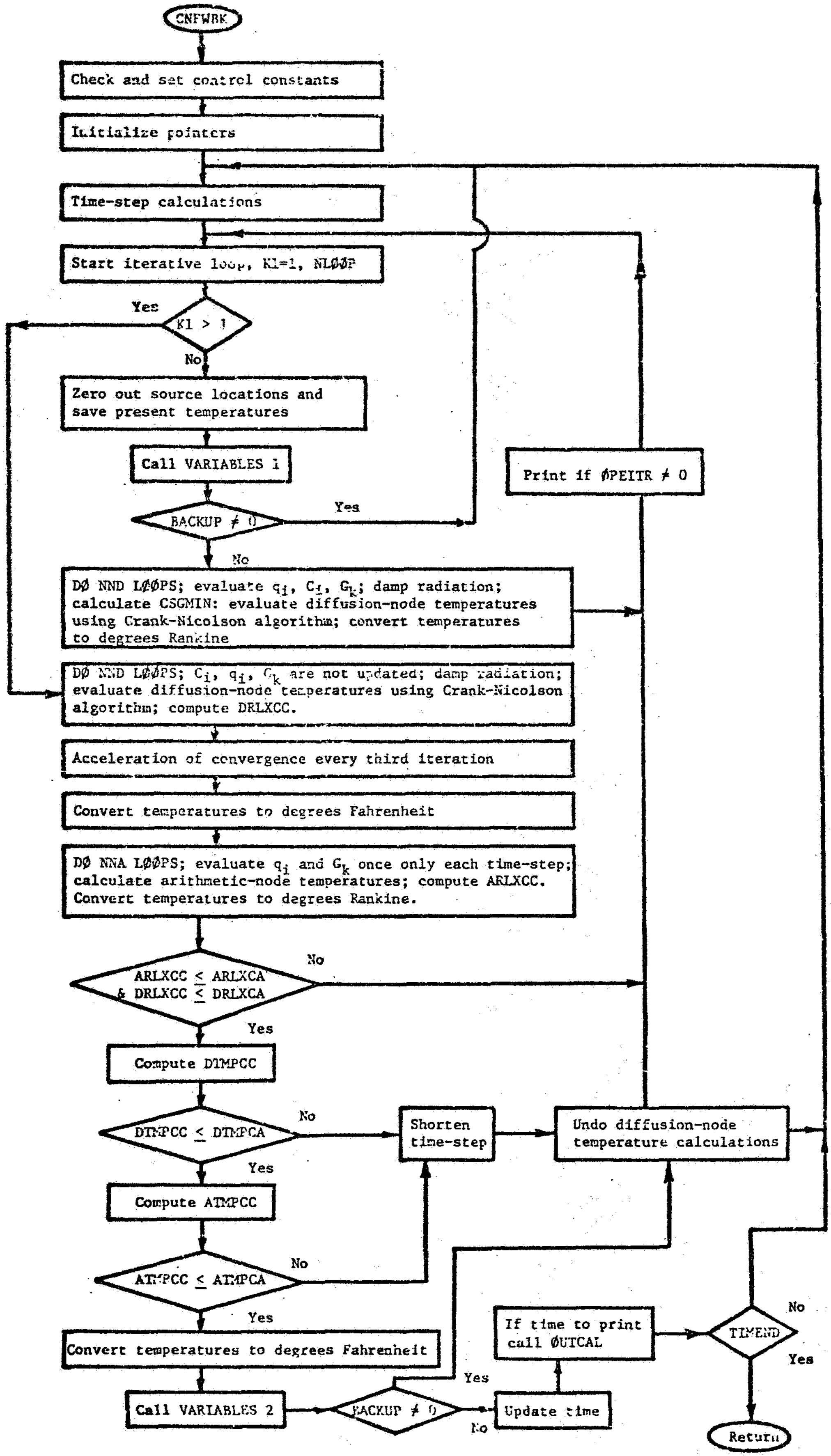
17. Conversion of temperatures back to degrees Fahrenheit.

18. Calling of VARIABLES 2 and checking of BACKUP (refer to Section 6.2.2.3 and 6.2.3.2).

19. Advancing of time, checking of time to print, and the printing of the output interval.

20. Calling of ØUTPUT CALLS.

21. Checking for problem end-time stored in user specified control constant TIMEND.



6.4-2. Functional Flow Chart for CNFWBK

6.4.3 Subroutine: CNVARB

6.4.3.1 General Comments

Subroutine CNVARB uses an implicit finite difference algorithm that is a composition of forward-differencing and backward-differencing. The proportion of forward to backward to be used is calculated internally by using a weighting factor, β, that is dependent upon the ratio of the explicit stability criterion as stored in the control constant CSGMIN divided by the computational time-step stored in DTIMEU. The weighting factor can vary each time-step but is constrained to range, $0 \leq \beta \leq 1/2$ (refer to Section 6.2.5.2 or Section 6.4.3.2). A β of one-half yields the Crank-Nicolson half-forward and half-backward expression, whereas a β of zero yields the standard backward-difference expression.

Except for the weighting factor, β, the computational procedure and the use of the various control constants in CNVARB is essentially identical to subroutine CNFWBK.

Solution characteristics should be very similar to CNFWBK solutions with expectation that CNVARB solutions would be more optimum in terms of accuracy and solution time. Solutions are not presently available to verify or refute the expected advantages of CNVARB solutions.

6.4.3.2 Finite Difference Approximation and Computational Algorithm

The numerical solution algorithm used in subroutine CNVARB is a combination of forward-differencing and backward-differencing with the weighting of each determined by the ratio of control constants CSGMIN/DTIMEU.

The combination forward-backward differencing with weighting can be expressed as:

$$\frac{C_i}{\Delta t}(T_{i,n+1}-T_{i,n}) = \beta\left(q_{i,n}+\sum_{j=1}^{p} a_{ij,n}(T_{j,n}-T_{i,n})\right) + \sum_{j=1}^{p}\sigma b_{ij,n}(T_{j,n}^4-T_{i,n}^4)$$

$$+ (1.0-\beta)\left(q_{i,n}+\sum_{j=1}^{p} a_{ij,n}(T_{j,n+1}-T_{i,n+1})\right)+\sum_{j=1}^{p}\sigma b_{ij,n}(T_{j,n+1}^4-T_{i,n+1}^4) \qquad (6.4\text{-}9)$$

$i = 1,2,\ldots,N$

n = nth time-step

β = weighting factor $(0 < \beta \leq 1/2)$

$T_{j,n};\ T_{j,n+1}$ = constant, $N < j \leq p$

If equation (6.4-9) is multiplied by 2.0 and the known quantities (at time-step, n) and the unknown quantities (at time-step, n+1) separated, the algorithm used in CNVARB may be obtained readily.

Diffusion Nodes

Diffusion-node temperatures are solved by "successive point" iteration. The finite difference iterative form as used in CNVARB can be found by multiplying equation (6.4-9) by 2.0 and by using appropriate time-step, n, and iteration, k subscripts.

$$T_{i,k+1} = DD^* \; T_{i,k} + DN^* \; [Q_{sum} - (q_i)_{ave}]/G_{sum} \qquad (6.4\text{-}10)$$

where,

$$Q_i = 2\, q_{i,n} + 2\, \overline{C}_{i,n}\, T_{i,n} + \beta' \left(\sum_{j=1}^{p} a_{ij,n} \; (T_{j,n} - T_{i,n}) + \sum_{j=1}^{p} \sigma b_{ij,n} \; (T^4_{j,n} - T^4_{i,n}) \right) \qquad (6.4\text{-}11)$$

$$Q_{sum} = Q_i + (2.0-\beta')\left(\sum_{j=1}^{i} G_{ij,n}\, T_{j,k+1} + \sum_{j=i+1}^{p} G_{ij,n}\, T_{j,k} \right) \qquad (6.4\text{-}12)$$

$$G_{sum} = 2\, \overline{C}_{i,n} + (2.0-\beta') \sum_{j=1}^{p} a_{ij,n} \qquad (6.4\text{-}13)$$

$$G_{ij,n} = a_{ij,n} + \sigma b_{ij,n}\, T^3_{j,\ell} \qquad (6.4\text{-}14)$$

(ℓ = k, if $j \geq i$ and ℓ = k+1, if $j < i$)

$$(q_i)_{ave} = \frac{(2.0-\beta')}{2} \sum_{j-1}^{p} \sigma b_{ij,n} \; [(T^4_{i,k}) + (T^4_{i,k})_2] \qquad (6.4\text{-}15)$$

average heat loss from the ith node, called radiation damping (refer to Section 6.2.6 for details)

= 0, if radiation is not present

β' = 2.0*CSGMIN/DTIMEU (range allowed, $0 \leq \beta' \leq 1.0$, note $\beta' = 2\beta$)

n = nth time-step; k = kth iteration

C_i, q_i, a_{ij}, b_{ij} = optionally specified (refer to Tables 6.2-1 - 6.2-4)

$\overline{C}_{i,n} = C_{i,n}/\Delta t$

i = 1,2,...,NND

$T_{j,n}$; $T_{j,k}$ = constant, (NND + NNA) < $j \leq p$ (p is the total number of nodes and NNA is the number of arithmetic nodes)

Arithmetic Nodes

Arithmetic nodes are calculated in the same manner in all the SINDA numerical solution routines. For the finite difference algorithm refer to either Section 6.3.1.2 or Section 6.2.5.2.

6.4.3.3 Comments on the Computational Procedure

The important steps of the computational procedure used in subroutine CNVARB are indicated in Table 6.4-3. For a detailed step-by-step computational description, the user must examine the computer listing for CNVARB in Appendix B, but some general computational details are given in Section 6.2.5.2. A functional flow chart of CNVARB is shown in Figure 6.4-3.

The computational flow pattern for CNVARB is very similar to CNFWBK or CNBACK; the slight difference is shown in the flow chart of Figure 6.4-3. The basic difference between CNVARB and the other two implicit routines is the use of a variable beta, β', which is calculated internally by the routine. Thus, the updating of the variable capacitance C_i, the impressed source q_i and the variable coefficients (a_{ij} for conduction and σb_{ij} for radiation) during the first iteration and the subsequent calculation of diffusion-node temperatures in subsequent iterations are identical to CNFWBK except for the finite difference algorithm. Use of the various control constants and checks are identical to CNFWBK.

6.4.3.4 Control Constants

Control constants for CNVARB are used in exactly the same way as used in CNFWBK. Control constant ARLXCA, DRLXCA, DTIMEI, NLØØP, ØUTPUT, and TIMEND must be specified as indicated in Table 6.2-5 and as described in Section 6.2.3.2; otherwise "run" will terminate with an appropriate error message. The function of optionally specified control constants ATMPCA, BACKUP DAMPA, DAMPD, DTIMEH, DTMPCA and TIMEØ is described in Section 6.2.3.2.

6.4.3.5 Error and Other Messages

If control constants ARLXCA, DRLXCA, DTIMEI, NLØØP, ØUTPUT and TIMEND are not specified, the following error message will be printed for each,

ARLXCA	"NØ ARLXCA"
DRLXCA	"NØ DRLXCA"
DTIMEI	"NØ DTIMEI"
NLØØP	"NØ NLØØP"
ØUTPUT	"NØ OUTPUT INTERVAL"
TIMEND	"TRANSIENT TIME NØT SPECIFIED"

If the long pseudo-compute sequence LPCS is not specified, the error message will be,

"CNVARB REQUIRES LØNG PSEUDØ-CØMPUTE SEQUENCE"

If the dynamic storage allocation is not sufficient (NDIM < (3*NND + NNA + NNB)), the error message will be,

"_____ LØCATIONS AVAILABLE"

Note that the number presented will be negative indicating the additional storage locations required.

If CSGMIN $\leq$ 0, the following message will be printed,

"CSGMIN ZERØ or NEGATIVE"

If either ARLXCA or DRLXCA is not satisifed with NLØØP iterations, the following message will be printed,

"RELAXATION CRITERIA NØT MET"

Checks on the control constants, the pseudo-compute sequence and the dynamic storage allocation are made in the following sequence with the "run" terminating if a single check is not satisfied,

NLØØP, TIMEND, ØUTPUT, ARLXCA, LPCS and dynamic storage allocation.

Table 6.4-3. Basic Computational Steps for CNVARB

1. Specification of control constants. Control constants ARLXCA (if NNA > 0), DRLXCA (if NND > 0), DTIMEI, NLØØP, ØUTPUT and TIMEND (TIMEND $\geq$ TIMEØ) must be specified. LPCS is required. (Refer to Table 6.2-5 for values and Section 6.2.3.2 for description.)

2. Sufficiency check on dynamic storage. Requirements = 3*NND + NNA + NNB (NND = diffusion nodes, NNA = arithmetic nodes and NNB = boundary nodes).

3. Setting and/or calculation of time-step, Δt. (Refer to Section 6.2.4 for detailed procedure.) Time-step = DTIMEI.

4. Setting of iterative DØ loop, 1 to NLØØP.

5. Setting of source locations to zero.

6. Calling of Variables 1 (refer to Section 6.2.2.2 for description).

7. Checking of BACKUP (refer to Section 6.2.3.2 for description).

8. Diffusion-node temperature calculations, first iteration only.

   Checking of stable stability criteria.
   Calculation of weighting factor β'= 2.0*CSGMIN/DTIMEU. ($0 \leq \beta' \leq 1.0$)
   Conversion of temperatures to degrees Rankine.
   Damping of radiation heat transfer (refer to Section 6.2.5.2).
   Calculation of diffusion-node temperatures using forward-backward algorithm with variable beta (β').
   Calculation of DRLXCC.

9. Diffusion-node temperature calculations, successive iterations after first. Repeating of step 8 except that q_i, C_i and G_k are not updated. Calculation of DRLXCC.

10. Acceleration of convergence every third iteration if linear extrapolation criterion is met (refer to Section 6.2.7).

11. Conversion of $T_{i,k+1}$ to degrees Fahrenheit.

12. Calculation of arithmetic-node temperatures every iteration (refer to Section 6.2.5.2 for details).

13. Conversion of temperatures to degrees Rankine.

14. Checking of ARLXCA and DRLXCA for convergence and ØPEITER for output. If both ARLXCA and DRLXCA are satisfied, iterations during a time-step cease, otherwise NLØØP iterations are performed.

15. Checking of ATMPCA and DTMPCA. If either one is not satisfied time-step is shortened, previous temperatures erased, and temperatures recalculated for shortened time-steps (refer to Section 6.2.5.2).

16. Conversion of temperatures back to degrees Fahrenheit.

17. Calling of VARIABLES 2 and checking of BACKUP (refer to Section 6.2.2.3 and 6.2.3.2).

18. Advancing of time, checking of time to print, and the printing of the output interval.

19. Calling of ØUTPUT CALLS.

20. Checking for problem end time stored in user specified control constant TIMEND.

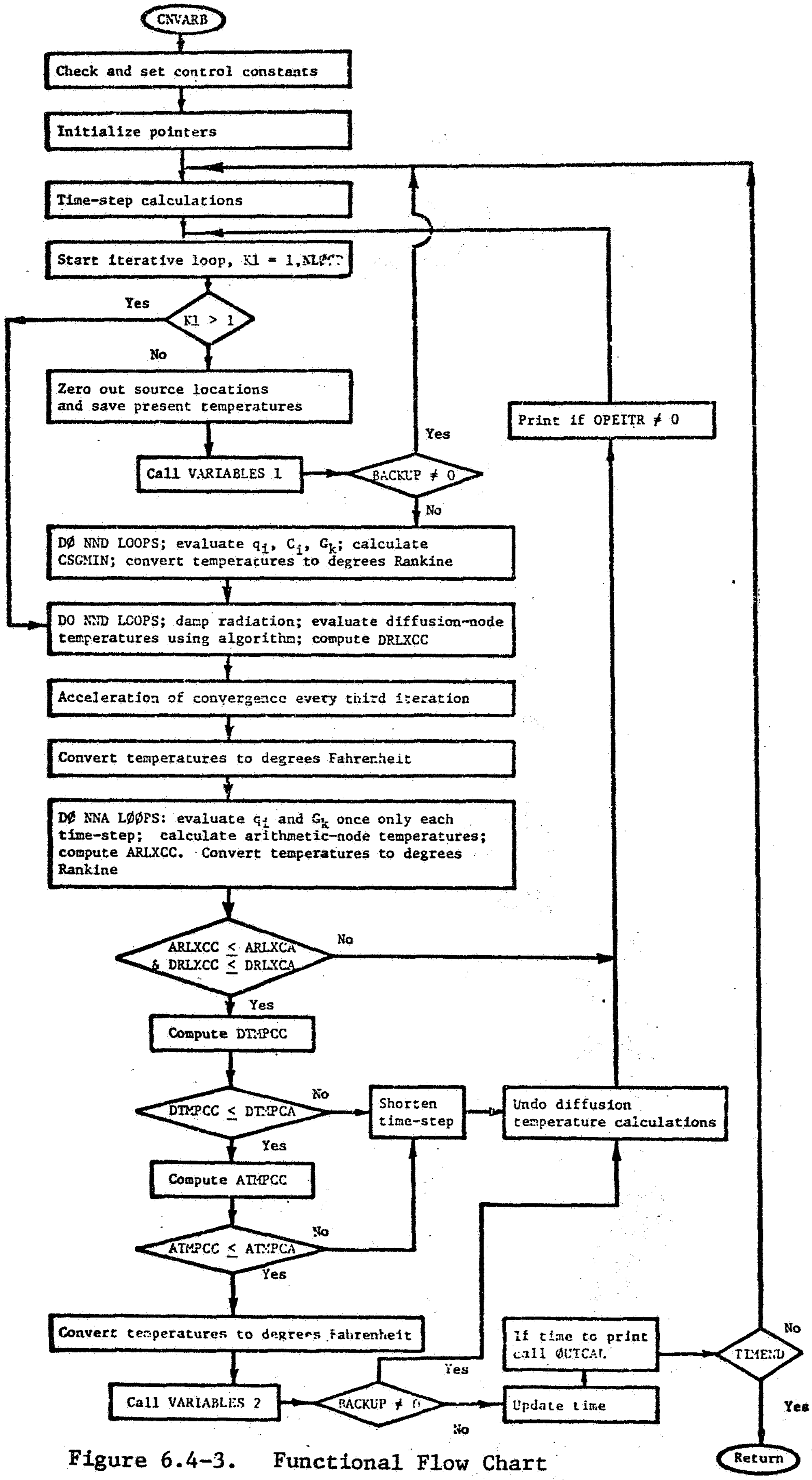


Figure 6.4-3. Functional Flow Chart for CNVARB

6.5 Steady State Numerical Solution Routines

SINDA steady state numerical solution routines number three. These steady state routines are identified as follows:

CINDSS Block iterative method
Requires short pseudo-compute sequence (SPCS)

CINDSL Successive point iterative method
Requires long pseudo-compute sequence (LPCS)

CINDSM Modified CINDSL for radiation-dominated problems
Requires long pseudo-compute sequence (LPCS)

A detailed description of steady state routines is presented in the pages to follow with liberal reference to materials presented in Section 6.2. A brief description of these routines follows.

CINDSS which uses the short pseudo-compute sequence (SPCS) was the first steady state routine developed for SINDA (via CINDA and CINDA-3G); as a result, some of the features contained in subsequent steady state routines are not used in CINDSS. If a transient analysis is to be performed following a steady state analysis, CINDSS must be used with a transient routine that also requires SPCS. The "block" iterative method (refer to Section 5.2.3) used by CINDSS should lend itself to some types of problems which are highly nonlinear with terms such as G_{ij} $(T_j^4 - T_i^4)$. With "block" iteration, both T_j and T_i are changed simultaneously. Solution convergence is based upon a temperature relaxation criterion stored in DRLXCA for diffusion nodes and ARLXCA for arithmetic nodes.

CINDSL requires the long pseudo-compute sequence (LPCS) and uses the "successive point" iteration method (refer to Section 5.2.3). Any transient analysis routine coupled with CINDSL must require LPCS. Solution time for CINDSL is less than CINDSS; as a result, it is used more often than CINDSS. A major problem with CINDSL is that a highly nonlinear problem can present convergence difficulties unless considerable amount of damping is used. For example, a radiation-dominated problem contains many σb_{ij} $(T_j^4 - T_i^4)$. With "successive point" iteration, T_j may be updated and T_i not for a given conductor; as a result, the resultant heat flow calculation could present difficulties because of large change in values. CINDSL has the acceleration of convergence feature, whereas CINDSS does not.

Solution convergence is based upon temperature relaxation criterion stored in DRLXCA for diffusion nodes and ARLXCA for arithmetic nodes.

CINDSM is the latest addition to the SINDA library of steady state routines. CINDSM requires the long pseudo-compute sequence and uses "successive point" iteration. The routine was specifically developed to solve radiation-dominated problems. Solution convergence is based upon system energy criterion stored in BALENG.

6.5.1 Subroutine: CINDSS

6.5.1.1 General Comments

Subroutine CINDSS is a steady state routine that requires the short pseudo-compute sequence (SPCS) and ignores the capacitance values of diffusion nodes to calculate steady state temperatures. Diffusion nodes are solved by a "block" iterative method as discussed in Section 6.5.2.3, whereas arithmetic nodes are solved by a "successive point" iterative method also discussed in Section 6.5.2.3. For steady state solutions diffusion nodes are not necessary; as a matter of fact, solutions will be achieved more quickly if all diffusion nodes are specified as arithmetic. The use of diffusion nodes in a steady state solution allows for the direct use of the transient model.

A series of steady state solutions at various points in a time period can be accomplished by specifying control constants TIMEN and ØUTPUT. ØUTPUT is used both as the output interval and the computational interval. The instructions with the appropriate call are made in VARIABLES 1 to modify boundary conditions with time.

The CINDSS call can be followed by a call to one of the transient solution subroutines which has the same short pseudo-compute sequence requirements such as CNFRWD. In this manner the steady state solution becomes the initial conditions for the transient analysis. It is important to remember that control constants specified for the steady state routine will be used by the transient routine unless initialized to the desired values. Since CINDSS utilizes control constants TIMEND and ØUTPUT for the steady state-transient problem, the user must specify their values in the execution block after the steady state call and prior to the transient analysis call. CINDSS does not utilize the acceleration of convergence feature as discussed in Section 6.2.7.

Solution convergence is based upon a temperature relaxation criterion stored in control constants DRLXCA for diffusion nodes and ARLXCA for arithmetic nodes. Normally, identical values are specified for both DRLXCA and ARLXCA. Sufficient information is not presently available to indicate different values for DRLXCA and ARLXCA. A method to indicate the accuracy of the "converged" temperatures is not presently available. It should also be noted that "converged" temperatures could have large system energy unbalance.

6.5.1.2 Finite Difference Approximation and Computational Algorithm

The steady state heat balance equation at the ith node may be readily expressed as,

$$q_i + \sum_{j=1}^{p} a_{ij} (T_j - T_i) + \sum_{j=1}^{p} \sigma b_{ij} (T_j^4 - T_i^4) = 0 \qquad (6.5\text{-}1)$$

$$i = 1,2,..,N$$
$$T_j = \text{constant},\ N < j \leq p$$

Equation (6.5-1) represents a set of nonlinear algebraic equations to be solved simultaneously. Since CINDSS solves temperature of nodes specified as diffusion (nodes with capacitance even though a steady state solution is desired) by the "block" iteration method and temperatures of nodes specified as arithmetic (no capacitance) by the "successive point" iteration method, two successive approximation algorithms are used.

Diffusion Nodes (if any)

$$T_{i,k+1} = DD{*}\ T_{i,k} + \frac{DN * (q_{i,k} + \sum_{j=1}^{p} G_{ij,k} T_{j,k})}{\sum_{j=1}^{p} G_{ij,k}} \qquad (6.5\text{-}2)$$

where, k = kth iteration; $i = 1,2,...,NND$ (number of diffusion nodes)

q_i, a_{ij}, b_{ij} = may be optionally specified (refer to Tables 6.2-1 - 6.2-4)

$T_{j,k}$ = constant, $(NND + NNA) < j \leq p$ (NNA is the number of arithmetic nodes and p is the total number of nodes)

$$G_{ij,k} = a_{ij,k} + \sigma b_{ij,k} (T_{j,k}^2 + T_{i,k}^2)(T_{j,k} + T_{i,k})$$

DN ≡ DAMPD (diffusion node damping factor)

DD = 1.0 - DN

Arithmetic Nodes (if any)

$$T_{i,k+1} = AD{*}\ T_{i,k} + \frac{AN{*}\ (q_{i,k} + \sum_{j=1}^{i} G_{ij,k} T_{j,k+1} + \sum_{j=i+1}^{p} G_{ij,k} T_{j,k})}{\sum_{j=1}^{p} G_{ij,k}}$$

where, k = kth iteration; $i = (NND + 1), (NND + 2), \ldots, (NND + NNA)$

q_i, a_{ij}, b_{ij} = optionally specified (refer to Tables 6.2-1 - 6.2-4)

$T_{j,k}$ = constant, $(NND + NNA) < j \leq p$ (NNA is the number of arithmetic nodes and p is the total number of nodes

$$G_{ij,k} = a_{ij,k} + \sigma b_{ij,k} (T_{j,\ell}^2 + T_{i,}^2)(T_{j,\ell} + T_{i,k})$$

$(\ell = k, \text{ if } j \geq i \text{ and } \ell = k+1, \text{ if } j < i)$

AN $\equiv$ DAMPA (arithmetic node damping factor)

AD = 1.0 - AN

6.5.1.3 Comments on the Computational Procedure

The important steps of the computational procedure used in the steady state subroutine CINDSS are indicated in Table 6.5-1. For a detailed procedural description, the user must examine the computer listing for CINDSS in Appendix C, but some general computational details are given in Section 6.2.5.3. A functional flow chart of CINDSS is shown in Figure 6.5-1. The user is required to specify the maximum number of iterations to be performed via control constant NLØØP and the diffusion-node temperature change relaxation criteria DRLXCA and the arithmetic-node temperature change criteria ARLXCA. The iterations continue until either NLØØP is satisfied or both DRLXCA and ARLXCA are satisfied. If DRLXCA and ARLXCA are not satisfied with NLØØP iterations, an appropriate message is printed. VARIABLES 1 and ØUTPUT CALLS are performed at the start and VARIABLES 2 and ØUTPUT CALLS are performed upon completion. Control constants DAMPD for diffusion nodes and DAMPA for arithmetic nodes are so-called damping factors which are multipliers of the "new" temperatures; the factor 1.0 - DAMPD (or 1.0 - DAMPA) is a multiplier for the "old" temperatures. This weighting of "old" and "new" temperatures is useful for damping oscillations due to nonlinearities. For nonlinear systems, the damping factors are specified to be less than one. If not specified, the damping factor is set to 1.0. As a point of interest, it appears that if a linear system is to be solved, the convergence could be accelerated by using the damping factor greater than one. The diffusion nodes receive a "block" iteration, whereas the arithmetic nodes receive a "successive point" iteration; acceleration features are not utilized.

6.5.1.4 Control Constants

Control constant NLØØP must be specified and control constants ARLXCA and DRLXCA must be specified if NNA > 0 and NND > 0, respectively; otherwise "run" will terminate with an appropriate error message. Control constants DAMPA and DAMPD may be optionally specified among others. Control constant characteristics are tabulated in Table 6.2-5 and description of these control constant is presented in Section 6.2.3.2. Specification of NLØØP is dependent upon the values of ARLXCA and DRLXCA and thus the accuracy of solution. Since the type of problem will influence accuracy, it appears that a trial and error procedure is the only practical way of determining realistic control constant values.

6.5.1.5 Error and Other Messages

If control constants ARLXCA, DRLXCA and NLØØP are not specified, the following error message will be printed for each,

ARLXCA	"NØ ARLXCA"
DRLXCA	"NØ DRLXCA"
NLØØP	"NØ NLØØP"

If the short pseudo-compute sequence SPCS is not specified, the error message will be,

"CINDSS REQUIRES SHØRT PSEUDO-CØMPUTE SEQUENCE"

If the dynamic storage allocation is not sufficent (NDIM < NND) will be,

"_____ LØCATIONS AVAILABLE"

Note that the number printed will be negative indicating the additional storage locations required.

If both temperature change relaxation criteria ARLXCA and DRLXCA are not met with NLØØP iterations, the message will be,

"ITERATIØN CØUNT EXCEEDED, LØØPCT = _____"

Checks on the control constants, the pseudo-compute sequence, and the dynamic storage allocation are made in the following order with the "run" terminating if a single check is not satisfied.

NLØØP, ARLXCA, DRLXCA, SPCS, and dynamic storage allocation.

Table 6.5.1. Basic Computational Steps for CINDSS

1. Specification of control constants. Control constants ARLXCA (if NNA > 0), DRLXCA (if NND > 0) and NLØØP must be specified. SPCS is required. (Refer to Table 6.2-5 for values and Section 6.2.3.2 for description.)

2. Sufficiency check on dynamic storage. Requirements = NND (NND = diffusion nodes).

3. Setting of TIMEN for first iteration and succeeding iterations.

   TIMEN = TIMEØ, first iteration
   TIMEN = TIMEØ + ØUTPUT, succeeding iterations

4. Setting of iterative loop for all nodes, kl = 1, NLØØP

5. Setting of source locations to zero.

6. Calling of VARIABLES 1 (refer to Section 6.2.2.2 for description).

7. Calculation of diffusion-node temperatures by "block" iteration if NND > 0 (refer to sections 6.2.5.3 and 6.5.1.2).

$$T_{i,k+1} = DD^* \, T_{i,k} + \frac{DN^* \left(q_{i,k} + \sum_{j=1}^{p} G_{ij,k} \, T_{j,k}\right)}{\sum_{j=1}^{p} G_{ij,k}}$$

$$DN = DAMPD \text{ and } DD = 1.0 - DN$$

8. Calculation of DRLXCC.

9. Calculation of arithmetic-node temperatures by "successive point" iteration if NNA > 0 (refer to Sections 6.2.5.3 and 6.5.1.2).

$$T_{i,k+1} = AD^* \, T_{i,k} + \frac{AN^* \left(q_{i,k} + \sum_{j=1}^{i} G_{ij,k} \, T_{j,k+1} + \sum_{j=i+1}^{p} G_{ij,k} \, T_{j,k}\right)}{\sum_{j=1}^{p} G_{ij,k}}$$

$$AN = DAMPA$$
$$AD = 1.0 - DAMPA$$

10. Calculation of ARLXCC.

11. Checking of DRLXCC and ARLXCC against the relaxation criteria DRLXCA and ARLXCA, respectively, for convergence. If both ARLXCA and DRLXCA are satisfied, iterations cease, otherwise NLØØP iterations are performed.

12. Calculation of system energy balance which is stored in ENGBAL.

13. Call VARIABLES 2 and ØUTCAL, print ENGBAL and LØØPCT.

14. Check if TIMEND = TIMEN.

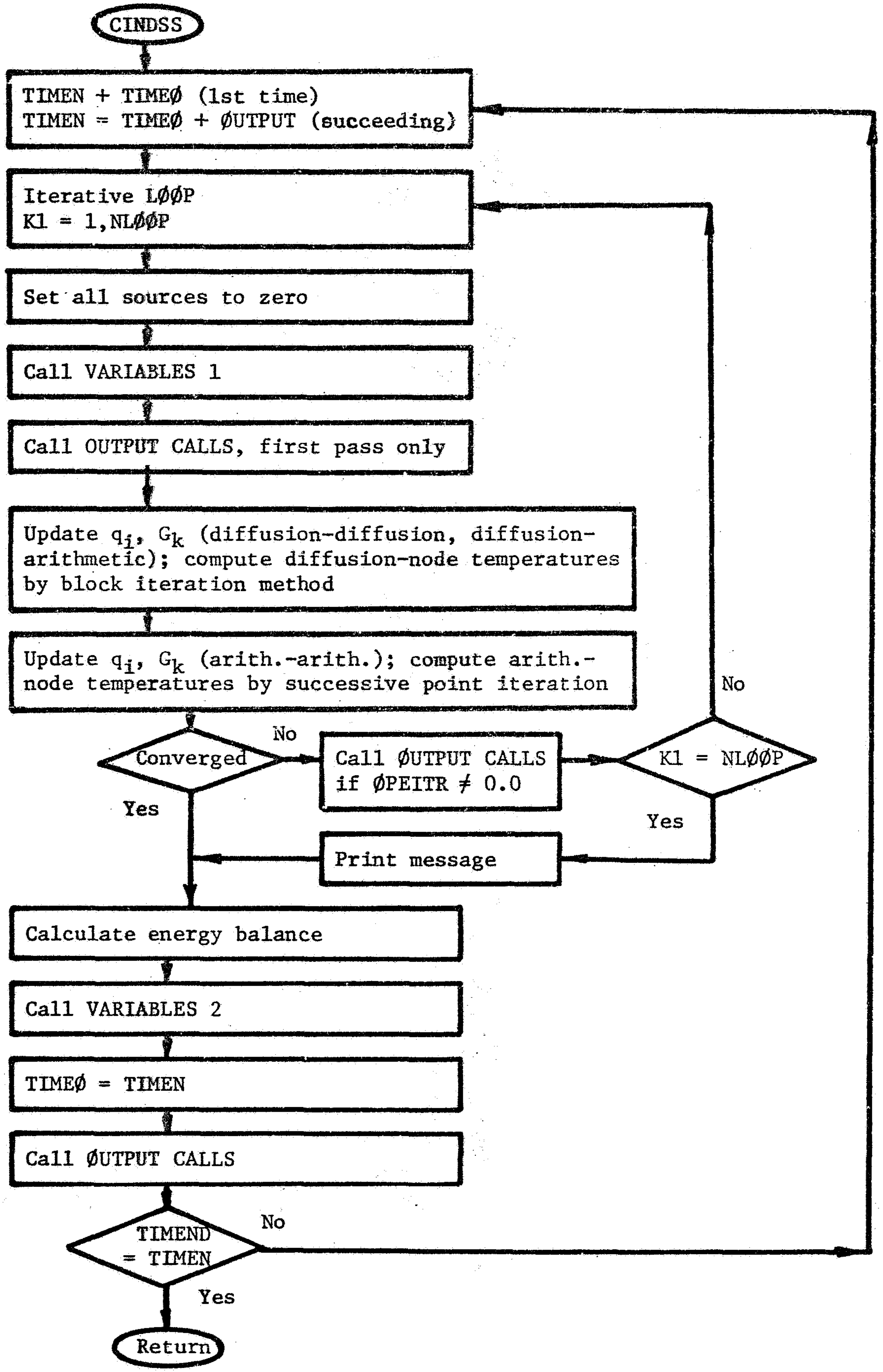


Figure 6.5-1. Functional Flow Chart for CINDSS

6.5.2 Subroutine: CINDSL

6.5.2.1 General Comments

Subroutine CINDSL is a steady state routine that requires the long pseudo-compute sequence (LPCS). Both diffusion- and arithmetic-node temperatures are calculated by a "successive point" iteration computational technique. Every third iteration a linear extrapolation is performed to accelerate convergence. CINDSL generally yields significantly faster solutions than CINDSS, but nonlinear problems such as those with radiation heat transfer can pose considerable convergence difficulties unless a large amount of damping (low values of DAMPA and DAMPD) is imposed.

A series of steady state solutions at various points in time can be generated by specifying control constants TIMEND and ØUTPUT. ØUTPUT is used both as the output interval and the computation interval; this requires appropriate calls in VARIABLES 1 to modify boundary conditions with time.

CINDSL can be followed by a call to one of the transient numerical solution routines which have the same LPCS requirements. Used in this manner the steady state solutions become the initial conditions for the transient analysis. Note that since CINDSL utilizes control constants TIMEND and ØUTPUT for the coupled steady state-transient problem, the user must specify the values of TIMEND and ØUTPUT in the execution block after the steady state call and prior to the transient analysis call.

Solution convergence is based upon a temperature relaxation criterion stored in control constants DRLXCA for diffusion nodes and ARLXCA for arithmetic nodes. Normally, identical values are specified for both DRLXCA and ARLXCA for lack of anything better. The damping factors DAMPD for diffusion nodes and DAMPA for arithmetic nodes are merely multipliers of "new" temperatures and the factor 1.0 - DAMPD (or 1.0 - DAMPA) is a multiplier of the "old" temperatures. Normally, these damping factors are specified to be less than 1.0, but for a linear system the convergence probably could be accelerated by using a damping factor greater than one.

6.5.2.2 Finite Difference Approximation and Computational Algorithm

The set of steady state heat balance equations,

$$q_i + \sum_{j=1}^{p} a_{ij} (T_j - T_i) + \sum_{j=1}^{p} \sigma b_{ij} (T_j^4 - T_i^4) = 0$$

$$i = 1,2,\ldots,N$$

$$T_j = \text{constant } N < j \leq p$$

is solved by a re-iterative scheme called a "successive point" iterative method here. Both diffusion-node and arithmetic-node temperatures are solved in this manner. The only difference between the two algorithms is that control constant DAMPD is used with diffusion nodes and control constant DAMPA is used with arithmetic nodes.

Diffusion Nodes (if any)

$$T_{i,k+1} = DD^* \, T_{i,k} + DN^* \, \frac{\left(q_{i,k} + \sum_{j=1}^{i} G_{ij,k} \, T_{j,k+1} + \sum_{j=i+1}^{p} G_{ij,k} \, T_{j,k}\right)}{\sum_{j=1}^{p} G_{ij,k}} \qquad (6.5\text{-}4)$$

where, $i = 1,2,\ldots,NND$; k = kth iteration

q_i, a_{ij}, b_{ij} = may be optionally specified (refer to Tables 6.2-1 - 6.2-4)

$T_{j,k}$ = constant, $(NND + NNA) < j \leq p$ (NNA is the number of arithmetic nodes and p is the total number of nodes)

DN = DAMPD (diffusion-node damping factor)

DD = 1.0 - DN

$$G_{ij,k} = a_{ij,k} + \sigma b_{ij,k} \, (T_{j,\ell}^2 + T_{i,k}^2)(T_{j,\ell} + T_{i,k})$$

$(\ell = k \text{ if } j \geq i \text{ and } \ell = k{+}1 \text{ if } j < i)$

Arithmetic Nodes (if any)

$$T_{i,k+1} = AD^* \, T_{i,k} + AN^* \, \frac{\left(q_{i,k} + \sum_{j=1}^{p} G_{ij,k} \, T_{j,k+1} + \sum_{j=i+1}^{p} G_{ij,k} \, T_{j,k}\right)}{\sum_{j=1}^{p} G_{ij,k}} \qquad (6.5\text{-}5$$

where, $i = (NND + 1), (NND + 2), \ldots, (NND + NNA)$

q_i, a_{ij}, b_{ij} = may be optionally specified (refer to Tables 6.2-1 - 6.2-4)

$T_{j,k}$ = constant $(NND + NNA) < j \leq p$ (NNA is the number of arithmetic nodes and p is the total number of nodes)

AN = DAMPA (arithmetic-node damping factor)

AD = 1.0 - AN

$$G_{ij,k} = a_{ij,k} + \sigma b_{ij,k} \, (T_{j,\ell}^2 + T_{i,k}^2)(T_{j,\ell} + T_{i,k})$$

$(\ell = k \text{ if } j \geq i \text{ and } \ell = k{+}1 \text{ if } j < i)$

6.5.2.3 Comments on the Computational Procedure

The important steps of the computational procedure used in the steady state subroutine CINDSL are indicated in Table 6.5-2. For a detailed procedural description, the user must examine the computer listing for CINDSL in Appendix C, but some general computational details are given in Section 6.2.5.3. A functional flow chart of CINDSL is shown in Figure 6.5-2.

The computational pattern of CINDSL is very similar to CINDSS with the differences being that CINDSL uses the long pseudo-compute sequence, whereas CINDSS uses the short pseudo-compute sequence, and that CINDSL contains the acceleration convergence feature, whereas CINDSS does not. The user is required to specify the maximum number of iterations to be performed via control constant NLØØP and the diffusion-node temperature change relaxation criteria DRLXCA and the arithmetic-node temperature change relaxation criteria ARLXCA. The iterations continue until either NLØØP is satisfied or both DRLXCA and ARLXCA are satisfied. If DRLXCA and ARLXCA are not satisfied with NLØØP, an appropriate message is printed. Acceleration of convergence is performed every third iteration if a temperature is converging over two time-steps.

6.5.2.4 Control Constants

Control constant NLØØP must be specified and control constants ARLXCA and DRLXCA must be specified if NNA > 0 and NND > 0, respectively; otherwise "run" will terminate with an appropriate error message. Control constants DAMPA and DAMPD may be optionally specified among others. Control constant characteristics are tabulated in Table 6.2-5 and description of these control constants is presented in Section 6.2.3.2. Specification of NLØØP is dependent upon the values of ARLXCA and DRLXCA and thus the accuracy of the solution. Since the type of problem will influence accuracy, it appears that a trial and error procedure is the only practical way of determining realistic control constant values.

6.5.2.5 Error and Other Messages

If control constants ARLXCA, DRLXCA and NLØØP are not specified, the following error message will be printed for each,

ARLXCA	"NØ ARLXCA"
DRLXCA	"NØ DRLXCA"
NLØØP	"NØ NLØØP"

If the long pseudo-compute sequence LPCS is not specified, the error message will be,

"CINDSL REQUIRES LØNG PSEUDØ-COMPUTE SEQUENCE"

If the dynamic storage allocation is not sufficient, (NDIM < 2* (NNA + NND)), the message will be,

"_____ LØCATIONS AVAILABLE"

Note that the number printed will be negative indicating the additional storage locations required.

"LØØPCT = _____ and ENGBAL = _____"

If both temperature change relaxation criteria, ARLXCA and DRLXCA, are not met with NLØØP iterations, the message will be,

"ITERATIØN CØUNT EXCEEDED, LØØPCT = _____"

Checks on the control constants, the pseudo-compute sequence, and the dynamic storage allocation are made in the following order with the "run" terminating if a single check is not satisfied.

NLØØP, ARLXCA, DRLXCA, LPCS, and dynamic storage allocation.

Table 6.5.2 Basic Computational Steps for CINDSL

1. Specification of control constants. Control constants ARLXCA (if NNA > 0), DRLXCA (if NND > 0) and NLØØP must be specified. LPCS is required. (Refer to Table 6.2-5 for values and Section 6.2.3.2 for dexcription.)

2. Sufficiency check on dynamic storage. Requirements = 2* (NND + NNA) (NND = diffusion nodes and NNA = arithmetic nodes).

3. Setting of TIMEN for first and succeeding iterations.

   TIMEN = TIMEØ, first iteration
   TIMEN = TIMEØ + ØUTPUT, succeeding iterations

4. Setting of iterative loop for all nodes, kl = 1, NLØØP.

5. Setting of source locations to zero.

6. Calling of VARIABLES 1 (refer to Section 6.2.2.2 for description).

7. Calculation of diffusion-node temperatures by "block" iteration if NND > 0 (refer to Section 6.2.5.2 and 6.5.1.2).

$$T_{i,k+1} = DD^* \, T_{i,k} + \frac{DN^* \left(q_{i,k} + \sum_{j=1}^{i} G_{ij,k} T_{j,k+1} + \sum_{j=i+1}^{p} G_{ij,k} T_{j,k}\right)}{\sum_{j=1}^{p} G_{ij,k}}$$

DN = DAMPD and DD = 1.0 - DN

8. Calculation of DRLXCC.

9. Calculation of arithmetic-node temperatures by "successive point" iteration if NNA > 0 (refer to Sections 6.2.5.3 and 6.5.1.2).

$$T_{i,k+1} = AD^* \, T_{i,k} + \frac{AN^* \left(q_{i,k} + \sum_{j=1}^{i} G_{ij,k} T_{j,k+1} + \sum_{j=i+1}^{p} G_{ij,k} T_{j,k}\right)}{\sum_{j=1}^{p} G_{ij,k}}$$

10. Calculation of ARLXCC.

11. Checking of DRLXCC and ARLXCC against the relaxation criteria DRLXCA and ARLXCA, respectively, for convergence. If both ARLXCA and DRLXCA are satisfied, iterations cease, otherwise NLOOP iterations are performed.

12. Acceleration of convergence each third iteration, if linear extrapolation criterion is met (refer to Section 6.2.7).

13. Calculation of system energy balance which is stored in ENGBAL.

14. Call VARIABLES 2 and ØUTCAL, print ENGBAL and LØØPCT.

15. Check if TIMEND = TIMEN.

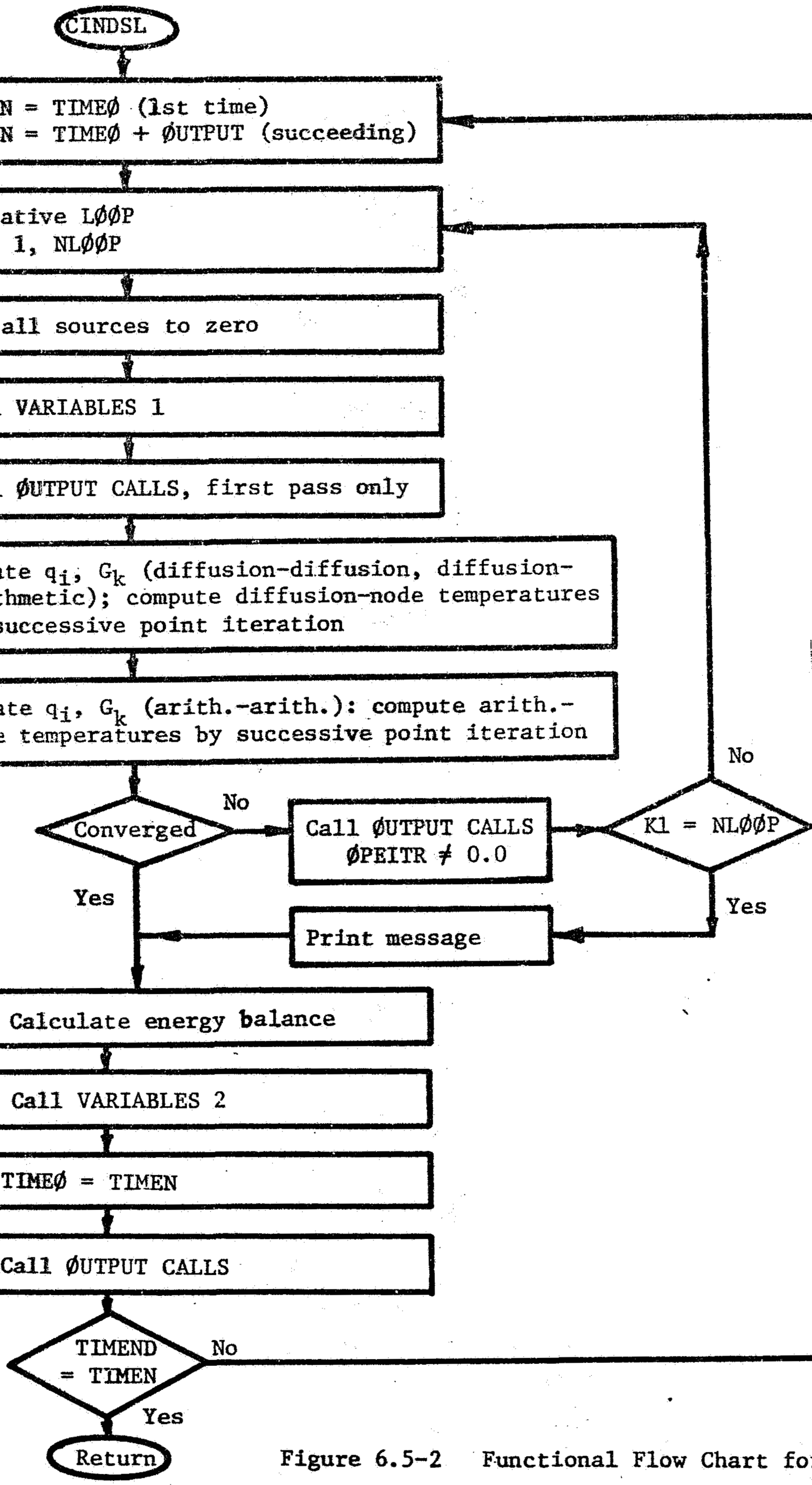


Figure 6.5-2 Functional Flow Chart for CINDSL

6.5.3 Subroutine: CINDSM

6.5.3.1 General Comments

Subroutine CINDSM is a steady state routine specifically generated for radiation dominated problems. CINDSM requires the long pseudo-compute sequence (LPCS) and is considerably different from CINDSL. CINDSM is based on the use of pseudo linear equations which are the result of linearizing the radiation conductors. These equations are solved by using the "successive point" method with LAXFAC iterations. Updating of the properties as well as the linearized conductors occur outside of the iterative loops. Temperature convergence is based on a criterion that is continually tightened until either the NLØØP iterations or the system energy balance criterion stored in BALENG has been satisfied.

The acceleration of convergence by linear extrapolation as used in CINDSM is essentially the same as used in the other SINDA numerical solution routines, but in lieu of limiting the extrapolation by an allowable slope value (refer to Section 6.2.7) the maximum temperature change of the network on the last iteration is used as the allowable value.

Information available at this time indicates that each problem appears to have an optimum combination of NLØØP, DAMPD, and LAXFAC values. An NLØØP of 100, a DAMPD of 0.5 and a LAXFAC of 10 has been successfully applied to spacecraft problems with radiation domination, but the solution time is rather long.

6.5.3.2 Finite Difference Approximation and Computational Algorithm

The set of steady state heat balance equations,

$$q_i + \sum_{j=1}^{p} a_{ij} (T_j - T_i) + \sum_{j=1}^{p} \sigma b_{ij} (T_j^4 - T_i^4) = 0$$

$$i = 1,2,\ldots,N$$
$$T_j = \text{constant},\ N < j \leq p$$

is solved by a re-iterative "successive point" method after linearization. Linearization is achieved by letting $\sigma b_{ij} (T_j^4 - T_i^4) = G_r (T_j - T_i)$ with $G_r = \sigma b_{ij} (T_j^2 + T_i^2)(T_j + T_i)$. This yields

$$q_i + \sum_{j=1}^{p} a_{ij} (T_j - T_i) + \sum_{j=1}^{p} G_r (T_j - T_i) = 0 \qquad (6.5\text{-}6)$$

Diffusion and Arithmetic Nodes

No distinction is made between diffusion and arithmetic nodes. As a result, the following algorithm applies to both types of nodes,

$$T_{i,k} = DD^* \, T_{i,k} + DN^* \frac{\left(q_{i,L} + \sum_{j=1}^{i} G_{ij,L} \, T_{j,k+1} + \sum_{j=i+1}^{p} G_{ij,L} \, T_{j,k}\right)}{\sum_{j=1}^{p} G_{ij,L}} \qquad (6.5\text{-}7)$$

where, $i = 1,2,\ldots,(NND + NNA)$; p = total number of nodes

k = kth iteration

L = before each LAXFAC iterative loop

$T_{j,k}$ = constant, $(NND + NNA) < j \leq p$

DN = DAMPD (diffusion-node damping factor; DAMPA is not used)

DD = 1.0 - DAMPD

$G_{ij,L} = a_{ij,L} + \sigma b_{ij,L} (T^2_{j,L} + T^2_{i,L})(T_{j,L} + T_{i,L})$

($G_{ij,L}$ is updated once before each LAXFAC iterative loop

NNA = number of arithmetic nodes

NND = number of diffusion nodes

q_i, a_{ij}, b_{ij} = may be optionally specified (refer to Tables 6.2-1 - 6.2-4)

6.5.3.3 Comments on the Computational Procedure

A detailed step-by-step computational procedure as used in the steady state routine CINDSM is presented in Table 6.5-3. For a more detailed procedural description, the user must examine the computer listing in Appendix C. A functional flow chart that is compatible with the step-by-step description of Table 6.5-3 is shown in Figure 6.5-3.

CINDSM is considerably different from either CINDSS or CINDSL because of the use of a variable convergence criterion which is internally updated. Overall, from a total system basis, control constants NLØØP and BALENG are the ultimate criteria.

It should be particularly noted here that unlike CINDSS or CINDSL, which use both DAMPA and DAMPD, CINDSM uses only DAMPD. The reason for this is that CINDSM does not treat the nodal types as diffusion or arithmetic.

6.5.3.4 Control Constant

Control constants BALENG, LAXFAC AND NLØØP must be specified; otherwise the "run" will terminate with an appropriate error message. Control constant DAMPD may be optionally specified among others. Control constant characteristics are tabulated in Table 6.2.5 and description of these control constants is presented in Section 6.2.3.2. Specification of BALENG, LAXFAC and NLOOP appears to be a trial and error procedure.

6.5.3.5 Error and Other Messages

If control constants BALENG, LAXFAC, and NLØØP are not specified, the following error message will be printed for each,

BALENG	"NØ BALENG"
LAXFAC	"NØ LAXFAC"
NLØØP	"NØ NLØØP"

If the long pseudo-compute sequence LPCS is not specified, the error message will be,

"CINDSM REQUIRES LØNG PSEUDØ-CØMPUTE SEQUENCE"

If the dynamic storage allocation is not sufficient, (NDIM < (3* NNA + 3* NND + NGT)), the message will be,

"_____ LOCATIONS AVAILABLE"

Note that the number printed will be negative indicating the additional storage locations required.

If either NLØØP iterations has been made or if ENGBAL $\leq$ BALENG, the following message is printed,

"LØØPCT = _____ and ENGBAL _____"

Checks on the control constants, the pseudo-compute sequence and the dynamic storage allocation are made in the following order with the "run" terminating if a single check is not satisfied,

NLØØP, LPCS, BALENG, LAXFAC and dynamic storage allocation.

Table 6.5.3. Basic Computational Steps for CINDSM

1. Specification of control constants. Control constants BALENG, LAXFAC and NLØØP must be specified. The long pseudo-compute sequence (LPCS) is required. (Refer to Table 6.2-5 for values and Section 6.2.3.2 for description.)

2. Sufficiency check on dynamic storage. Requirement = 3* (NND + NNA) + NGT (NND = diffusion nodes, NNA = arithmetic nodes and NGT = total number of conductors).

3. Setting of TIMEN for the first and succeeding iterations.

   TIMEN = TIMEØ, first iteration
   TIMEN = TIMEØ + ØUTPUT, succeeding iterations

4. Constants used in CINDSM

   NLAX = NLØØP/LAXFAC (both NLØØP and LAXFAC are specified by the user)
   RELAX = .05 (initial value used in CINDSM as the allowable temperature change)
   DELXXX = .05/NLAX (a number used in reducing RELAX for a tighter criterion)
   XXXDUM = .001 (a value of RELAX used in CINDSM for a tighter criterion)
   = .001/5 (a subsequent value of RELAX for a tighter criterion)
   DAMP = DAMPD (damping factor for all nodes; DAMPA is not used)

5. Updating of variables and linearization of radiation .

   Variable q_i and G_k are evaluated by calling subroutine NØNLIN.

   Linearization means that the radiation exchange expressed as $\sigma b_{ij}(T_j^4-T_i^4)$. Normally, G_{ij} would be updated each iteration as done in CINDSS or CINDSL, but in CINDSM G_{ij} is not updated within the DØ-LØØP (kl = 1,LAXFAC) but is updated outside of the loop.

6. Iterative DØ-LØØP (kl = 1,LAXFAC) is established.

   Temperatures of all nodes are calculated by "successive point" iteration <u>with no damping</u>.

$$T_{i,k+1} = \frac{q_i + \sum_{j=1}^{i} G_{ij}\, T_{j,k+1} + \sum_{j=i+1}^{p} G_{ij}\, T_{j,k}}{\sum_{j=1}^{p} G_{ij}} \qquad (6.5\text{-}5)$$

   where, $G_{ij} = a_{ij} + \sigma b_{ij}\,(T_j^2 + T_i^2)(T_j + T_i)$ (q_i and G_{ij} are not updated during the LAXFAC iterations)

   Check on temperature convergence. Temperatures have converged if,

$$|T_{i,k+1} - T_{i,k}|_{max} \leq \text{RELAX } (= .05)$$

   If temperatures have converged, the computation goes out of the iteration loop to step (7).

Every third iteration, acceleration of convergence is attempted if linear extrapolation criterion is met (refer to Section 6.2.7).

Iteration ceases if LAXFAC iterations have been performed or if the temperatures have converged.

7. Check on NLAX iterations.

   If in step (6) the number of iterations, LOOPCT $\geq$ NLAX, the computational procedures go to step (9). However, in step (6) if the number of iterations LOOPCT < NLAX, then a set of temperature calculations is made using "successive point" method with a damping factor and no iterations.

$$T_{i,k+1} = DD^* \, T_{i,k} + DN^* \, \frac{\left(q_i + \sum_{j=1}^{i} G_{ij} \, T_{j,k+1} + \sum_{j=i+1}^{p} G_{ij} \, T_{j,k}\right)}{\sum_{j=1}^{p} G_{ij}}$$

   where, DN = DAMPD (diffusion node damping factor; note DAMPA is not used)
   G_{ij} = constant

   Allowable temperature change criterion RELAX is reduced to,

   RELAX = .05 - (.05/NLAX)

   and computational procedure goes to step (5).

8. Repetition of steps (5) through (7) except for temperature convergence criterion.

   Temperatures have converged if,

$$|T_{i,k+1} - T_{i,k}|_{max} \leq \text{RELAX } (= .05 - .05/\text{NLAX})$$

9. Assuming step (7) has been satisfied, LØØPCT is checked against NLØØP.
   If LOOPCT $\geq$ NLOOP, the computation proceeds to step (12).
   If LOOPCT < NLOOP computation proceeds to step (10).

10. Reduce RELAX to .001.

11. Check on temperature convergence.
    If $|T_{i,k+1} - T_{i,k}| \leq$ RELAX (= .001) go to step (12).
    $|T_{i,k+1} - T_{i,k}| >$ RELAX (= .001), LAXFAC is reduced to
    LAXFAC = NLØØP - LØØPCT,
    and steps (5) through (11) are repeated.

12. Compute system energy balance and store in control constant ENGBAL.

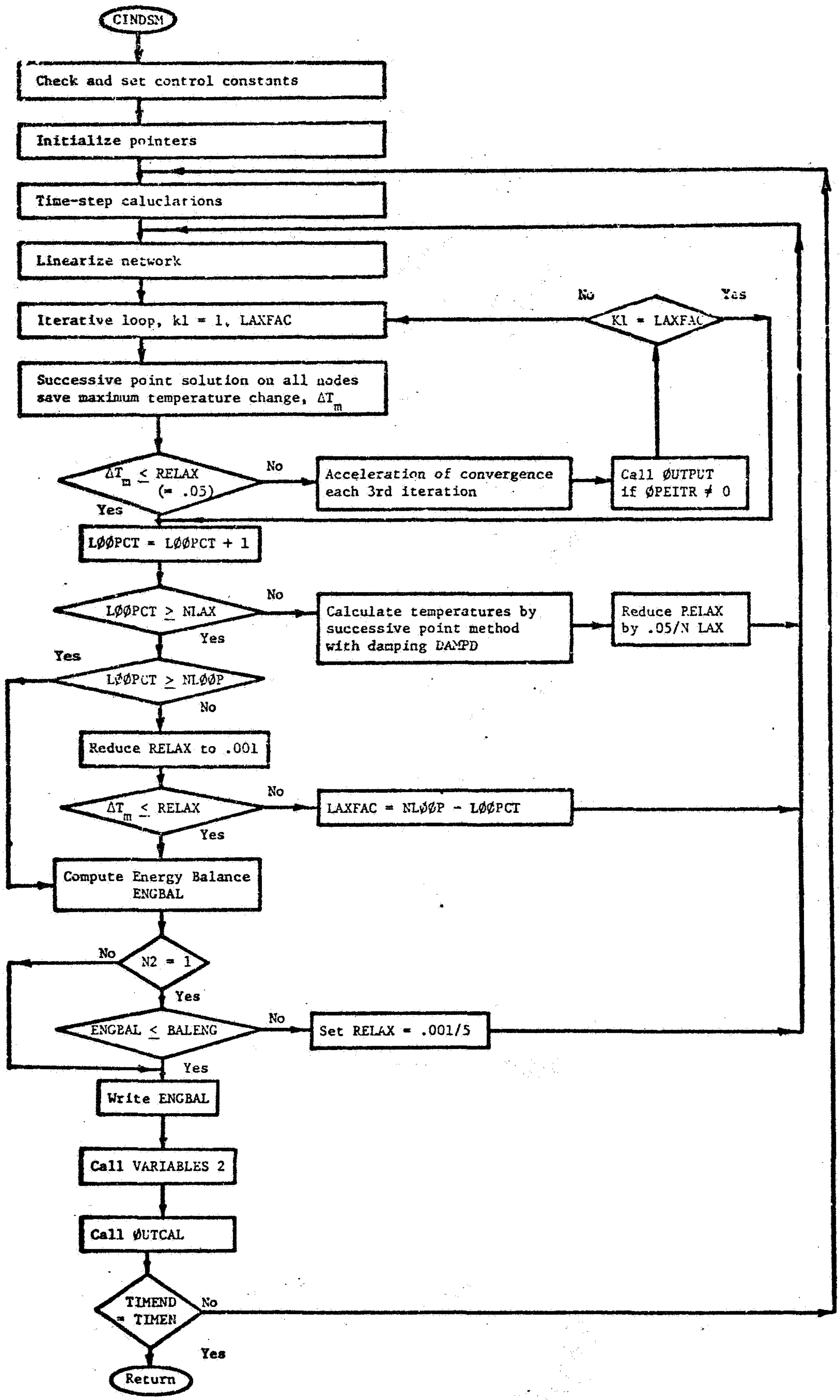
13. If LØØPCT $\geq$ LAXFAC (original user input value), go to step (15)

14. If LOOPCT $\leq$ LAXFAC (original user input value), ENGBAL is checked against BALENG.

    If ENGBAL $\leq$ BALENG, go to step (14)

    If ENGBAL > BALENG, RELAX is set to, RELAX = .001/5, and steps (5) through (14) are repeated with the new RELAX values.

15. Print ENGBAL; call VARIABLES 2; call ØUTCAL; check if TIMEND = TIMEØ.



6.5-3. Functional Flow Chart for CINDSM

7.1 REFERENCES

1. J. D. Gaski and D. R. Lewis, "Chrysler Improved Numerical Differencing Analyzer," TN-AP-66-15, April 30, 1966, Chrysler Corporation Space Division, New Orleans, Louisiana.

2. D. R. Lewis, J. D. Gaski and L. R. Thompson, "Chrysler Improved Numerical Differencing Analyzer for 3rd Generation Computers," TN-AP-67-287, October 20, 1967, Chrysler Corporation Space Division, New Orleans, Louisiana.

3. J. D. Gaski, L. C. Fink and T. Ishimoto, "Systems Improved Numerical Differencing Analyzer, Users Manual," 11027-6003-RO-00, September 1970, TRW Systems Group.

4. J. P. Smith, "SINDA User's Manual," 14690-H001-RO-00, June 1971, TRW Systems Group.

5. TRW Thermal Analyzer Program, Users Manual, Part II, Program Description, CDRC Heat Transfer Group, Program No. 3123.23-31, May 22, 1967.

6. T. Ishimoto, J. D. Gaski and L. C. Fink, "Final Report, Development of Digital Computer Program for Thermal Network Correction; Phase II - Program Development, Phase III - Demonstration/Application," September 1970, 11027-6602-RO-00, TRW Systems Group.

7. W. J. Karplus, <u>Analog Simulation, Solution of Field Problems</u>, McGraw-Hill Book Company, 1958.

8. J. Crank and P. Nicolson, "A Practical Method for Numerical Evaluation of Solutions of Partial Differential Equations of the Heat Conduction Type," <u>Proc. Camb. Phil Soc</u>., vol. 43, 1947, pp. 50-54.

9. E. C. Dufort and S. P. Frankel, "Stability Conditions in the Numerical Treatment of Parabolic Differential Equations," <u>Math. Tables and Other Aids to Computation</u>, vol. 7, pp. 135, 1953.

10. J. Douglas, Jr., "On the Numerical Integration of $\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = \frac{\partial u}{\partial t}$ by Implicit Methods," <u>Journal of the Society of Industrial and Applied Mathematics</u>, 3:42-65, March 1955.

11. J. Douglas, Jr., and D. W. Peaceman, "Numerical Solution of Two-Dimensional Heat Flow Problems," <u>A.I.Ch.E. Journal</u>, vol. 1, No. 4, 1955, pp. 505-512.

12. R. D. Richtmyer, <u>Difference Methods for Initial-Value Problems</u>, Interscience Publishers, Inc., 1957.

13. J. B. Diaz, R. F. Clippinger, B. Friedman, E. Isaacson and R. Richtmyer, "6. Partial Differential Equations," ch. 14, Numerical Analysis, Handbook of Automation, Computation and Control, vol. 1, Control Fundamentals, 1958.

14. G. M. Dusinberre, Heat Transfer Calculations by Finite Differences, International Textbook Company, 1961.

15. G. R. Gaumer, "The Stability of Three Finite Difference Methods of Solving for Transient Temperatures," Fifth U. S. Navy Symposium on Aeroballistics, McDonnell Aircraft Corporation, 18 October 1961.

16. R. S. Varga, "On Higher Order Stable Implicit Methods for Solving Parabolic Partial Differential Equations," Journal of Mathematics and Physics, vol. 40, no. 3, October 1961.

17. B. K. Larkin, "Some Finite Difference Methods for Problems in Transient Heat Flow," A.I.Ch.E Preprint 16, Seventh National Heat Transfer Conference, August 9 to 12, 1964, Cleveland, Ohio.

18. S. R. Allada and D. Quon, "A Stable Explicit Numerical Solution of the Conduction Equation for Multi-Dimensional Non-Homogeneous Media," A.I.Ch.E. Preprint 27, Eighth National Heat Transfer Conference, August 8 to 11, 1965, Los Angeles, California.

19. R. S. Barker and H. M. Stephens, "Investigation of Numerical Methods for Determining Transient Temperatures in Flowing Fluids," A.I.A.A. Fourth Aerospace Sciences Meeting, 27 June 1965, Los Angeles, California, Douglas Paper No. 3823.

20. H. Z. Barakat and J. A. Clark, "On the Solution of the Diffusion Equations by Numerical Methods," Journal of Heat Transfer, November 1966.

21. S. Gross, "An Improved Finite-Difference Method for Heat Transfer Calculation," J. of Spacecraft, vol. 4, no. 4, pp. 538, 1967.

22. B. P. Jones, "A Study of Several Numerical Methods for Solving a Particular System of Ordinary Differential Equations," TMX-53121, NASA/MSFC.

23. G. L. Kusic and A. Lavi, "Stability of Difference Methods for Initial-Value Type Partial Differential Equations," J. of Computational Physics 3, pp. 358-378, 1969.

24. J. H. Smith, "Survey of Three-Dimensional Finite Difference Forms of Heat Equation," SC-M-70-83, Sandia Laboratories, March 1970.

25. O. C. Zienkiewicz, The Finite Element Method in Structural and Continuum Mechanics, McGraw-Hill Publishing Company, pp. 148-169, 1967.

26. W. Visser, "A Finite Element Method for the Determination of Nonstationary Temperature Distribution and Thermal Deformations," Matrix Methods in Structural Mechanics, Proceedings of the Conference held at Wright-Patterson AFB, Ohio, October 26-28, 1965, AFFDL-TR-66-80, November 1966, pp. 925-943.

27. A. F. Emery and W. W. Carson, "Evaluation of Use of the Finite Element Method in Computation of Temperature," A.S.M.E. 69-WA/HT-38, A.S.M.E. Winter Annual Meeting, November 16-20, 1969, Los Angeles, California.

28. E. C. Lemmon and H. S. Heaton, "Accuracy, Stability and Oscillation Characteristics of Finite Element Method for Solving Heat Conduction Equation," 69-WA/HT-35, A.S.M.E. Winter Annual Meeting, November 16-20, 1969, Los Angeles, California.

29. P. D. Richardson and Y. M. Shum, "Use of Finite Element Methods in Solution of Transient Heat Conduction Problems," 69-WA/HT-36, A.S.M.E. Winter Annual Meeting, November 16-20, 1969, Los Angeles, California.

30. H. S. Carslaw and J. C. Jaeger, "Conduction of Heat in Solids," Oxford University Press, 1959.

31. "Temperatures in Structures," NAVORD Report 5558,

    Part 1. General discussion and one-dimension, one layer problem; analytic solutions, C. J. Thorne, 10 July 1957.

    Part 2. One dimension, two- and three-layer plate problems; analytic solutions, C. J. Thorne, 28 March 1958.

    Part 3. Two- and three-layer cylindrical shields; one-space variable, linear, O. H. Strand and C. J. Thorne, 20 August 1958.

    Part 4. Two- and three-layer spherical shell segments, one-space variable, linear, R. E. Smithson and C. J. Thorne, 4 September 1958.

    Part 5. One layer, two-space variables, linear problems, L. C. Barrett, H. J. Fletcher and C. J. Thorne, 27 May 1958.

32. S. S. Abarbanel, "On Some Problems in Radiation Heat Transfer," O.S.R. Technical Report No. 59-531, Massachusetts Institute of of Technology, April 1959.

33. P. L. Chambré, "Nonlinear Heat Transfer Problem," J. of Applied Physics, vol. 20, no. 11, pp. 1683-1688, November 1959.

34. D. C. Stickler and L. S. Han, "Three Linear Approximations to the Stefan-Boltzmann Radiation Problem for the Plane Slab," RADC-TDR-62-557, Supl. 1, Ohio State University, July 1962.

35. R. H. MacNeal, "An Asymmetrical Finite Difference Network," Quarterly Applied Mathematics, vol. 11, pp. 295-310, 1953.

36. T. Ishimoto and J. T. Bevans, "Temperature Variance in Spacecraft Thermal Analysis," TRW Systems Group, A.I.A.A. Paper No. 68-62, J. of Spacecraft, vol. 5, no. 11, pp. 1372-1376, 1968.

37. B. Bussell, "Properties of a Variable Structure Computer System in the Solution of Parabolic Partial Differential Equations," Report No. 62-46, September 1962, University of California, Los Angeles, California.

38. P. F. Strong and A. G. Emslie, "The Method of Zones for the Calculation of Temperature Distribution," A.S.M.E. 65-WA/HT-47, A.S.M.E. Winter Annual Meeting, November 7-11, 1965, Chicago, Illinois.

39. A. L. Edwards, "TRUMP: A Computer Program for Transient and Steady State Temperature Distributions in Multidimensional Systems," TID-4500, UC-32, Lawrence Radiation Laboratory, University of California, Livermore, May 1, 1968.

40. A. K. Oppenheim, "Radiation Analysis by the Network Method," Trans. A.S.M.E., 78, pp. 725-735, 1956.

41. G. L. Polyak, "Radiative Transfer Between Surfaces of Arbitrary Spatial Distribution of Reflection," Translation TT-9, School of Aeronautical and Engineering Sciences, Purdue Univeristy, Lafayette, Indiana; translated from Konvek tionyi i Luchisty i Teploobmen, pp. 118-132, Akad. Nauk. SSSR, Moskva (1960).

42. E. M. Sparrow, E. R. G. Eckert and V. K. Jonsson, "An Enclosure Theory for Radiative Exchange Between Specularly and Diffusively Reflecting Surface," J. of Heat Transfer, Trans. A.S.M.E., 84, Series C, pp. 924-300, 1962.

43. R. P. Bobco, "Gray Surface Radiation Between a Differential Area and Infinite Plane: Nonuniform Local Heat-Flux," ARS Journal, August 1962.

44. E. M. Sparrow, "Heat Radiation Between Simply-Arranged Surfaces Having Different Temperatures and Emissivities," A.I.Ch.E. Journal, March 1962.

45. J. T. Bevans and D. K. Edwards, "Radiation Exchange in an Enclosure with Directional Wall Properties," Trans. A.S.M.E., J. Heat Transfer 87C, pp. 388-396, 1965.

46. J. T. Bevans, T. Ishimoto, B. R. Loya and E. E. Luedke, "Prediction of Space Vehicle Thermal Characteristics," AFFDL-TR-65-139, TRW Systems, October 1965.

47. E. M. Sparrow and R. D. Cess, Radiation Heat Transfer, Brooks/Cole Publishing Company, 1966.

48. R. Viskanta, J. R. Schornhorst and J. S. Toor, "Analysis and Experiment of Radiant Heat Exchange Between Simply Arranged Surfaces," AFFDL-TR-67-94, Purdue University, June 1967.

49. O. W. Clausen and T. Ishimoto, "Thermal Test of a Model Space Vehicle, Part I. Description and Thermal Analysis Model, AFFDL-TR-67-42, Part I, TRW Systems Group, June 1967.

50. R. G. Hering and R. P. Bobco, "Local Radiant Flux and Temperature in Semigray Nondiffuse Enclosures," A.I.A.A. No. 68-60, A.I.A.A. 6th Aerospace Sciences Meeting, New York, New York, January 22-24, 1968.

51. T. Ishimoto and J. T. Bevans, "Method of Evaluating F Variance for Thermal Variance Analysis," A.I.A.A. Journal, vol. 6, no. 6, p. 1178-1180, June 1968.

52. J. R. Howell and R. Siegel, Thermal Radiation Heat Transfer, Volume II, Radiation Exchange Between Surfaces and in Enclosures, NASA SP-164, 169.

53. R. P. Bobco, "Radiation Heat Transfer in Semigray Enclosures with Specularly and Diffusely Reflecting Surfaces, J. Heat Transfer, Trans. A.S.H.E., Ser. C, vol. 86, no. 1, pp. 123-130, February 1964.

54. A. F. Sarofim and H. C. Hottel, "Radiative Exchange Among Non-Lambert Surfaces," J. Heat Transfer, Trans, A.S.M.E., Ser. C, vol. 88, no. 1, February 1966.

55. T. Ishimoto and J. T. Bevans, "Method of Evaluating Script F for Radiant Exchange Within an Enclosure," A.I.A.A. Journal, vol. 1, no. 6, pp. 1428-1429, June 1963.

56. K. A. Toups, "A General Computer Program for the Determination of Radiant Interchange Configurations and Form Factors - CONFAC II," North American Aviation, Inc., Space and Information Systems Division, S.I.D. Report 65-1043-2, October 1965

57. R. P. Bobco, "Analytical Determination of Radiation Interchange Factors," SSD 90190R, Hughes Aircraft Company, June 1969.

58. R. J. McGrath and F. L. Egendorf, "A Computer Program for Determination of Radiation Interchange Factors - RADFAC, User's Manual," SSD 90191R, Hughes Aircraft Company, June 1969.

59. "Measurement of Radiation Interchange Factors, Final Report and Facility Design Description," Thermophysics Department Staff, Project Manager, R. P. Bobco, SSD 00577R, NASA Contract NAS 9-10271, Nov. 1970.

A. COMPUTER LISTINGS OF SINDA EXPLICIT SOLUTION ROUTINES

```
                    CNFRWD.CNFRWD
QIW   FOR,*         CNFRWD,CNFRWD                                                             21 FEB 71
UNIVAC 1108 FORTRAN V ATHENA VERSION 131K-10D   CREATED ON 20 AUG 70
THIS COMPILATION WAS DONE ON 09 JUN 70 AT 14:00:37

   SUBROUTINE CNFRWD    ENTRY POINT 003522

   STORAGE USED (BLOCK, NAME, LENGTH)

         0001    *CODE          003535
         0000    *CONST+TEMP    000072
         0002    *SIMPLE VAR    000050
         0004    *ARRAYS        000000
         0005    *BLANK         000000
         0006    TITLE    000001
         0007    TEMP     000001
         0010    CAP      000001
         0011    SOURCE   000001
         0012    COND     000001
         0013    PC1      000001
         0014    PC2      000001
         0015    KONST    000001
         0016    ARRAY    000001
         0017    FIXCON   000001
         0020    XSPACE   000003
         0021    DIMENS   000010

   EXTERNAL REFERENCES (BLOCK, NAME)

         0022    VARBL1
         0023    D1D1WM
         0024    PLYAWM
         0025    D2D1WM
         0026    VARBL2
         0027    OUTCAL
         0030    EXIT
         0031    NERR2$
         0032    NWDU$
         0033    NI02$
         0034    NER10$

   STORAGE ASSIGNMENT FOR VARIABLES (BLOCK, TYPE, RELATIVE LOCATION, NAME)

    0016 R 000000 A        0010 R 000000 C        0002 R 000015 CKM      0017 R 000000 CON      0002 R 000021 C1
    0002 R 000022 C2       0002 R 000041 DD       0002 R 000037 DELTA    0002 R 000040 DN       0012 R 000000 G
    0002 R 000034 GV       0002 R 000030 G1       0002 R 000031 G2       0006 R 000000 H        0002 I 000004 I
    0002 I 000002 IE       0002 I 000043 JJ1      0002 I 000044 JJ2      0002 I 000012 J1       0002 I 000013 J2
    0015 I 000000 K        0017 I 000000 KON      0002 I 000045 L        0002 I 000017 LA       0002 I 000042 LAX
    0002 I 000011 LE       0002 I 000036 LEA      0002 I 000025 LG       0002 I 000020 LK       0021 I 000006 LSQ1
    0021 I 000007 LSQ2     0002 I 000026 LTA      0002 I 000005 L1       0021 I 000005 NAT      0021 I 000004 NCT
    0020 I 000000 NDIM     0021 I 000003 NGT      0002 I 000003 NLA      0021 I 000001 NNA      0002 I 000001 NNC
    0021 I 000000 NND      0021 I 000002 NNT      0013 I 000000 NSQ1     0014 I 000000 NSQ2     0020 I 000001 NTH
    0002 I 000016 NTYPE    0020   000002 NX       0002 R 000000 PASS     0011 R 000000 Q        0002 R 000035 QDOT
    0002 R 000023 Q1       0002 R 000024 Q2       0002 R 000046 SUMC     0002 R 000047 SUMCV    0007 R 000000 T
    0002 R 000014 TCGM     0002 R 000027 TM       0002 R 000007 TPRINT   0002 R 000006 TSTEP    0002 R 000010 TSUM
    0002 R 000032 T1       0002 R 000033 T2       0020 R 000002 X        0015 R 000000 XK       0001   000125 10L
```

CNFRWD,CNFRWD

```
0001  003500 1000L    0001  000327 1005L    0001  000350 1010L    0001  003336 1011G    0001  000366 1012L
0001  000421 1015L    0001  000427 1020L    0001  000467 1025L    0001  000513 1030L    0301  000534 1032L
0001  000572 1035L    0001  000600 1040L    0001  000643 1045L    0001  002071 105L     0001  002123 115L
0001  002134 120L     0001  002142 125L     0001  002150 130L     0001  002500 135L     0001  003246 140L
0001  003251 145L     0001  000134 15L      0001  003316 165L     0001  003327 175L     0001  003360 180L
0001  003371 185L     0001  003420 190L     0001  003423 195L     0001  000665 1998L    0001  000670 1999L
0001  000206 200G     0001  000673 2000L    0001  001235 2005L    0001  001242 2007L    0001  001262 2010L
0001  001265 2015L    0001  001303 2017L    0001  001337 2020L    0001  001345 2025L    0001  001405 2030L
0001  001412 2032L    0001  001435 2035L    0001  001440 2040L    0001  001461 2042L    0001  001520 2045L
0001  001526 2050L    0001  001571 2055L    0001  001617 2060L    0001  001623 2065L    0001  000225 212G
0001  000256 232G     0001  000146 25L      0001  001627 2998L    0001  001655 2999L    0001  000161 30L
0001  001660 3000L    0001  002566 3005L    0001  002573 3007L    0001  002613 3010L    0001  002616 3015L
0001  002634 3017L    0001  002670 3020L    0001  002676 3025L    0001  002736 3030L    0001  002743 3032L
0001  002766 3035L    0001  002771 3040L    0001  003012 3042L    0001  003051 3045L    0001  003057 3050L
0001  003122 3055L    0001  003150 3060L    0001  003154 3065L    0001  003160 3998L    0001  003206 3999L
0001  003211 4000L    0001  000740 4005L    0001  000745 4010L    0001  000746 4012L    0001  000765 4015L
0001  000766 4017L    0001  001003 4020L    0001  001017 4022L    0001  001034 4025L    0001  001042 4030L
0001  001100 4035L    0001  001114 4037L    0001  001131 4040L    0001  000237 45L      0001  001137 4998L
0001  001143 4999L    0001  000113 5L       0001  001146 50L      0001  001146 5000L    0001  002272 5005L
0001  002277 5010L    0001  002300 5012L    0001  002317 5015L    0001  002320 5017L    0001  002335 5020L
0001  002351 5022L    0001  002366 5025L    0001  002374 5030L    0001  002432 5035L    0001  002446 5037L
0001  002004 504G     0001  002463 5040L    0001  002075 535G     0001  001715 55L      0001  002177 573G
0001  002471 5998L    0001  002475 5999L    0001  001720 60L      0001  002500 6000L    0001  002220 602G
0001  001767 65L      0001  001776 85L      0000  000000 885F     0000  000011 886F     0000  000016 887F
0000  000023 888F     0000  000030 889F     0001  002025 90L      0001  002044 95L      0001  003442 995L
0001  003450 996L     0001  003456 997L     0001  003464 998L     0001  003473 999L
```

```
00101   1*        SUBROUTINE CNFRWD
00101   2*   C    EXPLICIT FORWARD DIFFERENCING EXECUTION SUBROUTINE FOR SINDA F-V
00101   3*   C    THE SHORT PSEUDO-COMPUTE SEQUENCE IS REQUIRED
00103   4*        INCLUDE COMM,LIST
00104   4*        COMMON /TITLE/H(1) /TEMP/T(1) /CAP/C(1) /SOURCE/Q(1) /COND/G(1)
00105   4*        COMMON /PC1/NSQ1(1) /PC2/NSQ2(1) /KONST/K(1) /ARRAY/A(1)
00106   4*        COMMON /FIXCON/KON(1) /XSPACE/NDIM,NTH,X(1)
00107   4*        COMMON /DIMENS/ NND,NNA,NNT,NGT,NCT,NAT,LSQ1,LSQ2
00110   4*        DIMENSION CON(1),XK(1),NX(1)
00111   4*        EQUIVALENCE (KON(1),CON(1)),(K(1),XK(1)),(X(1),NX(1))
00112   4*        END
00113   5*        INCLUDE DEFF,LIST
00113   5*   C*************  CONTROL CONSTANT DEFINITIONS AND NAMES  *****************
00113   5*   C    CONTROL CONSTANT 1 CONTAINS THE NEW PROBLEM TIME              (TIMEN)
00113   5*   C    CONTROL CONSTANT 2 CONTAINS THE TIME STEP USED               (DTIMEU)
00113   5*   C    CONTROL CONSTANT 3 CONTAINS THE PROBLEM STOP TIME            (TIMEND)
00113   5*   C    CONTROL CONSTANT 4 CONTAINS THE TIME STEP FACTOR,EXPLICIT (CSGFAC)
00113   5*   C    CC5 IS THE INPUT NUMBER OF ITERATION DO LOOPS, INTEGER        (NLOOP)
00113   5*   C    CC6 CONTAINS THE DIFFUSION TEMPERATURE CHANGE ALLOWED        (DTMPCA)
00113   5*   C    CC7 CONTAINS THE OUTPUT EACH ITERATION SWITCH                (OPEITR)
00113   5*   C    CC8 CONTAINS THE MAXIMUM ALLOWED TIME STEP                   (DTIMEH)
00113   5*   C    CC9 CONTAINS THE NEW ARITHMETIC TEMP. DAMPING FACTOR          (DAMPA)
00113   5*   C    CC10 CONTAINS THE NEW DIFFUSION TEMP. DAMPING FACTOR          (DAMPD)
00113   5*   C    CC11 CONTAINS THE MAXIMUM ALLOWED ARITHMETIC TEMP. CHANGE (ATMPCA)
00113   5*   C    CC12 CONTAINS THE BACKUP SWITCH CHECKED AFTER VARIABLES      (BACKUP)
00113   5*   C    CC13 CONTAINS THE PRESENT TIME OR PROBLEM START TIME          (TIMEO)
00113   5*   C    CC14 CONTAINS THE MEAN TIME BETWEEN AN ITERATION              (TIMEM)
00113   5*   C    CC15 CONTAINS THE DIFFUSION TEMPERATURE CHANGE CALCULATED (DTMPCC)
00113   5*   C    CC16 CONTAINS ARITHMETIC TEMPERATURE CHANGE CALCULATED      (ATMPCC)
00113   5*   C    CONTROL CONSTANT 17 IS RESERVED FOR THE C/SG MINIMUM        (CSGMIN)
```

```
00113    5*  C     CONTROL CONSTANT 18 CONTAINS THE OUTPUT INTERVAL          (OUTPUT)
00113    5*  C     CC19 CONTAINS THE ARITHMETIC RELAXATION CRITERIA ALLOWED  (ARLXCA)
00113    5*  C     CC20 CONTAINS THE NUMBER OF RELAXATION LOOPS USED,INTEGER (LOOPCT)
00113    5*  C     CC21 CONTAINS THE MINIMUM ALLOWED TIME STEP               (DTIMEL)
00113    5*  C     CC22 IS FOR THE INPUT TIME STEP IMPLICIT                  (DTIMEI)
00113    5*  C     CC23 CONTAINS THE C/SG MAXIMUM                            (CSGMAX)
00113    5*  C     CC24 CONTAINS THE C/SG RANGE ALLOWED                      (CSGRAL)
00113    5*  C     CC25 CONTAINS THE C/SG RANGE CALCULATED                   (CSGRCL)
00113    5*  C     CC26 CONTAINS THE DIFFUSION RELAXATION CRITERIA ALLOWED   (DRLXCA)
00113    5*  C     CC27 CONTAINS THE DIFFUSION RELAXATION CHANGE CALCULATED  (DRLXCC)
00113    5*  C     CC28 CONTAINS THE LINE COUNTER, INTEGER                   (LINECT)
00113    5*  C     CC29 CONTAINS THE PAGE COUNTER, INTEGER                   (PAGECT)
00113    5*  C     CC30 CONTAINS ARITHMETIC RELAXATION CHANGE CALCULATED     (ARLXCC)
00113    5*  C     CC31 IS INDICATOR, 0=THERMAL SPCS,1=THERMAL LPCS,2=GENERAL (LSPCS)
00113    5*  C     CC32 CONTAINS THE ENERGY BALANCE OF THE SYSTEM, IN - OUT  (ENGBAL)
00113    5*  C     CC33 CONTAINS THE DESIRED ENERGY BALANCE, USER INPUT      (BALENG)
00113    5*  C     CC34 CONTAINS THE NOCOPY SWITCH FOR MATRIX USERS          (NOCOPY)
00113    5*  C     CC35 CONTAINS RELATIVE NODE NUMBER OF CSGMIN
00113    5*  C     CC36 CONTAINS RELATIVE NODE NUMBER OF DTMPCC
00113    5*  C     CC37 CONTAINS RELATIVE NODE NUMBER OF ARLXCC
00113    5*  C     CC38 CONTAINS RELATIVE NODE NUMBER OF ATMPCC
00113    5*  C     CC39-40-41-42-43 CONTAIN DUMMY INTEGER CONSTANTS   (I-J-K-L-MTEST)
00113    5*  C     CC44-45-46-47-48 CONTAIN DUMMY FLOATING CONSTANTS  (R-S-T-U-VTEST)
00113    5*  C     CC49 IS THE QUASI-LINEARIZATION INTERVAL FOR CINDSM       (LAXFAC)
00113    5*  C     CC50 IS NOT USED AT PRESENT
00114    5*        END
00115    6*        IF(CON(4).LT.1.0) CON(4) = 1.0
00117    7*        IF(KON(5).LE.0) KON(5) = 1
00121    8*        IF(CON(6).LE.0.) CON(6) = 1.E+8
00123    9*        IF(CON(8).LE.0.) CON(8) = 1.E+8
00125   10*        IF(CON(9).LE.0.) CON(9) = 1.0
00127   11*        IF(CON(11).LE.0.) CON(11) = 1.E+8
00131   12*        IF(CON(18).LE.0.) GO TO 999
00133   13*        IF(CON(19).LE.0.) CON(19) = 1.E+8
00135   14*        IF(KON(31).NE.0) GO TO 995
00137   15*        PASS = -1.0
00140   16*        NNC = NND+NNA
00141   17*        IE = NTH
00142   18*        NLA = NDIM
00143   19*        NTH = NTH+NNC
00144   20*        NDIM = NDIM-NNC
00144   21*  C     CHECK FOR EXTRA LOCATIONS FOR CALCULATED NODES
00145   22*        I = NLA-NNC
00146   23*        IF(I.LT.0) GO TO 998
00150   24*        L1 = NND+1
00151   25*        TSTEP = CON(18)
00152   26*        TPRINT = CON(13)
00152   27*  C     INITALIZE TIME SUM BETWEEN OUTPUT INTERVALS
00153   28*      5 TSUM = 0.0
00153   29*  C     DOES OLD TIME PLUS THE OUTPUT INTERVAL EXCEED THE STOP TIME
00154   30*        IF(CON(13)+CON(18).LE.CON(3)) GO TO 10
00154   31*  C     DONT EXCEED IT
00156   32*        CON(18) = CON(3)-CON(13)
00156   33*  C     IS THE TIME STEP LARGER THAN ALLOWED
00157   34*     10 IF(TSTEP.LE.CON(8)) GO TO 15
00161   35*        TSTEP = CON(8)
00161   36*  C     DOES THE TIME SUM PLUS THE TIME STEP EXCEED OUTPUT INTERVAL
00162   37*     15 IF(TSUM+TSTEP-CON(18)) 25,30,20
00162   38*  C     DONT EXCEED IT
00165   39*     20 TSTEP = CON(18)-TSUM
```

CNFRWD.CNFRWD

```
00166   40*          GO TO 30
00166   41*    C     DOES TIME SUM PLUS TWO TIME STEPS EXCEED OUTPUT INTERVAL
00167   42*       25 IF(TSUM+2.0*TSTEP.LE.CON(18)) GO TO 30
00167   43*    C     APPROACH THE OUTPUT INTERVAL GRADUALLY
00171   44*          TSTEP = (CON(18)-TSUM)/2.0
00171   45*    C     STORE DELTA TIME STEP IN THE CONSTANTS
00172   46*       30 CON(2) = TSTEP
00172   47*    C     IS THE TIME STEP USED LESS THAN THE TIME STEP ALLOWED
00173   48*          IF(TSTEP.LT.CON(21)) GO TO 997
00173   49*    C     CALCULATE THE NEW TIME
00175   50*          CON(1) = TPRINT+TSUM+TSTEP
00175   51*    C     COMPUTE THE MEAN TIME BETWEEN ITERATIONS
00176   52*          CON(14) = (CON(1)+CON(13))/2.0
00176   53*    C     ZERO OUT ALL SOURCE LOCATIONS AND EXTRA LOCATIONS
00177   54*          DO 35 I = 1,NND
00202   55*          LE = IE+I
00203   56*          X(LE) = 0.0
00204   57*          Q(I) = 0.0
00205   58*       35 CONTINUE
00205   59*    C     SHIFT THE ARITHMETIC TEMPERATURES INTO THE EXTRA LOCATIONS
00207   60*          IF(NNA.LE.0) GO TO 45
00211   61*          DO 40 I = L1,NNC
00214   62*          Q(I) = 0.0
00215   63*          LE = IE+I
00216   64*          X(LE) = T(I)
00217   65*       40 CONTINUE
00221   66*       45 KON(12) = 0
00222   67*          CALL VARBL1
00223   68*          IF(KON(12).NE.0) GO TO 10
00225   69*          J1 = 0
00226   70*          J2 = 1
00227   71*          TCGM = 0.0
00230   72*          CKM = 1.E+8
00230   73*    C     CALCULATE Q SUM AND G SUM
00231   74*          DO 65 I = 1,NND
00234   75*          LE = IE+I
00235   76*          INCLUDE VARC,LIST
00236   76*          IF(FLD(1,1,NSQ1(J1+1)).EQ.0) GO TO 2000
00240   76*          NTYPE = FLD(0,5,NSQ2(J2))                                       *NEW
00241   76*          LA = FLD(5,17,NSQ2(J2))                                         *NEW
00242   76*          LK = FLD(22,14,NSQ2(J2))                                        *NEW
00243   76*          GO TO (1005,1010,1015,1020,1025,1030,1035,1040,1045), NTYPE     **-3
00244   76*     1005 CALL D1D1WM(T(I),A(LA),XK(LK),C(I))
00245   76*          GO TO 1999
00246   76*     1010 CALL D1D1WM(T(I),A(LA),XK(LK),C1)
00247   76*     1012 J2 = J2+1
00250   76*          LA = FLD(5,17,NSQ2(J2))                                         *NEW
00251   76*          LK = FLD(22,14,NSQ2(J2))                                        *NEW
00252   76*          CALL D1D1WM(T(I),A(LA),XK(LK),C2)                               **-2
00253   76*          GO TO 1998
00254   76*     1015 C1 = XK(LK)*XK(LA)
00255   76*          GO TO 1012
00256   76*     1020 CALL D1D1WM(T(I),A(LA),XK(LK),C1)
00257   76*          J2 = J2+1
00260   76*          LA = FLD(5,17,NSQ2(J2))                                         *NEW
00261   76*          LK = FLD(22,14,NSQ2(J2))                                        *NEW
00262   76*          C2 = XK(LK)*XK(LA)                                              **-2
00263   76*          GO TO 1998
00264   76*     1025 CALL PLYAWM(A(LA),T(I),A(LA+1),XK(LK),C(I))
00265   76*          GO TO 1999
```

```
76*      1030 CALL PLYAWM(A(LA),T(I),A(LA+1),XK(LK),C1)
76*      1032 J2 = J2+1
76*           LA = FLD(5,17,NSQ2(J2))                                        *NEW
76*           LK = FLD(22,14,NSQ2(J2))                                       *NEW
76*           CALL PLYAWM(A(LA),T(I),A(LA+1),XK(LK),C2)                      **-2
76*           GO TO 1998
76*      1035 C1 = XK(LK)*XK(LA)
76*           GO TO 1032
76*      1040 CALL PLYAWM(A(LA),T(I),A(LA+1),XK(LK),C1)
76*           J2 = J2+1
76*           LA = FLD(5,17,NSQ2(J2))                                        *NEW
76*           LK = FLD(22,14,NSQ2(J2))                                       *NEW
76*           C2 = XK(LK)*XK(LA)                                             **-2
76*           GO TO 1998
76*      1045 CALL D2D1WM(T(I),CON(14),A(LA),XK(LK),C(I))
76*           GO TO 1999
76*      1998 C(I) = C1+C2
76*      1999 J2 = J2+1
76*      2000 CONTINUE
76*           END
77*           INCLUDE VARQ,LIST
77*           IF(FLD(4,1,NSQ1(J1+1)).EQ.0) GO TO 5000
77*           NTYPE = FLD(0,5,NSQ2(J2))                                      *NEW
77*           LA = FLD(5,17,NSQ2(J2))                                        *NEW
77*           LK = FLD(22,14,NSQ2(J2))                                       *NEW
77*           GO TO (4005,4010,4015,4020,4025,4030,4035,4040,4030), NTYPE    **-3
77*      4005 Q(I) = XK(LK)+Q(I)
77*           GO TO 4999
77*      4010 Q1 = 0.0
77*      4012 CALL D1D1WM(T(I),A(LA),XK(LK),Q2)
77*           GO TO 4998
77*      4015 Q1 = 0.0
77*      4017 CALL D1D1WM(CON(14),A(LA),XK(LK),Q2)
77*           GO TO 4998
77*      4020 CALL D1D1WM(CON(14),A(LA),XK(LK),Q1)
77*      4022 J2 = J2+1
77*           LA = FLD(5,17,NSQ2(J2))                                        *NEW
77*           LK = FLD(22,14,NSQ2(J2))                                       *NEW
77*           GO TO 4017                                                     **-2
77*      4025 Q1 = XK(LK)*XK(LA)
77*           GO TO 4022
77*      4030 CALL D1D1WM(CON(14),A(LA),XK(LK),Q1)
77*           J2 = J2+1
77*           LA = FLD(5,17,NSQ2(J2))                                        *NEW
77*           LK = FLD(22,14,NSQ2(J2))                                       *NEW
77*           Q2 = XK(LK)*XK(LA)                                             **-2
77*           GO TO 4998
77*      4035 CALL D1D1WM(CON(14),A(LA),XK(LK),Q1)
77*      4037 J2 = J2+1
77*           LA = FLD(5,17,NSQ2(J2))                                        *NEW
77*           LK = FLD(22,14,NSQ2(J2))                                       *NEW
77*           GO TO 4012                                                     **-2
77*      4040 Q1 = XK(LK)*XK(LA)
77*           GO TO 4037
77*      4998 Q(I) = Q1+Q2+Q(I)
77*      4999 J2 = J2+1
77*      5000 CONTINUE
77*           END
78*        50 J1 = J1+1
79*           LG = FLD(5,16,NSQ1(J1))
```

CNFRWD.CNFRWD

```
80*   C      CHECK FOR LAST CONDUCTOR
81*          IF(LG.EQ.0) GO TO 85
82*          LTA = FLD(22,14,NSQ1(J1))
83*          INCLUDE VARG,LIST
84*   C      CHECK FOR RADIATION CONDUCTOR
84*          IF(FLD(2,1,NSQ1(J1)).EQ.0) GO TO 3000
84*          NTYPE = FLD(0,5,NSQ2(J2))                                              *NEW
84*          LA = FLD(5,17,NSQ2(J2))                                                *NEW
84*          LK = FLD(22,14,NSQ2(J2))                                               *NEW
84*          GOTO(2005,2010,2015,2020,2025,2030,2035,2040,2045,2050,2055,           **-3
84*         $     2060,2065), NTYPE
84*     2005 TM = (T(I)+T(LTA))/2.0
84*     2007 CALL D1D1WM(TM,A(LA),XK(LK),G(LG))
84*          GO TO 2999
84*     2010 TM = T(I)
84*          GO TO 2007
84*     2015 CALL D1D1WM(T(I),A(LA),XK(LK),G1)
84*     2017 J2 = J2+1
84*          LA = FLD(5,17,NSQ2(J2))                                                *NEW
84*          LK = FLD(22,14,NSQ2(J2))                                               *NEW
84*          CALL D1D1WM(T(LTA),A(LA),XK(LK),G2)                                    **-2
84*          GO TO 2998
84*     2020 G1 = XK(LK)*XK(LA)
84*          GO TO 2017
84*     2025 CALL D1D1WM(T(I),A(LA),XK(LK),G1)
84*          J2 = J2+1
84*          LA = FLD(5,17,NSQ2(J2))                                                *NEW
84*          LK = FLD(22,14,NSQ2(J2))                                               *NEW
84*          G2 = XK(LK)*XK(LA)                                                     **-2
84*          GO TO 2998
84*     2030 TM = (T(I)+T(LTA))/2.0
84*     2032 CALL PLYAWM(A(LA),TM,A(LA+1),XK(LK),G(LG))
84*          GO TO 2999
84*     2035 TM = T(I)
84*          GO TO 2032
84*     2040 CALL PLYAWM(A(LA),T(I),A(LA+1),XK(LK),G1)
84*     2042 J2 = J2+1
84*          LA = FLD(5,17,NSQ2(J2))                                                *NEW
84*          LK = FLD(22,14,NSQ2(J2))                                               *NEW
84*          CALL PLYAWM(A(LA),T(LTA),A(LA+1),XK(LK),G2)                            **-2
84*          GO TO 2998
84*     2045 G1 = XK(LK)*XK(LA)
84*          GO TO 2042
84*     2050 CALL PLYAWM(A(LA),T(I),A(LA+1),XK(LK),G1)
84*          J2 = J2+1
84*          LA = FLD(5,17,NSQ2(J2))                                                *NEW
84*          LK = FLD(22,14,NSQ2(J2))                                               *NEW
84*          G2 = XK(LK)*XK(LA)                                                     **-2
84*          GO TO 2998
84*     2055 TM = (T(I)+T(LTA))/2.0
84*          CALL D2D1WM(TM,CON(14),A(LA),XK(LK),G(LG))
84*          GO TO 2999
84*     2060 TM = T(LTA)
84*          GO TO 2007
84*     2065 TM = T(LTA)
84*          GO TO 2032
84*     2998 G(LG) = 1./(1./G1+1./G2)
84*          IF(FLD(3,1,NSQ1(J1)).EQ.1) G(LG) = G1*G2
84*     2999 J2 = J2+1
84*     3000 CONTINUE
```

CNFRWD.CNFRWD

```
00457            END
00460            IF(FLD(3,1,NSQ1(J1)).EQ.0) GO TO 55
00462            T1 = T(I)+460.0
00463            T2 = T(LTA)+460.0
00464            GV = G(LG)*(T1*T1+T2*T2)*(T1+T2)
00465            GO TO 60
00466         55 GV = G(LG)
00466      C     OBTAIN THE Q RATE THRU THE CONDUCTOR
00467         60 QDOT = GV*(T(LTA)-T(I))
00470            Q(I) = Q(I)+QDOT
00470      C     SAVE SUMMATION OF CONDUCTORS
00471            X(LE) = X(LE)+GV
00471      C     CHECK FOR ADJOINING DIFFUSION NODE
00472            IF(LTA.GT.NND.OR.FLD(2,1,NSQ1( [illegible]
00472      C     SAVE SUMMATION OF CONDUCTORS FOR ADJOINING NODE
00474            LEA = IE+LTA
00475            X(LEA) = X(LEA)+GV
00476            Q(LTA) = Q(LTA)-QDOT
00476      C     CHECK FOR LAST CONDUCTOR
00477         65 IF(NSQ1(J1).GT.0) GO TO 50
00501         85 CONTINUE
00501      C     OBTAIN NEW DIFFUSION TEMPERATURES, DTMPCC AND CSGMIN
00503            DO 100 I = 1,NND
00506            LE = IE+I
00506      C     CALCULATE C/SK MINIMUM
00507            T1 = C(I)/X(LE)
00510            IF(T1.GE.CKM) GO TO 90
00512            CKM = T1
00513            KON(35) = I
00513      C     COMPUTE NEW TEMPERATURES USING CALCULATED SOURCE TERMS
00514         90 T1 = TSTEP*Q(I)/C(I)
00514      C     CALCULATE THE ABSOLUTE VALUE TEMPERATURE CHANGE
00515            T2 = ABS(T1)
00515      C     SAVE THE LARGEST TEMPERATURE CHANGE
00516            IF(TCGM.GE.T2) GO TO 95
00520            TCGM = T2
00521            KON(36) = I
00521      C     STORE THE TEMPERATURES
00522         95 X(LE) = T(I)
00523            T(I) = T(I)+T1
00524        100 CONTINUE
00526            CON(17) = CKM
00527            DELTA = CKM/CON(4)
00530            IF(CKM.LE.0.0) GO TO 996
00530      C     CHECK FOR FIRST PASS
00532            IF(PASS.GT.0.0) GO TO 115
00532      C     UNDO THE TEMPERATURE CALCULATIONS
00534        105 DO 110 I = 1,NNC
00537            LE = IE+I
00540            T(I) = X(LE)
00541        110 CONTINUE
00543            IF(PASS.GT.0.0) GO TO 15
00545            PASS = 1.0
00546            CON(1) = TPRINT
00547            CON(2) = 0.0
00550            TSTEP = DELTA*0.95
00551            GO TO 195
00551      C     IS THE TIME STEP USED LESS THAN THE TIME STEP CALCULATED
00552        115 IF(TSTEP.LE.DELTA) GO TO 130
00552      C     COMPUTE THE TIME STEP
```

CNFRWD.CNFRWD

```
00554  144*          TSTEP = DELTA*0.95
00555  145*          GO TO 105
00556  146*      120 TSTEP = 0.95*TSTEP*CON(6)/TCGM
00557  147*          GO TO 105
00560  148*      125 TSTEP = 0.95*TSTEP*CON(11)/TCGM
00561  149*          GO TO 105
00561  150*     C    SEE IF THE TEMPERATURE CHANGE WAS TOO LARGE
00562  151*      130 IF(TCGM.GT.CON(6)) GO TO 120
00562  152*     C    STORE THE MAXIMUM DIFFUSION TEMPERATURE CHANGE
00564  153*          CON(15) = TCGM
00564  154*     C    CHECK TO SEE IF THERE ARE ANY ARITHMETIC NODES
00565  155*          IF(NNA.LE.0) GO TO 185
00565  156*     C    COMPUTE ARITHMETIC TEMPERATURES BY SUCCESSIVE POINT OVER-RELAX
00567  157*          DN = CON(9)
00570  158*          DD = 1.0-DN
00571  159*          LAX = KON(5)
00572  160*          DO 170 I = 1,LAX
00575  161*          JJ1 = J1
00576  162*          JJ2 = J2
00577  163*          TCGM = 0.0
00600  164*          KON(20) = I
00601  165*          DO 165 L = L1,NNC
00604  166*          SUMC = 0.0
00605  167*          SUMCV = 0.0
00606  168*          IF(I.GT.1) GO TO 6000
00610  169*          INCLUDE VRQ2,LIST
00611  169*          IF(FLD(4,1,NSQ1(JJ1+1)).EQ.0) GO TO 6000
00613  169*          NTYPE = FLD(0,5,NSQ2(JJ2))                                          *NEW
00614  169*          LA = FLD(5,17,NSQ2(JJ2))                                            *NEW
00615  169*          LK = FLD(22,14,NSQ2(JJ2))                                           *NEW
00616  169*          GO TO (5005,5010,5015,5020,5025,5030,5035,5040,5030), NTYPE         **-3
00617  169*     5005 Q(L) = XK(LK)+Q(L)
00620  169*          GO TO 5999
00621  169*     5010 Q1 = 0.0
00622  169*     5012 CALL D1D1WM(T(L),A(LA),XK(LK),Q2)
00623  169*          GO TO 5998
00624  169*     5015 Q1 = 0.0
00625  169*     5017 CALL D1D1WM(CON(14),A(LA),XK(LK),Q2)
00626  169*          GO TO 5998
00627  169*     5020 CALL D1D1WM(CON(14),A(LA),XK(LK),Q1)
00630  169*     5022 JJ2 = JJ2+1
00631  169*          LA = FLD(5,17,NSQ2(JJ2))                                            *NEW
00632  169*          LK = FLD(22,14,NSQ2(JJ2))                                           *NEW
00633  169*          GO TO 5017                                                          **-2
00634  169*     5025 Q1 = XK(LK)*XK(LA)
00635  169*          GO TO 5022
00636  169*     5030 CALL D1D1WM(CON(14),A(LA),XK(LK),Q1)
00637  169*          JJ2 = JJ2+1
00640  169*          LA = FLD(5,17,NSQ2(JJ2))                                            *NEW
00641  169*          LK = FLD(22,14,NSQ2(JJ2))                                           *NEW
00642  169*          Q2 = XK(LK)*XK(LA)                                                  **-2
00643  169*          GO TO 5998
00644  169*     5035 CALL D1D1WM(CON(14),A(LA),XK(LK),Q1)
00645  169*     5037 JJ2 = JJ2+1
00646  169*          LA = FLD(5,17,NSQ2(JJ2))                                            *NEW
00647  169*          LK = FLD(22,14,NSQ2(JJ2))                                           *NEW
00650  169*          GO TO 5012                                                          **-2
00651  169*     5040 Q1 = XK(LK)*XK(LA)
00652  169*          GO TO 5037
00653  169*     5996 Q(L) = Q1+Q2+Q(L)
```

```
 5999 JJ2 = JJ2+1
 6000 CONTINUE
      END
  135 JJ1 = JJ1+1
      LG = FLD(5,16,NSQ1(JJ1))
      LTA = FLD(22,14,NSQ1(JJ1))
      IF(I.GT.1) GO TO 4000
      INCLUDE VRG2,LIST
C     CHECK FOR RADIATION CONDUCTOR
      IF(FLD(2,1,NSQ1(JJ1)).EQ.0) GO TO 4000
      NTYPE = FLD(0,5,NSQ2(JJ2))                                                  *NEW
      LA = FLD(5,17,NSQ2(JJ2))                                                    *NEW
      LK = FLD(22,14,NSQ2(JJ2))                                                   *NEW
      GOTO(3005,3010,3015,3020,3025,3030,3035,3040,3045,3050,3055,                **-3
     $     3060,3065), NTYPE
 3005 TM = (T(L)+T(LTA))/2.0
 3007 CALL D1D1WM(TM,A(LA),XK(LK),G(LG))
      GO TO 3999
 3010 TM = T(L)
      GO TO 3007
 3015 CALL D1D1WM(T(L),A(LA),XK(LK),G1)
 3017 JJ2 = JJ2+1
      LA = FLD(5,17,NSQ2(JJ2))                                                    *NEW
      LK = FLD(22,14,NSQ2(JJ2))                                                   *NEW
      CALL D1D1WM(T(LTA),A(LA),XK(LK),G2)                                         **-2
      GO TO 3998
 3020 G1 = XK(LK)*XK(LA)
      GO TO 3017
 3025 CALL D1D1WM(T(L),A(LA),XK(LK),G1)
      JJ2 = JJ2+1
      LA = FLD(5,17,NSQ2(JJ2))                                                    *NEW
      LK = FLD(22,14,NSQ2(JJ2))                                                   *NEW
      G2 = XK(LK)*XK(LA)                                                          **-2
      GO TO 3998
 3030 TM = (T(L)+T(LTA))/2.0
 3032 CALL PLYAWM(A(LA),TM,A(LA+1),XK(LK),G(LG))
      GO TO 3999
 3035 TM = T(L)
      GO TO 3032
 3040 CALL PLYAWM(A(LA),T(L),A(LA+1),XK(LK),G1)
 3042 JJ2 = JJ2+1
      LA = FLD(5,17,NSQ2(JJ2))                                                    *NEW
      LK = FLD(22,14,NSQ2(JJ2))                                                   *NEW
      CALL PLYAWM(A(LA),T(LTA),A(LA+1),XK(LK),G2)                                 **-2
      GO TO 3998
 3045 G1 = XK(LK)*XK(LA)
      GO TO 3042
 3050 CALL PLYAWM(A(LA),T(L),A(LA+1),XK(LK),G1)
      JJ2 = JJ2+1
      LA = FLD(5,17,NSQ2(JJ2))                                                    *NEW
      LK = FLD(22,14,NSQ2(JJ2))                                                   *NEW
      G2 = XK(LK)*XK(LA)                                                          **-2
      GO TO 3998
 3055 TM = (T(L)+T(LTA))/2.0
      CALL D2D1WM(TM,CON(14),A(LA),XK(LK),G(LG))
      GO TO 3999
 3060 TM = T(LTA)
      GO TO 3007
 3065 TM = T(LTA)
      GO TO 3032
```

CNFRWD.CNFRWD

```
00750  175*      3998 G(LG) = 1./(1./G1+1./G2)
00751  175*           IF(FLD(3,1,NSQ1(JJ1)).EQ.1) G(LG) = G1*G2
00753  175*      3999 JJ2 = JJ2+1
00754  175*      4000 CONTINUE
00755  175*           END
00756  176*           IF(FLD(3,1,NSQ1(JJ1)).EQ.0) GO TO 140
00760  177*           T1 = T(L)+460.0
00761  178*           T2 = T(LTA)+460.0
00762  179*           GV = G(LG)*(T1*T1+T2*T2)*(T1+T2)
00763  180*           GO TO 145
00764  181*       140 GV = G(LG)
00765  182*       145 SUMC = SUMC+GV
00766  183*           SUMCV = SUMCV+GV*T(LTA)
00766  184*    C      CHECK FOR LAST CONDUCTOR
00767  185*           IF(NSQ1(JJ1).GT.0) GO TO 135
00771  186*           T2 = DD*T(L)+DN*(SUMCV+Q(L))/SUMC
00771  187*    C      OBTAIN THE CALCULATED TEMPERATURE DIFFERENCE
00772  188*           T1 = ABS(T(L)-T2)
00772  189*    C      STORE THE NEW TEMPERATURE
00773  190*           T(L) = T2
00773  191*    C      SAVE THE MAXIMUM ARITHMETIC RELAXATION CHANGE
00774  192*           IF(TCGM.GE.T1) GO TO 165
00776  193*           TCGM = T1
00777  194*           KON(37) = L
01000  195*       165 CONTINUE
01000  196*    C      SEE IF RELAXATION CRITERIA WAS MET
01002  197*           IF(TCGM.LE.CON(19)) GO TO 175
01004  198*       170 CONTINUE
01004  199*    C      STORE THE MAXIMUM ARITHMETIC RELAXATION CHANGE
01006  200*       175 CON(30) = TCGM
01006  201*    C      COMPUTE THE ARITHMETIC TEMPERATURE CHANGE
01007  202*           TCGM = 0.0
01010  203*           DO 180 I = L1,NNC
01013  204*           LE = IE+I
01014  205*           T1 = ABS(T(I)-X(LE))
01015  206*           IF(T1.LT.TCGM) GO TO 180
01017  207*           TCGM = T1
01020  208*           KON(38) = I
01021  209*       180 CONTINUE
01021  210*    C      SEE IF ATMPCA WAS SATISFIED
01023  211*           IF(TCGM.GT.CON(11)) GO TO 125
01025  212*           CON(16) = TCGM
01026  213*       185 KON(12) = 0
01027  214*           CALL VARBL2
01027  215*    C      CHECK THE BACKUP SWITCH
01030  216*           IF(KON(12).NE.0) GO TO 105
01030  217*    C      ADVANCE TIME
01032  218*           CON(13) = CON(1)
01033  219*           TSUM = TSUM+TSTEP
01034  220*           TSTEP = DELTA*0.95
01034  221*    C      CHECK FOR TIME TO PRINT
01035  222*           IF(TSUM.GE.CON(18)) GO TO 190
01035  223*    C      CHECK FOR PRINT EVERY ITERATION
01037  224*           IF(KON(7).EQ.0) GO TO 10
01041  225*           CALL OUTCAL
01042  226*           GO TO 10
01042  227*    C      TRY TO EVEN THE OUTPUT INTERVALS
01043  228*       190 TPRINT = TPRINT+TSUM
01044  229*       195 CALL OUTCAL
01044  230*    C      IS TIME GREATER THAN END COMPUTE TIME
```

CNFRWD,CNFRWD

```
01045          IF(CON(1)*1.000001.LT.CON(3)) GO TO 5
01047          NTH = IE
01050          NDIM = NLA
01051          RETURN
01052      995 WRITE(6,885)
01054          GO TO 1000
01055      996 WRITE(6,886)
01057          GO TO 1000
01060      997 WRITE(6,887)
01062          GO TO 1000
01063      998 WRITE(6,888) I
01066          GO TO 1000
01067      999 WRITE(6,889)
01071     1000 CALL OUTCAL
01072          CALL EXIT
01073      885 FORMAT(46H CNFRWD REQUIRES SHORT PSEUDO-COMPUTE SEQUENCE)
01074      886 FORMAT(24H CSGMIN ZERO OR NEGATIVE)
01075      887 FORMAT(20H TIME STEP TOO SMALL)
01076      888 FORMAT(I8,20H LOCATIONS AVAILABLE)
01077      889 FORMAT(19H NO OUTPUT INTERVAL)
01100          END

    END OF UNIVAC 1108 FORTRAN V COMPILATION.     0 *DIAGNOSTIC* MESSAGE(S)
  CNFRWD           SYMBOLIC
  CNFRWD  CODE     RELOCATABLE
```

```
                    CNFRDL,CNFRDL
QIW   FOR,*        CNFRDL,CNFRDL                                                        21 FEB 71
UNIVAC 1108 FORTRAN V ATHENA VERSION 131K-10D   CREATED ON 20 AUG 70
THIS COMPILATION WAS DONE ON 09 JUN 70 AT 14:00:31

   SUBROUTINE CNFRDL     ENTRY POINT 003463

   STORAGE USED (BLOCK, NAME, LENGTH)

         0001    *CODE          003476
         0000    *CONST+TEMP    000072
         0002    *SIMPLE VAR    000046
         0004    *ARRAYS        000000
         0005    *BLANK         000000
         0006    TITLE    000001
         0007    TEMP     000001
         0010    CAP      000001
         0011    SOURCE   000001
         0012    COND     000001
         0013    PC1      000001
         0014    PC2      000001
         0015    KONST    000001
         0016    ARRAY    000001
         0017    FIXCON   000001
         0020    XSPACE   000003
         0021    DIMENS   000010

   EXTERNAL REFERENCES (BLOCK, NAME)

         0022    VARBL1
         0023    D1D1WM
         0024    PLYAWM
         0025    D2D1WM
         0026    VARBL2
         0027    OUTCAL
         0030    EXIT
         0031    NERR2$
         0032    NWDU$
         0033    NI02$
         0034    NER10$

   STORAGE ASSIGNMENT FOR VARIABLES (BLOCK, TYPE, RELATIVE LOCATION, NAME)

    0016 R 000000 A          0010 R 000000 C          0002 R 000015 CKM        0017 R 000000 CON        0002 R 000021 C1
    0002 R 000022 C2         0002 R 000037 DD         0002 R 000035 DELTA      0002 R 000036 DN         0012 R 000000 G
    0002 R 000034 GV         0002 R 000030 G1         0002 R 000031 G2         0006 R 000000 H          0002 I 000004 I
    0002 I 000002 IE         0002 I 000041 JJ1        0002 I 000042 JJ2        0002 I 000012 J1         0002 I 000013 J2
    0015 I 000000 K          0017 I 000000 KON        0002 I 000043 L          0002 I 000017 LA         0002 I 000040 LAX
    0002 I 000011 LE         0002 I 000025 LG         0002 I 000020 LK         0021 I 000006 LSQ1       0021 I 000007 LSQ2
    0002 I 000026 LTA        0002 I 000005 L1         0021 I 000005 NAT        0021 I 000004 NCT        0020 I 000000 NDIM
    0021 I 000003 NGT        0002 I 000003 NLA        0021 I 000001 NNA        0002 I 000001 NNC        0021 I 000000 NND
    0021 I 000002 NNT        0013 I 000000 NSQ1       0014 I 000000 NSQ2       0020 I 000001 NTH        0002 I 000016 NTYPE
    0020   000002 NX         0002 R 000000 PASS       0011 R 000000 Q          0002 R 000023 Q1         0002 R 000024 Q2
    0002 R 000044 SUMC       0002 R 000045 SUMCV      0007 R 000000 T          0002 R 000014 TCGM       0002 R 000027 TM
    0002 R 000007 TPRINT     0002 R 000006 TSTEP      0002 R 000010 TSUM       0002 R 000032 T1         0002 R 000033 T2
    0020 R 000002 X          0015 R 000000 XK         0001   000126 10L        0001   003441 1000L      0001   003277 1001G
```

```
CNFRDL,CNFRDL

0001  000330 1005L   0001  000351 1010L   0001  000367 1012L   0001  000422 1015L   0001  000430 1020L
0001  000470 1025L   0001  000514 1030L   0001  000535 1032L   0001  000573 1035L   0001  000601 1040L
0001  000644 1045L   0001  002032 105L    0001  002064 115L    0001  002075 120L    0001  002103 125L
0001  002111 130L    0001  002441 135L    0001  000135 15L     0001  003207 155L    0001  003212 160L
0001  003257 165L    0001  003270 175L    0001  003321 180L    0001  003332 185L    0001  003361 190L
0001  003364 195L    0001  000666 1998L   0001  000671 1999L   0001  000207 200G    0001  000674 2000L
0001  001233 2005L   0001  001240 2007L   0001  001260 2010L   0001  001263 2015L   0001  001301 2017L
0001  001335 2020L   0001  001343 2025L   0001  001403 2030L   0001  001410 2032L   0001  001433 2035L
0001  001436 2040L   0001  001457 2042L   0001  001516 2045L   0001  001524 2050L   0001  001567 2055L
0001  001615 2060L   0001  001621 2065L   0001  000226 212G    0001  000257 232G    0001  000147 25L
0001  001625 2998L   0001  001653 2999L   0001  000162 30L     0001  001656 3000L   0001  002527 3005L
0001  002534 3007L   0001  002554 3010L   0001  002557 3015L   0001  002575 3017L   0001  002631 3020L
0001  002637 3025L   0001  002677 3030L   0001  002704 3032L   0001  002727 3035L   0001  002732 3040L
0001  002753 3042L   0001  003012 3045L   0001  003020 3050L   0001  003063 3055L   0001  003111 3060L
0001  003115 3065L   0001  003121 3998L   0001  003147 3999L   0001  003152 4000L   0001  000741 4005L
0001  000746 4010L   0001  000747 4012L   0001  000766 4015L   0001  000767 4017L   0001  001004 4020L
0001  001020 4022L   0001  001035 4025L   0001  001043 4030L   0001  001101 4035L   0001  001115 4037L
0001  001132 4040L   0001  000240 45L     0001  001745 474G    0001  001140 4998L   0001  001144 4999L
0001  000114 5L      0001  001147 5000L   0001  002233 5005L   0001  002240 5010L   0001  002241 5012L
0001  002260 5015L   0001  002261 5017L   0001  002276 5020L   0001  002312 5022L   0001  002327 5025L
0001  002335 5030L   0001  002373 5035L   0001  002407 5037L   0001  002424 5040L   0001  002036 525G
0001  002140 563G    0001  002161 572G    0001  002432 5998L   0001  002436 5999L   0001  002441 6000L
0001  001147 70L     0001  001713 75L     0001  001716 80L     0000  000000 885F    0000  000011 886F
0000  000016 887F    0000  000023 888F    0000  000030 889F    0001  001766 90L     0001  002005 95L
0001  003403 995L    0001  003411 996L    0001  003417 997L    0001  003425 998L    0001  003434 999L
```

```
00101    1*          SUBROUTINE CNF:DL
00101    2*    C     EXPLICIT FORWARD DIFFERENCING EXECUTION SUBROUTINE FOR SINDA F-V
00101    3*    C     THE LONG PSEUDO-COMPUTE SEQUENCE IS REQUIRED
00103    4*          INCLUDE COMM,LIST
00104    4*          COMMON /TITLE/H(1) /TEMP/T(1) /CAP/C(1) /SOURCE/Q(1) /COND/G(1)
00105    4*          COMMON /PC1/NSQ1(1) /PC2/NSQ2(1) /KONST/K(1) /ARRAY/A(1)
00106    4*          COMMON /FIXCON/KON(1) /XSPACE/NDIM,NTH,X(1)
00107    4*          COMMON /DIMENS/ NND,NNA,NNT,NGT,NCT,NAT,LSQ1,LSQ2
00110    4*          DIMENSION CON(1),XK(1),NX(1)
00111    4*          EQUIVALENCE (KON(1),CON(1)),(K(1),XK(1)),(X(1),NX(1))
00112    4*          END
00113    5*          INCLUDE DEFF,LIST
00113    5*    C*************  CONTROL CONSTANT DEFINITIONS AND NAMES  *****************
00113    5*    C     CONTROL CONSTANT 1 CONTAINS THE NEW PROBLEM TIME              (TIMEN)
00113    5*    C     CONTROL CONSTANT 2 CONTAINS THE TIME STEP USED               (DTIMEU)
00113    5*    C     CONTROL CONSTANT 3 CONTAINS THE PROBLEM STOP TIME            (TIMEND)
00113    5*    C     CONTROL CONSTANT 4 CONTAINS THE TIME STEP FACTOR,EXPLICIT (CSGFAC)
00113    5*    C     CC5 IS THE INPUT NUMBER OF ITERATION DO LOOPS, INTEGER        (NLOOP)
00113    5*    C     CC6 CONTAINS THE DIFFUSION TEMPERATURE CHANGE ALLOWED       (DTMPCA)
00113    5*    C     CC7 CONTAINS THE OUTPUT EACH ITERATION SWITCH               (OPEITR)
00113    5*    C     CC8 CONTAINS THE MAXIMUM ALLOWED TIME STEP                  (DTIMEH)
00113    5*    C     CC9 CONTAINS THE NEW ARITHMETIC TEMP. DAMPING FACTOR         (DAMPA)
00113    5*    C     CC10 CONTAINS THE NEW DIFFUSION TEMP. DAMPING FACTOR         (DAMPD)
00113    5*    C     CC11 CONTAINS THE MAXIMUM ALLOWED ARITHMETIC TEMP. CHANGE (ATMPCA)
00113    5*    C     CC12 CONTAINS THE BACKUP SWITCH CHECKED AFTER VARIABLES     (BACKUP)
00113    5*    C     CC13 CONTAINS THE PRESENT TIME OR PROBLEM START TIME         (TIMEO)
00113    5*    C     CC14 CONTAINS THE MEAN TIME BETWEEN AN ITERATION             (TIMEM)
00113    5*    C     CC15 CONTAINS THE DIFFUSION TEMPERATURE CHANGE CALCULATED (DTMPCC)
00113    5*    C     CC16 CONTAINS ARITHMETIC TEMPERATURE CHANGE CALCULATED      (ATMPCC)
00113    5*    C     CONTROL CONSTANT 17 IS RESERVED FOR THE C/SG MINIMUM        (CSGMIN)
00113    5*    C     CONTROL CONSTANT 18 CONTAINS THE OUTPUT INTERVAL            (OUTPUT)
```

```
00113     5*   C      CC19 CONTAINS THE ARITHMETIC RELAXATION CRITERIA ALLOWED  (ARLXCA)
00113     5*   C      CC20 CONTAINS THE NUMBER OF RELAXATION LOOPS USED.INTEGER (LOOPCT)
00113     5*   C      CC21 CONTAINS THE MINIMUM ALLOWED TIME STEP               (DTIMEL)
00113     5*   C      CC22 IS FOR THE INPUT TIME STEP IMPLICIT                  (DTIMEI)
00113     5*   C      CC23 CONTAINS THE C/SG MAXIMUM                            (CSGMAX)
00113     5*   C      CC24 CONTAINS THE C/SG RANGE ALLOWED                      (CSGRAL)
00113     5*   C      CC25 CONTAINS THE C/SG RANGE CALCULATED                   (CSGRCL)
00113     5*   C      CC26 CONTAINS THE DIFFUSION RELAXATION CRITERIA ALLOWED   (DRLXCA)
00113     5*   C      CC27 CONTAINS THE DIFFUSION RELAXATION CHANGE CALCULATED  (DRLXCC)
00113     5*   C      CC28 CONTAINS THE LINE COUNTER, INTEGER                   (LINECT)
00113     5*   C      CC29 CONTAINS THE PAGE COUNTER, INTEGER                   (PAGECT)
00113     5*   C      CC30 CONTAINS ARITHMETIC RELAXATION CHANGE CALCULATED     (ARLXCC)
00113     5*   C      CC31 IS INDICATOR, 0=THERMAL SPCS,1=THERMAL LPCS,2=GENERAL (LSPCS)
00113     5*   C      CC32 CONTAINS THE ENERGY BALANCE OF THE SYSTEM, IN - OUT  (ENGBAL)
00113     5*   C      CC33 CONTAINS THE DESIRED ENERGY BALANCE, USER INPUT      (BALENG)
00113     5*   C      CC34 CONTAINS THE NOCOPY SWITCH FOR MATRIX USERS          (NOCOPY)
00113     5*   C      CC35 CONTAINS RELATIVE NODE NUMBER OF CSGMIN
00113     5*   C      CC36 CONTAINS RELATIVE NODE NUMBER OF DTMPCC
00113     5*   C      CC37 CONTAINS RELATIVE NODE NUMBER OF ARLXCC
00113     5*   C      CC38 CONTAINS RELATIVE NODE NUMBER OF ATMPCC
00113     5*   C      CC39-40-41-42-43 CONTAIN DUMMY INTEGER CONSTANTS   (I-J-K-L-MTEST)
00113     5*   C      CC44-45-46-47-48 CONTAIN DUMMY FLOATING CONSTANTS  (R-S-T-U-VTEST)
00113     5*   C      CC49 IS THE QUASI-LINEARIZATION INTERVAL FOR CINDSM       (LAXFAC)
00113     5*   C      CC50 IS NOT USED AT PRESENT
00114     5*          END
00115     6*          IF(CON(4).LT.1.0) CON(4) = 1.0
00117     7*          IF(KON(5).LE.0) KON(5) = 1
00121     8*          IF(CON(6).LE.0.) CON(6) = 1.E+8
00123     9*          IF(CON(8).LE.0.) CON(8) = 1.E+8
00125    10*          IF(CON(9).LE.0.) CON(9) = 1.0
00127    11*          IF(CON(11).LE.0.) CON(11) = 1.E+8
00131    12*          IF(CON(18).LE.0.) GO TO 999
00133    13*          IF(CON(19).LE.0.) CON(19) = 1.E+8
00135    14*          IF(KON(31).NE.1) GO TO 995
00137    15*          PASS = -1.0
00140    16*          NNC = NND+NNA
00141    17*          IE = NTH
00142    18*          NLA = NDIM
00143    19*          NTH = NTH+NNC
00144    20*          NDIM = NDIM-NNC
00144    21*   C      CHECK FOR EXTRA LOCATIONS FOR CALCULATED NODES
00145    22*          I = NLA-NNC
00146    23*          IF(I.LT.0) GO TO 998
00150    24*          L1 = NND+1
00151    25*          TSTEP = CON(18)
00152    26*          TPRINT = CON(13)
00152    27*   C      INITALIZE TIME SUM BETWEEN OUTPUT INTERVALS
00153    28*        5 TSUM = 0.0
00153    29*   C      DOES OLD TIME PLUS THE OUTPUT INTERVAL EXCEED THE STOP TIME
00154    30*          IF(CON(13)+CON(18).LE.CON(3)) GO TO 10
00154    31*   C      DONT EXCEED IT
00156    32*          CON(18) = CON(3)-CON(13)
00156    33*   C      IS THE TIME STEP LARGER THAN ALLOWED
00157    34*       10 IF(TSTEP.LE.CON(8)) GO TO 15
00161    35*          TSTEP = CON(8)
00161    36*   C      DOES THE TIME SUM PLUS THE TIME STEP EXCEED OUTPUT INTERVAL
00162    37*       15 IF(TSUM+TSTEP-CON(18)) 25,30,20
00162    38*   C      DONT EXCEED IT
00165    39*       20 TSTEP = CON(18)-TSUM
00166    40*          GO TO 30
```

```
C       DOES TIME SUM PLUS TWO TIME STEPS EXCEED OUTPUT INTERVAL
     25 IF(TSUM+2.0*TSTEP.LE.CON(18)) GO TO 30
C       APPROACH THE OUTPUT INTERVAL GRADUALLY
        TSTEP = (CON(18)-TSUM)/2.0
C       STORE DELTA TIME STEP IN THE CONSTANTS
     30 CON(2) = TSTEP
C       IS THE TIME STEP USED LESS THAN THE TIME STEP ALLOWED
        IF(TSTEP.LT.CON(21)) GO TO 997
C       CALCULATE THE NEW TIME
        CON(1) = TPRINT+TSUM+TSTEP
C       COMPUTE THE MEAN TIME BETWEEN ITERATIONS
        CON(14) = (CON(1)+CON(13))/2.0
C       ZERO OUT ALL SOURCE LOCATIONS AND EXTRA LOCATIONS
        DO 35 I = 1,NND
        LE = IE+I
        X(LE) = 0.0
        Q(I) = 0.0
     35 CONTINUE
C       SHIFT THE ARITHMETIC TEMPERATURES INTO THE EXTRA LOCATIONS
        IF(NNA.LE.0) GO TO 45
        DO 40 I = L1,NNC
        Q(I) = 0.0
        LE = IE+I
        X(LE) = T(I)
     40 CONTINUE
     45 KON(12) = 0
        CALL VARBL1
        IF(KON(12).NE.0) GO TO 10
        J1 = 0
        J2 = 1
        TCGM = 0.0
        CKM = 1.E+8
C       CALCULATE Q SUM AND G SUM
        DO 85 I = 1,NND
        LE = IE+I
        INCLUDE VARC,LIST
        IF(FLD(1,1,NSQ1(J1+1)).EQ.0) GO TO 2000
        NTYPE = FLD(0,5,NSQ2(J2))                                        *NEW
        LA = FLD(5,17,NSQ2(J2))                                          *NEW
        LK = FLD(22,14,NSQ2(J2))                                         *NEW
        GO TO (1005,1010,1015,1020,1025,1030,1035,1040,1045), NTYPE      **-3
   1005 CALL D1D1WM(T(I),A(LA),XK(LK),C(I))
        GO TO 1999
   1010 CALL D1D1WM(T(I),A(LA),XK(LK),C1)
   1012 J2 = J2+1
        LA = FLD(5,17,NSQ2(J2))                                          *NEW
        LK = FLD(22,14,NSQ2(J2))                                         *NEW
        CALL D1D1WM(T(I),A(LA),XK(LK),C2)                                **-2
        GO TO 1998
   1015 C1 = XK(LK)*XK(LA)
        GO TO 1012
   1020 CALL D1D1WM(T(I),A(LA),XK(LK),C1)
        J2 = J2+1
        LA = FLD(5,17,NSQ2(J2))                                          *NEW
        LK = FLD(22,14,NSQ2(J2))                                         *NEW
        C2 = XK(LK)*XK(LA)                                               **-2
        GO TO 1998
   1025 CALL PLYAWM(A(LA),T(I),A(LA+1),XK(LK),C(I))
        GO TO 1999
   1030 CALL PLYAWM(A(LA),T(I),A(LA+1),XK(LK),C1)
```

CNFRDL,CNFRDL

```
76*     1032 J2 = J2+1
76*          LA = FLD(5,17,NSQ2(J2))                                      *NEW
76*          LK = FLD(22,14,NSQ2(J2))                                     *NEW
76*          CALL PLYAWM(A(LA),T(I),A(LA+1),XK(LK),C2)                    **-2
76*          GO TO 1998
76*     1035 C1 = XK(LK)*XK(LA)
76*          GO TO 1032
76*     1040 CALL PLYAWM(A(LA),T(I),A(LA+1),XK(LK),C1)
76*          J2 = J2+1
76*          LA = FLD(5,17,NSQ2(J2))                                      *NEW
76*          LK = FLD(22,14,NSQ2(J2))                                     *NEW
76*          C2 = XK(LK)*XK(LA)                                           **-2
76*          GO TO 1998
76*     1045 CALL D2D1WM(T(I),CON(14),A(LA),XK(LK),C(I))
76*          GO TO 1999
76*     1998 C(I) = C1+C2
76*     1999 J2 = J2+1
76*     2000 CONTINUE
76*          END
77*          INCLUDE VARQ,LIST
77*          IF(FLD(4,1,NSQ1(J1+1)).EQ.0) GO TO 5000
77*          NTYPE = FLD(0,5,NSQ2(J2))                                    *NEW
77*          LA = FLD(5,17,NSQ2(J2))                                      *NEW
77*          LK = FLD(22,14,NSQ2(J2))                                     *NEW
77*          GO TO (4005,4010,4015,4020,4025,4030,4035,4040,4030), NTYPE  **-3
77*     4005 Q(I) = XK(LK)+Q(I)
77*          GO TO 4999
77*     4010 Q1 = 0.0
77*     4012 CALL D1D1WM(T(I),A(LA),XK(LK),Q2)
77*          GO TO 4998
77*     4015 Q1 = 0.0
77*     4017 CALL D1D1WM(CON(14),A(LA),XK(LK),Q2)
77*          GO TO 4998
77*     4020 CALL D1D1WM(CON(14),A(LA),XK(LK),Q1)
77*     4022 J2 = J2+1
77*          LA = FLD(5,17,NSQ2(J2))                                      *NEW
77*          LK = FLD(22,14,NSQ2(J2))                                     *NEW
77*          GO TO 4017                                                   **-2
77*     4025 Q1 = XK(LK)*XK(LA)
77*          GO TO 4022
77*     4030 CALL D1D1WM(CON(14),A(LA),XK(LK),Q1)
77*          J2 = J2+1
77*          LA = FLD(5,17,NSQ2(J2))                                      *NEW
77*          LK = FLD(22,14,NSQ2(J2))                                     *NEW
77*          Q2 = XK(LK)*XK(LA)                                           **-2
77*          GO TO 4998
77*     4035 CALL D1D1WM(CON(14),A(LA),XK(LK),Q1)
77*     4037 J2 = J2+1
77*          LA = FLD(5,17,NSQ2(J2))                                      *NEW
77*          LK = FLD(22,14,NSQ2(J2))                                     *NEW
77*          GO TO 4012                                                   **-2
77*     4040 Q1 = XK(LK)*XK(LA)
77*          GO TO 4037
77*     4998 Q(I) = Q1+Q2+Q(I)
77*     4999 J2 = J2+1
77*     5000 CONTINUE
77*          END
78*       70 J1 = J1+1
79*          LG = FLD(5,16,NSQ1(J1))
80*          LTA = FLD(22,14,NSQ1(J1))
```

CNFRDL,CNFRDL

```
00364   81*          INCLUDE VARG,LIST
00364   82*     C    CHECK FOR RADIATION CONDUCTOR
00365   82*          IF(FLD(2,1,NSQ1(J1)).EQ.0) GO TO 3000
00367   82*          NTYPE = FLD(0,5,NSQ2(J2))                                          *NEW
00370   82*          LA = FLD(5,17,NSQ2(J2))                                            *NEW
00371   82*          LK = FLD(22,14,NSQ2(J2))                                           *NEW
00372   82*          GOTO(2005,2010,2015,2020,2025,2030,2035,2040,2045,2050,2055,       **-3
00372   82*        $      2060,2065), NTYPE
00373   82*     2005 TM = (T(I)+T(LTA))/2.0
00374   82*     2007 CALL D1D1WM(TM,A(LA),XK(LK),G(LG))
00375   82*          GO TO 2999
00376   82*     2010 TM = T(I)
00377   82*          GO TO 2007
00400   82*     2015 CALL D1D1WM(T(I),A(LA),XK(LK),G1)
00401   82*     2017 J2 = J2+1
00402   82*          LA = FLD(5,17,NSQ2(J2))                                            *NEW
00403   82*          LK = FLD(22,14,NSQ2(J2))                                           *NEW
00404   82*          CALL D1D1WM(T(LTA),A(LA),XK(LK),G2)                                **-2
00405   82*          GO TO 2998
00406   82*     2020 G1 = XK(LK)*XK(LA)
00407   82*          GO TO 2017
00410   82*     2025 CALL D1D1WM(T(I),A(LA),XK(LK),G1)
00411   82*          J2 = J2+1
00412   82*          LA = FLD(5,17,NSQ2(J2))                                            *NEW
00413   82*          LK = FLD(22,14,NSQ2(J2))                                           *NEW
00414   82*          G2 = XK(LK)*XK(LA)                                                 **-2
00415   82*          GO TO 2998
00416   82*     2030 TM = (T(I)+T(LTA))/2.0
00417   82*     2032 CALL PLYAWM(A(LA),TM,A(LA+1),XK(LK),G(LG))
00420   82*          GO TO 2999
00421   82*     2035 TM = T(I)
00422   82*          GO TO 2032
00423   82*     2040 CALL PLYAWM(A(LA),T(I),A(LA+1),XK(LK),G1)
00424   82*     2042 J2 = J2+1
00425   82*          LA = FLD(5,17,NSQ2(J2))                                            *NEW
00426   82*          LK = FLD(22,14,NSQ2(J2))                                           *NEW
00427   82*          CALL PLYAWM(A(LA),T(LTA),A(LA+1),XK(LK),G2)                        **-2
00430   82*          GO TO 2998
00431   82*     2045 G1 = XK(LK)*XK(LA)
00432   82*          GO TO 2042
00433   82*     2050 CALL PLYAWM(A(LA),T(I),A(LA+1),XK(LK),G1)
00434   82*          J2 = J2+1
00435   82*          LA = FLD(5,17,NSQ2(J2))                                            *NEW
00436   82*          LK = FLD(22,14,NSQ2(J2))                                           *NEW
00437   82*          G2 = XK(LK)*XK(LA)                                                 **-2
00440   82*          GO TO 2998
00441   82*     2055 TM = (T(I)+T(LTA))/2.0
00442   82*          CALL D2D1WM(TM,CON(14),A(LA),XK(LK),G(LG))
00443   82*          GO TO 2999
00444   82*     2060 TM = T(LTA)
00445   82*          GO TO 2007
00446   82*     2065 TM = T(LTA)
00447   82*          GO TO 2032
00450   82*     2998 G(LG) = 1./(1./G1+1./G2)
00451   82*          IF(FLD(3,1,NSQ1(J1)).EQ.1) G(LG) = G1*G2
00453   82*     2999 J2 = J2+1
00454   82*     3000 CONTINUE
00455   82*          END
00456   83*          IF(FLD(3,1,NSQ1(J1)).EQ.0) GO TO 75
00460   84*          T1 = T(I)+460.0
```

CNFRDL.CNFPDL

```
00461          T2 = T(LTA)+460.0
00462          GV = G(LG)*(T1*T1+T2*T2)*(T1+T2)
00463          GO TO 80
00464       75 GV = G(LG)
00464    C     OBTAIN THE Q RATE THRU THE CONDUCTOR
00465       80 Q(I) = Q(I)+GV*(T(LTA)-T(I))
00465    C     SAVE SUMMATION OF CONDUCTORS
00466          X(LE) = X(LE)+GV
00466    C     CHECK FOR LAST CONDUCTOR
00467          IF(NSQ1(J1).GT.0) GO TO 70
00471       85 CONTINUE
00471    C     OBTAIN NEW DIFFUSION TEMPERATURES, DTMPCC AND CSGMIN
00473          DO 100 I = 1,NND
00476          LE = IE+I
00476    C     CALCULATE C/SK MINIMUM
00477          T1 = C(I)/X(LE)
00500          IF(T1.GE.CKM) GO TO 90
00502          CKM = T1
00503          KON(35) = I
00503    C     COMPUTE NEW TEMPERATURES USING CALCULATED SOURCE TERMS
00504       90 T1 = TSTEP*Q(I)/C(I)
00504    C     CALCULATE THE ABSOLUTE VALUE TEMPERATURE CHANGE
00505          T2 = ABS(T1)
00505    C     SAVE THE LARGEST TEMPERATURE CHANGE
00506          IF(TCGM.GE.T2) GO TO 95
00510          TCGM = T2
00511          KON(36) = I
00511    C     STORE THE TEMPERATURES
00512       95 X(LE) = T(I)
00513          T(I) = T(I)+T1
00514      100 CONTINUE
00516          CON(17) = CKM
00517          DELTA = CKM/CON(4)
00520          IF(CKM.LE.0.0) GO TO 996
00520    C     CHECK FOR FIRST PASS
00522          IF(PASS.GT.0.0) GO TO 115
00522    C     UNDO THE TEMPERATURE CALCULATIONS
00524      105 DO 110 I = 1,NNC
00527          LE = IE+I
00530          T(I) = X(LE)
00531      110 CONTINUE
00533          IF(PASS.GT.0.0) GO TO 15
00535          PASS = 1.0
00536          CON(1) = TPRINT
00537          CON(2) = 0.0
00540          TSTEP = DELTA*0.95
00541          GO TO 195
00541    C     IS THE TIME STEP USED LESS THAN THE TIME STEP CALCULATED
00542      115 IF(TSTEP.LE.DELTA) GO TO 130
00542    C     COMPUTE THE TIME STEP
00544          TSTEP = DELTA*0.95
00545          GO TO 105
00546      120 TSTEP = 0.95*TSTEP*CON(6)/TCGM
00547          GO TO 105
00550      125 TSTEP = 0.95*TSTEP*CON(11)/TCGM
00551          GO TO 105
00551    C     SEE IF THE TEMPERATURE CHANGE WAS TOO LARGE
00552      130 IF(TCGM.GT.CON(6)) GO TO 120
00552    C     STORE THE MAXIMUM DIFFUSION TEMPERATURE CHANGE
00554          CON(15) = TCGM
```

```
145*   C      CHECK TO SEE IF THERE ARE ANY ARITHMETIC NODES
146*          IF(NNA.LE.0) GO TO 185
147*   C      COMPUTE ARITHMETIC TEMPERATURES BY SUCCESSIVE POINT OVER-RELAX
148*          DN = CON(9)
149*          DD = 1.0-DN
150*          LAX = KON(5)
151*          DO 170 I = 1,LAX
152*          JJ1 = J1
153*          JJ2 = J2
154*          TCGM = 0.0
155*          KON(20) = I
156*          DO 165 L = L1,NNC
157*          SUMC = 0.0
158*          SUMCV = 0.0
159*          IF(I.GT.1) GO TO 6000
160*          INCLUDE VRQ2,LIST
160*          IF(FLD(4,1,NSQ1(JJ1+1)).EQ.0) GO TO 6000
160*          NTYPE = FLD(0,5,NSQ2(JJ2))                                  *NEW
160*          LA = FLD(5,17,NSQ2(JJ2))                                    *NEW
160*          LK = FLD(22,14,NSQ2(JJ2))                                   *NEW
160*          GO TO (5005,5010,5015,5020,5025,5030,5035,5040,5030), NTYPE **-3
160*     5005 Q(L) = XK(LK)+Q(L)
160*          GO TO 5999
160*     5010 Q1 = 0.0
160*     5012 CALL D1D1WM(T(L),A(LA),XK(LK),Q2)
160*          GO TO 5998
160*     5015 Q1 = 0.0
160*     5017 CALL D1D1WM(CON(14),A(LA),XK(LK),Q2)
160*          GO TO 5998
160*     5020 CALL D1D1WM(CON(14),A(LA),XK(LK),Q1)
160*     5022 JJ2 = JJ2+1
160*          LA = FLD(5,17,NSQ2(JJ2))                                    *NEW
160*          LK = FLD(22,14,NSQ2(JJ2))                                   *NEW
160*          GO TO 5017                                                  **-2
160*     5025 Q1 = XK(LK)*XK(LA)
160*          GO TO 5022
160*     5030 CALL D1D1WM(CON(14),A(LA),XK(LK),Q1)
160*          JJ2 = JJ2+1
160*          LA = FLD(5,17,NSQ2(JJ2))                                    *NEW
160*          LK = FLD(22,14,NSQ2(JJ2))                                   *NEW
160*          Q2 = XK(LK)*XK(LA)                                          **-2
160*          GO TO 5998
160*     5035 CALL D1D1WM(CON(14),A(LA),XK(LK),Q1)
160*     5037 JJ2 = JJ2+1
160*          LA = FLD(5,17,NSQ2(JJ2))                                    *NEW
160*          LK = FLD(22,14,NSQ2(JJ2))                                   *NEW
160*          GO TO 5012                                                  **-2
160*     5040 Q1 = XK(LK)*XK(LA)
160*          GO TO 5037
160*     5998 Q(L) = Q1+Q2+Q(L)
160*     5999 JJ2 = JJ2+1
160*     6000 CONTINUE
160*          END
161*      135 JJ1 = JJ1+1
162*          LG = FLD(5,16,NSQ1(JJ1))
163*          LTA = FLD(22,14,NSQ1(JJ1))
164*          IF(I.GT.1) GO TO 4000
165*          INCLUDE VRG2,LIST
166*   C      CHECK FOR RADIATION CONDUCTOR
166*          IF(FLD(2,1,NSQ1(JJ1)).EQ.0) GO TO 4000
```

CNFRDL,CNFRDL

```
00657  166*        NTYPE = FLD(0,5,NSQ2(JJ2))                                        *NEW
00660  166*        LA = FLD(5,17,NSQ2(JJ2))                                          *NEW
00661  166*        LK = FLD(22,14,NSQ2(JJ2))                                         *NEW
00662  166*        GOTO(3005,3010,3015,3020,3025,3030,3035,3040,3045,3050,3055,      **-3
00662  166*       $     3060,3065), NTYPE
00663  166*   3005 TM = (T(L)+T(LTA))/2.0
00664  166*   3007 CALL D1D1WM(TM,A(LA),XK(LK),G(LG))
00665  166*        GO TO 3999
00666  166*   3010 TM = T(L)
00667  166*        GO TO 3007
00670  166*   3015 CALL D1D1WM(T(L),A(LA),XK(LK),G1)
00671  166*   3017 JJ2 = JJ2+1
00672  166*        LA = FLD(5,17,NSQ2(JJ2))                                          *NEW
00673  166*        LK = FLD(22,14,NSQ2(JJ2))                                         *NEW
00674  166*        CALL D1D1WM(T(LTA),A(LA),XK(LK),G2)                               **-2
00675  166*        GO TO 3998
00676  166*   3020 G1 = XK(LK)*XK(LA)
00677  166*        GO TO 3017
00700  166*   3025 CALL D1D1WM(T(L),A(LA),XK(LK),G1)
00701  166*        JJ2 = JJ2+1
00702  166*        LA = FLD(5,17,NSQ2(JJ2))                                          *NEW
00703  166*        LK = FLD(22,14,NSQ2(JJ2))                                         *NEW
00704  166*        G2 = XK(LK)*XK(LA)                                                **-2
00705  166*        GO TO 3998
00706  166*   3030 TM = (T(L)+T(LTA))/2.0
00707  166*   3032 CALL PLYAWM(A(LA),TM,A(LA+1),XK(LK),G(LG))
00710  166*        GO TO 3999
00711  166*   3035 TM = T(L)
00712  166*        GO TO 3032
00713  166*   3040 CALL PLYAWM(A(LA),T(L),A(LA+1),XK(LK),G1)
00714  166*   3042 JJ2 = JJ2+1
00715  166*        LA = FLD(5,17,NSQ2(JJ2))                                          *NEW
00716  166*        LK = FLD(22,14,NSQ2(JJ2))                                         *NEW
00717  166*        CALL PLYAWM(A(LA),T(LTA),A(LA+1),XK(LK),G2)                       **-2
00720  166*        GO TO 3998
00721  166*   3045 G1 = XK(LK)*XK(LA)
00722  166*        GO TO 3042
00723  166*   3050 CALL PLYAWM(A(LA),T(L),A(LA+1),XK(LK),G1)
00724  166*        JJ2 = JJ2+1
00725  166*        LA = FLD(5,17,NSQ2(JJ2))                                          *NEW
00726  166*        LK = FLD(22,14,NSQ2(JJ2))                                         *NEW
00727  166*        G2 = XK(LK)*XK(LA)                                                **-2
00730  166*        GO TO 3998
00731  166*   3055 TM = (T(L)+T(LTA))/2.0
00732  166*        CALL D2D1WM(TM,CON(14),A(LA),XK(LK),G(LG))
00733  166*        GO TO 3999
00734  166*   3060 TM = T(LTA)
00735  166*        GO TO 3007
00736  166*   3065 TM = T(LTA)
00737  166*        GO TO 3032
00740  166*   3998 G(LG) = 1./(1./G1+1./G2)
00741  166*        IF(FLD(3,1,NSQ1(JJ1)).EQ.1) G(LG) = G1*G2
00743  166*   3999 JJ2 = JJ2+1
00744  166*   4000 CONTINUE
00745  166*        END
00746  167*        IF(FLD(3,1,NSQ1(JJ1)).EQ.0) GO TO 155
00750  168*        T1 = T(L)+460.0
00751  169*        T2 = T(LTA)+460.0
00752  170*        GV = G(LG)*(T1*T1+T2*T2)*(T1+T2)
00753  171*        GO TO 160
```

```
00754        155 GV = G(LG)
00755        160 SUMC = SUMC+GV
00756            SUMCV = SUMCV+GV*T(LTA)
00756      C     CHECK FOR LAST CONDUCTOR
00757            IF(NSQ1(JJ1).GT.0) GO TO 135
00761            T2 = DD*T(L)+DN*(SUMCV+Q(L))/SUMC
00761      C     OBTAIN THE CALCULATED TEMPERATURE DIFFERENCE
00762            T1 = ABS(T(L)-T2)
00762      C     STORE THE NEW TEMPERATURE
00763            T(L) = T2
00763      C     SAVE THE MAXIMUM ARITHMETIC RELAXATION CHANGE
00764            IF(TCGM.GE.T1) GO TO 165
00766            TCGM = T1
00767            KON(37) = L
00770        165 CONTINUE
00770      C     SEE IF RELAXATION CRITERIA WAS MET
00772            IF(TCGM.LE.CON(19)) GO TO 175
00774        170 CONTINUE
00774      C     STORE THE MAXIMUM ARITHMETIC RELAXATION CHANGE
00776        175 CON(30) = TCGM
00776      C     COMPUTE THE ARITHMETIC TEMPERATURE CHANGE
00777            TCGM = 0.0
01000            DO 180 I = L1,NNC
01003            LE = IE+I
01004            T1 = ABS(T(I)-X(LE))
01005            IF(T1.LT.TCGM) GO TO 180
01007            TCGM = T1
01010            KON(38) = I
01011        180 CONTINUE
01011      C     SEE IF ATMPCA WAS SATISFIED
01013            IF(TCGM.GT.CON(11)) GO TO 125
01015            CON(16) = TCGM
01016        185 KON(12) = 0
01017            CALL VARBL2
01017      C     CHECK THE BACKUP SWITCH
01020            IF(KON(12).NE.0) GO TO 105
01020      C     ADVANCE TIME
01022            CON(13) = CON(1)
01023            TSUM = TSUM+TSTEP
01024            TSTEP = DELTA*0.95
01024      C     CHECK FOR TIME TO PRINT
01025            IF(TSUM.GE.CON(18)) GO TO 190
01025      C     CHECK FOR PRINT EVERY ITERATION
01027            IF(KON(7).EQ.0) GO TO 10
01031            CALL OUTCAL
01032            GO TO 10
01032      C     TRY TO EVEN THE OUTPUT INTERVALS
01033        190 TPRINT = TPRINT+TSUM
01034        195 CALL OUTCAL
01034      C     IS TIME GREATER THAN END COMPUTE TIME
01035            IF(CON(1)*1.000001.LT.CON(3)) GO TO 5
01037            NTH = IE
01040            NDIM = NLA
01041            RETURN
01042        995 WRITE(6,885)
01044            GO TO 1000
01045        996 WRITE(6,886)
01047            GO TO 1000
01050        997 WRITE(6,887)
01052            GO TO 1000
```

CNFRDL,CNFRDL

```
01053  232*      998 WRITE(6,888) I
01056  233*          GO TO 1000
01057  234*      999 WRITE(6,889)
01061  235*     1000 CALL OUTCAL
01062  236*          CALL EXIT
01063  237*      885 FORMAT(45H CNFRDL REQUIRES LONG PSEUDO-COMPUTE SEQUENCE)
01064  238*      886 FORMAT(24H CSGMIN ZERO OR NEGATIVE)
01065  239*      887 FORMAT(20H TIME STEP TOO SMALL)
01066  240*      888 FORMAT(I8,20H LOCATIONS AVAILABLE)
01067  241*      889 FORMAT(19H NO OUTPUT INTERVAL)
01070  242*          END

      END OF UNIVAC 1108 FORTRAN V COMPILATION.     0 *DIAGNOSTIC* MESSAGE(S)
   CNFRDL           SYMBOLIC
   CNFRDL   CODE    RELOCATABLE
```

```
                    CNFAST,CNFAST

QIW   FOR,*        CNFAST,CNFAST                                                        21 FEB 71
UNIVAC 1108 FORTRAN V ATHENA VERSION 131K-10D   CREATED ON 20 AUG 70
THIS COMPILATION WAS DONE ON 09 JUN 70 AT 14:00:25

  SUBROUTINE CNFAST    ENTRY POINT 003360

  STORAGE USED (BLOCK, NAME, LENGTH)

        0001    *CODE             003373
        0000    *CONST+TEMP       000070
        0002    *SIMPLE VAR       000047
        0004    *ARRAYS           000000
        0005    *BLANK            000000
        0006    TITLE    000001
        0007    TEMP     000001
        0010    CAP      000001
        0011    SOURCE   000001
        0012    COND     000001
        0013    PC1      000001
        0014    PC2      000001
        0015    KONST    000001
        0016    ARRAY    000001
        0017    FIXCON   000001
        0020    XSPACE   000003
        0021    DIMENS   000010

  EXTERNAL REFERENCES (BLOCK, NAME)

        0022    VARBL1
        0023    OUTCAL
        0024    D1D1WM
        0025    PLYAWM
        0026    D2D1WM
        0027    VARBL2
        0030    EXIT
        0031    NERR2$
        0032    NWDU$
        0033    NIO2$
        0034    NER10$

  STORAGE ASSIGNMENT FOR VARIABLES (BLOCK, TYPE, RELATIVE LOCATION, NAME)

   0016 R 000000 A          0010 R 000000 C          0002 R 000034 CKM        0017 R 000000 CON        0002 R 000017 C1
   0002 R 000020 C2         0002 R 000037 DAMPN      0002 R 000040 DAMPO      0012 R 000000 G          0002 R 000032 GV
   0002 R 000026 G1         0002 R 000027 G2         0006 R 000000 H          0002 I 000010 I          0002 I 000002 IE
   0002 I 000042 JJ1        0002 I 000043 JJ2        0002 I 000012 J1         0002 I 000013 J2         0015 I 000000 K
   0017 I 000000 KON        0002 I 000044 L          0002 I 000015 LA         0002 I 000036 LAX        0002 I 000011 LE
   0002 I 000033 LEA        0002 I 000023 LG         0002 I 000016 LK         0021 I 000006 LSQ1       0021 I 000007 LSQ2
   0002 I 000024 LTA        0021 I 000005 NAT        0021 I 000004 NCT        0020 I 000000 NDIM       0021 I 000003 NGT
   0002 I 000003 NLA        0002 I 000004 NN         0021 I 000001 NNA        0002 I 000001 NNC        0021 I 000000 NMD
   0021 I 000002 NNT        0013 I 000000 NSQ1       0014 I 000000 NSQ2       0020 I 000001 NTH        0002 I 000014 NTYPE
   0020   000002 NX         0002 R 000000 PASS       0011 R 000000 Q          0002 R 000021 Q1         0002 R 000022 Q2
   0002 R 000041 RLX        0002 R 000045 SUMC       0002 R 000046 SUMCV      0007 R 000000 T          0002 R 000035 TCGM
   0002 R 000025 TM         0002 R 000005 TPRINT     0002 R 000006 TSTEP      0002 R 000007 TSUM       0002 R 000030 T1
   0002 R 000031 T2         0020 R 000002 X          0015 R 000000 XK         0001   000104 10L        0001   002036 100L
```

CNFAST,CNFAST

```
0001  003336 1000L    0001  000322 1005L    0001  000343 1010L    0001  000361 1012L    0001  000414 1015L
0001  000422 1020L    0001  000462 1025L    0001  000506 1030L    0001  000527 1032L    0001  000565 1035L
0001  000573 1040L    0001  000636 1045L    0001  002055 105L     0001  002405 110L     0001  003153 115L
0001  003161 120L     0001  003223 140L     0001  003236 155L     0001  003240 160L     0001  000170 170G
0001  000660 1998L    0001  000663 1999L    0001  000135 20L      0001  000666 2000L    0001  001226 2005L
0001  001233 2007L    0001  000210 201G     0001  001253 2010L    0001  001256 2015L    0001  001274 2017L
0001  001330 2020L    0001  001336 2025L    0001  001376 2030L    0001  001403 2032L    0001  001426 2035L
0001  001431 2040L    0001  001452 2042L    0001  001511 2045L    0001  001517 2050L    0001  001562 2055L
0001  001610 2060L    0001  001614 2065L    0001  000251 225G     0001  000150 25L      0001  001620 2998L
0001  001646 2999L    0001  001651 3000L    0001  002473 3005L    0001  002500 3007L    0001  002520 3010L
0001  002523 3015L    0001  002541 3017L    0001  002575 3020L    0001  002603 3025L    0001  002643 3030L
0001  002650 3032L    0001  002673 3035L    0001  002676 3040L    0001  002717 3042L    0001  002756 3045L
0001  002764 3050L    0001  003027 3055L    0001  003055 3060L    0001  003061 3065L    0001  003065 3998L
0001  003113 3999L    0001  000212 40L      0001  003116 4000L    0001  000731 4005L    0001  000736 4010L
0001  000737 4012L    0001  000756 4015L    0001  000757 4017L    0001  000774 4020L    0001  001010 4022L
0001  001025 4025L    0001  001033 4030L    0001  001071 4035L    0001  001105 4037L    0001  001122 4040L
0001  000237 45L      0001  001130 4998L    0001  001134 4999L    0001  000072 5L       0001  001137 50L
0001  001773 500G     0001  001137 5000L    0001  002177 5005L    0001  002204 5010L    0001  002205 5012L
0001  002224 5015L    0001  002225 5017L    0001  002242 5020L    0001  002256 5022L    0001  002273 5025L
0001  002301 5030L    0001  002337 5035L    0001  002353 5037L    0001  002370 5040L    0001  002104 537G
0001  002125 546G     0001  001702 55L      0001  002376 5998L    0001  002402 5999L    0001  001705 60L
0001  002405 6000L    0001  001753 65L      0001  001762 85L      0000  000000 885F     0000  000011 886F
0000  000016 887F     0000  000023 888F     0000  000026 889F     0001  002014 90L      0001  002032 95L
0001  003300 995L     0001  003306 996L     0001  003314 997L     0001  003323 998L     0001  003331 999L
```

```
00101   1*          SUBROUTINE CNFAST
00101   2*    C     AN EXPLICIT EXECUTION SUBROUTINE FOR SINDA    FORTRAN V
00101   3*    C     THE SHORT PSEUDO COMPUTE SEQUENCE IS REQUIRED
00101   4*    C     NODES WITH CSG BELOW DTIMEI RECEIVE STEADY STATE SOLUTION
00101   5*    C     NO BACKING UP IS DONE OR ALLOWED.
00103   6*          INCLUDE COMM,LIST
00104   6*          COMMON /TITLE/H(1) /TEMP/T(1) /CAP/C(1) /SOURCE/Q(1) /COND/G(1)
00105   6*          COMMON /PC1/NSQ1(1) /PC2/NSQ2(1) /KONST/K(1) /ARRAY/A(1)
00106   6*          COMMON /FIXCON/KON(1) /XSPACE/NDIM,NTH,X(1)
00107   6*          COMMON /DIMENS/ NND,NNA,NNT,NGT,NCT,NAT,LSQ1,LSQ2
00110   6*          DIMENSION CON(1),XK(1),NX(1)
00111   6*          EQUIVALENCE (KON(1),CON(1)),(K(1),XK(1)),(X(1),NX(1))
00112   6*          END
00113   7*          INCLUDE DEFF,LIST
00113   7*    C*************  CONTROL CONSTANT DEFINITIONS AND NAMES  ****************
00113   7*    C     CONTROL CONSTANT 1 CONTAINS THE NEW PROBLEM TIME              (TIMEN)
00113   7*    C     CONTROL CONSTANT 2 CONTAINS THE TIME STEP USED               (DTIMEU)
00113   7*    C     CONTROL CONSTANT 3 CONTAINS THE PROBLEM STOP TIME            (TIMEND)
00113   7*    C     CONTROL CONSTANT 4 CONTAINS THE TIME STEP FACTOR,EXPLICIT   (CSGFAC)
00113   7*    C     CC5 IS THE INPUT NUMBER OF ITERATION DO LOOPS, INTEGER        (NLOOP)
00113   7*    C     CC6 CONTAINS THE DIFFUSION TEMPERATURE CHANGE ALLOWED        (DTMPCA)
00113   7*    C     CC7 CONTAINS THE OUTPUT EACH ITERATION SWITCH                (OPEITR)
00113   7*    C     CC8 CONTAINS THE MAXIMUM ALLOWED TIME STEP                   (DTIMEH)
00113   7*    C     CC9 CONTAINS THE NEW ARITHMETIC TEMP. DAMPING FACTOR          (DAMPA)
00113   7*    C     CC10 CONTAINS THE NEW DIFFUSION TEMP. DAMPING FACTOR          (DAMPD)
00113   7*    C     CC11 CONTAINS THE MAXIMUM ALLOWED ARITHMETIC TEMP. CHANGE   (ATMPCA)
00113   7*    C     CC12 CONTAINS THE BACKUP SWITCH CHECKED AFTER VARIABLES      (BACKUP)
00113   7*    C     CC13 CONTAINS THE PRESENT TIME OR PROBLEM START TIME          (TIMEO)
00113   7*    C     CC14 CONTAINS THE MEAN TIME BETWEEN AN ITERATION              (TIMEM)
00113   7*    C     CC15 CONTAINS THE DIFFUSION TEMPERATURE CHANGE CALCULATED   (DTMPCC)
00113   7*    C     CC16 CONTAINS ARITHMETIC TEMPERATURE CHANGE CALCULATED       (ATMPCC)
00113   7*    C     CONTROL CONSTANT 17 IS RESERVED FOR THE C/SG MINIMUM         (CSGMIN)
```

CNFAST,CNFAST

```
00113     7*   C    CONTROL CONSTANT 18 CONTAINS THE OUTPUT INTERVAL          (OUTPUT)
00113     7*   C    CC19 CONTAINS THE ARITHMETIC RELAXATION CRITERIA ALLOWED  (ARLXCA)
00113     7*   C    CC20 CONTAINS THE NUMBER OF RELAXATION LOOPS USED,INTEGER (LOOPCT)
00113     7*   C    CC21 CONTAINS THE MINIMUM ALLOWED TIME STEP               (DTIMEL)
00113     7*   C    CC22 IS FOR THE INPUT TIME STEP IMPLICIT                  (DTIMEI)
00113     7*   C    CC23 CONTAINS THE C/SG MAXIMUM                            (CSGMAX)
00113     7*   C    CC24 CONTAINS THE C/SG RANGE ALLOWED                      (CSGRAL)
00113     7*   C    CC25 CONTAINS THE C/SG RANGE CALCULATED                   (CSGRCL)
00113     7*   C    CC26 CONTAINS THE DIFFUSION RELAXATION CRITERIA ALLOWED   (DRLXCA)
00113     7*   C    CC27 CONTAINS THE DIFFUSION RELAXATION CHANGE CALCULATED  (DRLXCC)
00113     7*   C    CC28 CONTAINS THE LINE COUNTER, INTEGER                   (LINECT)
00113     7*   C    CC29 CONTAINS THE PAGE COUNTER, INTEGER                   (PAGECT)
00113     7*   C    CC30 CONTAINS ARITHMETIC RELAXATION CHANGE CALCULATED     (ARLXCC)
00113     7*   C    CC31 IS INDICATOR, 0=THERMAL SPCS,1=THERMAL LPCS,2=GENERAL (LSPCS)
00113     7*   C    CC32 CONTAINS THE ENERGY BALANCE OF THE SYSTEM, IN - OUT  (ENGBAL)
00113     7*   C    CC33 CONTAINS THE DESIRED ENERGY BALANCE, USER INPUT      (BALENG)
00113     7*   C    CC34 CONTAINS THE NOCOPY SWITCH FOR MATRIX USERS          (NOCOPY)
00113     7*   C    CC35 CONTAINS RELATIVE NODE NUMBER OF CSGMIN
00113     7*   C    CC36 CONTAINS RELATIVE NODE NUMBER OF DTMPCC
00113     7*   C    CC37 CONTAINS RELATIVE NODE NUMBER OF ARLXCC
00113     7*   C    CC38 CONTAINS RELATIVE NODE NUMBER OF ATMPCC
00113     7*   C    CC39-40-41-42-43 CONTAIN DUMMY INTEGER CONSTANTS    (I-J-K-L-MTEST)
00113     7*   C    CC44-45-46-47-48 CONTAIN DUMMY FLOATING CONSTANTS  (R-S-T-U-VTEST)
00113     7*   C    CC49 IS THE QUASI-LINEARIZATION INTERVAL FOR CINDSM       (LAXFAC)
00113     7*   C    CC50 IS NOT USED AT PRESENT
00114     7*        END
00115     8*        IF(KON(5).LE.0) KON(5) = 1
00117     9*        IF(CON(8).LE.0.) CON(8) = 1.E+8
00121    10*        IF(CON(9).LE.0.) CON(9) = 1.0
00123    11*        IF(CON(18).LE.0.) GO TO 999
00125    12*        IF(CON(19).LE.0.) CON(19) = 1.E+8
00127    13*        IF(CON(21).LE.0.) GO TO 998
00131    14*        IF(KON(31).NE.0) GO TO 995
00133    15*        PASS = -1.0
00134    16*        NNC = NNA+NND
00135    17*        IE = NTH
00136    18*        NLA = NDIM
00137    19*        NTH = NTH+NND
00140    20*        NDIM = NDIM-NND
00141    21*        IF(NDIM.LT.0) GO TO 997
00143    22*        NN = NND+1
00144    23*        TPRINT = CON(13)
00145    24*        TSTEP = CON(21)
00146    25*      5 TSUM = 0.0
00147    26*        IF(CON(13)+CON(18).GT.CON(3)) CON(18) = CON(3)-CON(13)
00151    27*     10 IF(TSTEP.GT.CON(8)) TSTEP = CON(8)
00153    28*        IF(TSTEP.LT.CON(21)) TSTEP = CON(21)*1.000001
00155    29*        IF(TSUM+TSTEP-CON(18)) 20,25,15
00160    30*     15 TSTEP = CON(18)-TSUM
00161    31*        GO TO 25
00162    32*     20 IF(TSUM+2.0*TSTEP.GT.CON(18)) TSTEP = 0.5*(CON(18)-TSUM)
00164    33*     25 CON(2) = TSTEP
00165    34*        CON(1) = TPRINT+TSUM+TSTEP
00166    35*        CON(14) = 0.5*(CON(1)+CON(13))
00167    36*        DO 30 I = 1,NND
00172    37*        O(I) = 0.0
00173    38*        LE = IE+I
00174    39*     30 X(LE) = 0.0
00176    40*        IF(NNA.LE.0) GO TO 40
00200    41*        DO 35 I = NN,NNC
```

CNFAST.CNFAST

```
00203   42*          Q(I) = 0.0
00204   43*       35 CONTINUE
00206   44*       40 KON(12) = 0
00207   45*          CALL VARBL1
00210   46*          IF(KON(12).NE.0) GO TO 10
00212   47*          IF(PASS.GT.0.) GO TO 45
00214   48*          PASS = 1.0
00215   49*          CON(1) = TPRINT
00216   50*          CON(2) = 0.0
00217   51*          CALL OUTCAL
00220   52*          CON(1) = TPRINT+TSTEP
00221   53*          CON(2) = TSTEP
00222   54*       45 J1 = 0
00223   55*          J2 = 1
00224   56*          DO 85 I = 1,NND
00227   57*          LE = IE+I
00230   58*          INCLUDE VARC,LIST
00231   58*          IF(FLD(1,1,NSQ1(J1+1)).EQ.0) GO TO 2000
00233   58*          NTYPE = FLD(0,5,NSQ2(J2))                                          *NEW
00234   58*          LA = FLD(5,17,NSQ2(J2))                                            *NEW
00235   58*          LK = FLD(22,14,NSQ2(J2))                                           *NEW
00236   58*          GO TO (1005,1010,1015,1020,1025,1030,1035,1040,1045), NTYPE        **-3
00237   58*     1005 CALL D1D1WM(T(I),A(LA),XK(LK),C(I))
00240   58*          GO TO 1999
00241   58*     1010 CALL D1D1WM(T(I),A(LA),XK(LK),C1)
00242   58*     1012 J2 = J2+1
00243   58*          LA = FLD(5,17,NSQ2(J2))                                            *NEW
00244   58*          LK = FLD(22,14,NSQ2(J2))                                           *NEW
00245   58*          CALL D1D1WM(T(I),A(LA),XK(LK),C2)                                  **-2
00246   58*          GO TO 1998
00247   58*     1015 C1 = XK(LK)*XK(LA)
00250   58*          GO TO 1012
00251   58*     1020 CALL D1D1WM(T(I),A(LA),XK(LK),C1)
00252   58*          J2 = J2+1
00253   58*          LA = FLD(5,17,NSQ2(J2))                                            *NEW
00254   58*          LK = FLD(22,14,NSQ2(J2))                                           *NEW
00255   58*          C2 = XK(LK)*XK(LA)                                                 **-2
00256   58*          GO TO 1998
00257   58*     1025 CALL PLYAWM(A(LA),T(I),A(LA+1),XK(LK),C(I))
00260   58*          GO TO 1999
00261   58*     1030 CALL PLYAWM(A(LA),T(I),A(LA+1),XK(LK),C1)
00262   58*     1032 J2 = J2+1
00263   58*          LA = FLD(5,17,NSQ2(J2))                                            *NEW
00264   58*          LK = FLD(22,14,NSQ2(J2))                                           *NEW
00265   58*          CALL PLYAWM(A(LA),T(I),A(LA+1),XK(LK),C2)                          **-2
00266   58*          GO TO 1998
00267   58*     1035 C1 = XK(LK)*XK(LA)
00270   58*          GO TO 1032
00271   58*     1040 CALL PLYAWM(A(LA),T(I),A(LA+1),XK(LK),C1)
00272   58*          J2 = J2+1
00273   58*          LA = FLD(5,17,NSQ2(J2))                                            *NEW
00274   58*          LK = FLD(22,14,NSQ2(J2))                                           *NEW
00275   58*          C2 = XK(LK)*XK(LA)                                                 **-2
00276   58*          GO TO 1998
00277   58*     1045 CALL D2D1WM(T(I),CON(14),A(LA),XK(LK),C(I))
00300   58*          GO TO 1999
00301   58*     1998 C(I) = C1+C2
00302   58*     1999 J2 = J2+1
00303   58*     2000 CONTINUE
00304   58*          END
```

CNFAST,CNFAST

```
59*          INCLUDE VARQ,LIST
59*          IF(FLD(4,1,NSQ1(J1+1)).EQ.0) GO TO 5000
59*          NTYPE = FLD(0,5,NSQ2(J2))                                          *NEW
59*          LA = FLD(5,17,NSQ2(J2))                                            *NEW
59*          LK = FLD(22,14,NSQ2(J2))                                           *NEW
59*          GO TO (4005,4010,4015,4020,4025,4030,4035,4040,4030), NTYPE        **-3
59*     4005 Q(I) = XK(LK)+Q(I)
59*          GO TO 4999
59*     4010 Q1 = 0.0
59*     4012 CALL D1D1WM(T(I),A(LA),XK(LK),Q2)
59*          GO TO 4998
59*     4015 Q1 = 0.0
59*     4017 CALL D1D1WM(CON(14),A(LA),XK(LK),Q2)
59*          GO TO 4998
59*     4020 CALL D1D1WM(CON(14),A(LA),XK(LK),Q1)
59*     4022 J2 = J2+1
59*          LA = FLD(5,17,NSQ2(J2))                                            *NEW
59*          LK = FLD(22,14,NSQ2(J2))                                           *NEW
59*          GO TO 4017                                                         **-2
59*     4025 Q1 = XK(LK)*XK(LA)
59*          GO TO 4022
59*     4030 CALL D1D1WM(CON(14),A(LA),XK(LK),Q1)
59*          J2 = J2+1
59*          LA = FLD(5,17,NSQ2(J2))                                            *NEW
59*          LK = FLD(22,14,NSQ2(J2))                                           *NEW
59*          Q2 = XK(LK)*XK(LA)                                                 **-2
59*          GO TO 4998
59*     4035 CALL D1D1WM(CON(14),A(LA),XK(LK),Q1)
59*     4037 J2 = J2+1
59*          LA = FLD(5,17,NSQ2(J2))                                            *NEW
59*          LK = FLD(22,14,NSQ2(J2))                                           *NEW
59*          GO TO 4012                                                         **-2
59*     4040 Q1 = XK(LK)*XK(LA)
59*          GO TO 4037
59*     4998 Q(I) = Q1+Q2+Q(I)
59*     4999 J2 = J2+1
59*     5000 CONTINUE
59*          END
60*       50 J1 = J1+1
61*          LG = FLD(5,16,NSQ1(J1))
62*          IF(LG.EQ.0) GO TO 85
63*          LTA = FLD(22,14,NSQ1(J1))
64*          INCLUDE VARG,LIST
64*          IF(FLD(2,1,NSQ1(J1)).EQ.0) GO TO 3000
64*          NTYPE = FLD(0,5,NSQ2(J2))                                          *NEW
64*          LA = FLD(5,17,NSQ2(J2))                                            *NEW
64*          LK = FLD(22,14,NSQ2(J2))                                           *NEW
64*          GOTO(2005,2010,2015,2020,2025,2030,2035,2040,2045,2050,2055,       **-3
64*         $     2060,2065), NTYPE
64*     2005 TM = (T(I)+T(LTA))/2.0
64*     2007 CALL D1D1WM(TM,A(LA),XK(LK),G(LG))
64*          GO TO 2999
64*     2010 TM = T(I)
64*          GO TO 2007
64*     2015 CALL D1D1WM(T(I),A(LA),XK(LK),G1)
64*     2017 J2 = J2+1
64*          LA = FLD(5,17,NSQ2(J2))                                            *NEW
64*          LK = FLD(22,14,NSQ2(J2))                                           *NEW
64*          CALL D1D1WM(T(LTA),A(LA),XK(LK),G2)                                **-2
64*          GO TO 2998
```

CNFAST.CNFAST

```
64*    2020 G1 = XK(LK)*XK(LA)
64*         GO TO 2017
64*    2025 CALL D1D1WM(T(I),A(LA),XK(LK),G1)
64*         J2 = J2+1
64*         LA = FLD(5,17,NSQ2(J2))                                  *NEW
64*         LK = FLD(22,14,NSQ2(J2))                                 *NEW
64*         G2 = XK(LK)*XK(LA)                                       **-2
64*         GO TO 2998
64*    2030 TM = (T(I)+T(LTA))/2.0
64*    2032 CALL PLYAWM(A(LA),TM,A(LA+1),XK(LK),G(LG))
64*         GO TO 2999
64*    2035 TM = T(I)
64*         GO TO 2032
64*    2040 CALL PLYAWM(A(LA),T(I),A(LA+1),XK(LK),G1)
64*    2042 J2 = J2+1
64*         LA = FLD(5,17,NSQ2(J2))                                  *NEW
64*         LK = FLD(22,14,NSQ2(J2))                                 *NEW
64*         CALL PLYAWM(A(LA),T(LTA),A(LA+1),XK(LK),G2)              **-2
64*         GO TO 2998
64*    2045 G1 = XK(LK)*XK(LA)
64*         GO TO 2042
64*    2050 CALL PLYAWM(A(LA),T(I),A(LA+1),XK(LK),G1)
64*         J2 = J2+1
64*         LA = FLD(5,17,NSQ2(J2))                                  *NEW
64*         LK = FLD(22,14,NSQ2(J2))                                 *NEW
64*         G2 = XK(LK)*XK(LA)                                       **-2
64*         GO TO 2998
64*    2055 TM = (T(I)+T(LTA))/2.0
64*         CALL D2D1WM(TM,CON(14),A(LA),XK(LK),G(LG))
64*         GO TO 2999
64*    2060 TM = T(LTA)
64*         GO TO 2007
64*    2065 TM = T(LTA)
64*         GO TO 2032
64*    2998 G(LG) = 1./(1./G1+1./G2)
64*         IF(FLD(3,1,NSQ1(J1)).EQ.1) G(LG) = G1*G2
64*    2999 J2 = J2+1
64*    3000 CONTINUE
64*         END
65*         IF(FLD(3,1,NSQ1(J1)).EQ.0) GO TO 55
66*         T1 = T(I)+460.0
67*         T2 = T(LTA)+460.0
68*         GV = G(LG)*(T1*T1+T2*T2)*(T1+T2)
69*         GO TO 60
70*      55 GV = G(LG)
71*      60 Q(I) = Q(I)+GV*T(LTA)
72*         X(LE) = X(LE)+GV
73*         IF(LTA.GT.NND.OR.FLD(21,1,NSQ1(J1)).EQ.1) GO TO 65
74*         LEA = IE+LTA
75*         X(LEA) = X(LEA)+GV
76*         Q(LTA) = Q(LTA)+GV*T(I)
77*      65 IF(NSQ1(J1).GT.0) GO TO 50
78*      85 CONTINUE
79*         CKM = 1.E+8
80*         TCGM = 0.0
81*         DO 105 I = 1,NND
82*         LE = IE+I
83*         T1 = C(I)/X(LE)
84*         IF(T1.GE.CKM) GO TO 90
85*         CKM = T1
```

```
00507    86*          KON(35) = I
00510    87*       90 IF(TSTEP.GT.T1) GO TO 95
00512    88*          T1 = T(I)+TSTEP*(Q(I)-X(LE)*T(I))/C(I)
00513    89*          GO TO 100
00514    90*       95 T1 = Q(I)/X(LE)
00515    91*      100 T2 = ABS(T1-T(I))
00516    92*          T(I) = T1
00517    93*          IF(T2.LT.TCGM) GO TO 105
00521    94*          TCGM = T2
00522    95*          KON(15) = I
00523    96*      105 CONTINUE
00525    97*          CON(15) = TCGM
00526    98*          CON(17) = CKM
00527    99*          IF(CKM.LE.0.) GO TO 996
00531   100*          IF(NNA.LE.0) GO TO 160
00533   101*          LAX = KON(5)
00534   102*          DAMPN = CON(9)
00535   103*          DAMPO = 1.0-DAMPN
00536   104*          DO 150 I = 1,LAX
00541   105*          KON(20) = I
00542   106*          RLX = 0.0
00543   107*          JJ1 = J1
00544   108*          JJ2 = J2
00545   109*          DO 145 L = NN,NNC
00550   110*          SUMC = 0.0
00551   111*          SUMCV = 0.0
00552   112*          IF(I.GT.1) GO TO 6000
00554   113*          INCLUDE VRQ2,LIST
00555   113*          IF(FLD(4,1,NSQ1(JJ1+1)).EQ.0) GO TO 6000
00557   113*          NTYPE = FLD(0,5,NSQ2(JJ2))                                      *NEW
00560   113*          LA = FLD(5,17,NSQ2(JJ2))                                        *NEW
00561   113*          LK = FLD(22,14,NSQ2(JJ2))                                       *NEW
00562   113*          GO TO (5005,5010,5015,5020,5025,5030,5035,5040,5030), NTYPE     **-3
00563   113*     5005 Q(L) = XK(LK)+Q(L)
00564   113*          GO TO 5999
00565   113*     5010 Q1 = 0.0
00566   113*     5012 CALL D1D1WM(T(L),A(LA),XK(LK),Q2)
00567   113*          GO TO 5998
00570   113*     5015 Q1 = 0.0
00571   113*     5017 CALL D1D1WM(CON(14),A(LA),XK(LK),Q2)
00572   113*          GO TO 5998
00573   113*     5020 CALL D1D1WM(CON(14),A(LA),XK(LK),Q1)
00574   113*     5022 JJ2 = JJ2+1
00575   113*          LA = FLD(5,17,NSQ2(JJ2))                                        *NEW
00576   113*          LK = FLD(22,14,NSQ2(JJ2))                                       *NEW
00577   113*          GO TO 5017                                                      **-2
00600   113*     5025 Q1 = XK(LK)*XK(LA)
00601   113*          GO TO 5022
00602   113*     5030 CALL D1D1WM(CON(14),A(LA),XK(LK),Q1)
00603   113*          JJ2 = JJ2+1
00604   113*          LA = FLD(5,17,NSQ2(JJ2))                                        *NEW
00605   113*          LK = FLD(22,14,NSQ2(JJ2))                                       *NEW
00606   113*          Q2 = XK(LK)*XK(LA)                                              **-2
00607   113*          GO TO 5998
00610   113*     5035 CALL D1D1WM(CON(14),A(LA),XK(LK),Q1)
00611   113*     5037 JJ2 = JJ2+1
00612   113*          LA = FLD(5,17,NSQ2(JJ2))                                        *NEW
00613   113*          LK = FLD(22,14,NSQ2(JJ2))                                       *NEW
00614   113*          GO TO 5012                                                      **-2
00615   113*     5040 Q1 = XK(LK)*XK(LA)
```

```
      GO TO 5037
5998 Q(L) = Q1+Q2+Q(L)
5999 JJ2 = JJ2+1
6000 CONTINUE
     END
 110 JJ1 = JJ1+1
     LG = FLD(5,16,NSQ1(JJ1))
     LTA = FLD(22,14,NSQ1(JJ1))
     IF(I.GT.1) GO TO 4000
     INCLUDE VRG2.LIST
C    CHECK FOR RADIATION CONDUCTOR
     IF(FLD(2,1,NSQ1(JJ1)).EQ.0) GO TO 4000
     NTYPE = FLD(0,5,NSQ2(JJ2))                                          *NEW
     LA = FLD(5,17,NSQ2(JJ2))                                            *NEW
     LK = FLD(22,14,NSQ2(JJ2))                                           *NEW
     GOTO(3005,3010,3015,3020,3025,3030,3035,3040,3045,3050,3055,        **-3
    $     3060,3065), NTYPE
3005 TM = (T(L)+T(LTA))/2.0
3007 CALL D1D1WM(TM,A(LA),XK(LK),G(LG))
     GO TO 3999
3010 TM = T(L)
     GO TO 3007
3015 CALL D1D1WM(T(L),A(LA),XK(LK),G1)
3017 JJ2 = JJ2+1
     LA = FLD(5,17,NSQ2(JJ2))                                            *NEW
     LK = FLD(22,14,NSQ2(JJ2))                                           *NEW
     CALL D1D1WM(T(LTA),A(LA),XK(LK),G2)                                 **-2
     GO TO 3998
3020 G1 = XK(LK)*XK(LA)
     GO TO 3017
3025 CALL D1D1WM(T(L),A(LA),XK(LK),G1)
     JJ2 = JJ2+1
     LA = FLD(5,17,NSQ2(JJ2))                                            *NEW
     LK = FLD(22,14,NSQ2(JJ2))                                           *NEW
     G2 = XK(LK)*XK(LA)                                                  **-2
     GO TO 3998
3030 TM = (T(L)+T(LTA))/2.0
3032 CALL PLYAWM(A(LA),TM,A(LA+1),XK(LK),G(LG))
     GO TO 3999
3035 TM = T(L)
     GO TO 3032
3040 CALL PLYAWM(A(LA),T(L),A(LA+1),XK(LK),G1)
3042 JJ2 = JJ2+1
     LA = FLD(5,17,NSQ2(JJ2))                                            *NEW
     LK = FLD(22,14,NSQ2(JJ2))                                           *NEW
     CALL PLYAWM(A(LA),T(LTA),A(LA+1),XK(LK),G2)                         **-2
     GO TO 3998
3045 G1 = XK(LK)*XK(LA)
     GO TO 3042
3050 CALL PLYAWM(A(LA),T(L),A(LA+1),XK(LK),G1)
     JJ2 = JJ2+1
     LA = FLD(5,17,NSQ2(JJ2))                                            *NEW
     LK = FLD(22,14,NSQ2(JJ2))                                           *NEW
     G2 = XK(LK)*XK(LA)                                                  **-2
     GO TO 3998
3055 TM = (T(L)+T(LTA))/2.0
     CALL D2D1WM(TM,CON(14),A(LA),XK(LK),G(LG))
     GO TO 3999
3060 TM = T(LTA)
     GO TO 3007
```

CNFAST.CNFAST

```
00712  119*     3065 TM = T(LTA)
00713  119*          GO TO 3032
00714  119*     3996 G(LG) = 1./(1./G1+1./G2)
00715  119*          IF(FLD(3,1,NSQ1(JJ1)).EQ.1) G(LG) = G1*G2
00717  119*     3999 JJ2 = JJ2+1
00720  119*     4000 CONTINUE
00721  119*          END
00722  120*          IF(FLD(3,1,NSQ1(JJ1)).EQ.0) GO TO 115
00724  121*          T1 = T(L)+460.0
00725  122*          T2 = T(LTA)+460.0
00726  123*          GV = G(LG)*(T1*T1+T2*T2)*(T1+T2)
00727  124*          GO TO 120
00730  125*      115 GV = G(LG)
00731  126*          T2 = T(LTA)
00732  127*      120 SUMC = SUMC+GV
00733  128*          SUMCV = SUMCV+GV*T2
00733  129*     C    CHECK FOR LAST CONDUCTOR TO THIS NODE
00734  130*          IF(NSQ1(JJ1).GT.0) GO TO 110
00736  131*          T1 = DAMPN*(SUMCV+Q(L))/SUMC+DAMPO*T(L)
00737  132*          T2 = ABS(T(L)-T1)
00740  133*          IF(RLX.GE.T2) GO TO 140
00742  134*          RLX = T2
00743  135*          KON(37) = L
00744  136*      140 T(L) = T1
00745  137*      145 CONTINUE
00747  138*          IF(RLX.LE.CON(19)) GO TO 155
00751  139*      150 CONTINUE
00753  140*      155 CON(30) = RLX
00754  141*      160 CALL VARBL2
00755  142*          CON(13) = CON(1)
00756  143*          TSUM = TSUM+TSTEP
00757  144*          TSTEP = CKM
00760  145*          IF(TSUM.LT.CON(18)) GO TO 10
00762  146*          TPRINT = TPRINT+TSUM
00763  147*          CALL OUTCAL
00764  148*          IF(CON(1)*1.000001.LT.CON(3)) GO TO 5
00766  149*          NTH = IE
00767  150*          NDIM = NLA
00770  151*          RETURN
00771  152*      995 WRITE(6,885)
00773  153*          GO TO 1000
00774  154*      996 WRITE (6,886)
00776  155*          GO TO 1000
00777  156*      997 WRITE(6,887) NDIM
01002  157*          GO TO 1000
01003  158*      998 WRITE (6,888)
01005  159*          GO TO 1000
01006  160*      999 WRITE(6,889)
01010  161*     1000 CALL OUTCAL
01011  162*          CALL EXIT
01012  163*      885 FORMAT(46H CNFAST REQUIRES SHORT PSEUDO-COMPUTE SEQUENCE)
01013  164*      886 FORMAT(22H C/SK ZERO OR NEGATIVE)
01014  165*     887  FORMAT(I8,20H LOCATIONS AVAILABLE)
01015  166*      888 FORMAT(10H NO DTIMEL)
01016  167*      889 FORMAT(19H NO OUTPUT INTERVAL)
01017  168*          END

   END OF UNIVAC 1108 FORTRAN V COMPILATION.     0 *DIAGNOSTIC* MESSAGE(S)
  CNFAST          SYMBOLIC
  CNFAST   CODE   RELOCATABLE
```

```
                              CNEXPN,CNEXPN
QIW   FOR,*        CNEXPN,CNEXPN                                                          21 FEB 71
UNIVAC 1108 FORTRAN V ATHENA VERSION 131K-10D   CREATED ON 20 AUG 70
THIS COMPILATION WAS DONE ON 09 JUN 70 AT 14:00:19

  SUBROUTINE CNEXPN    ENTRY POINT 003536

  STORAGE USED (BLOCK, NAME, LENGTH)

        0001   *CODE         003551
        0000   *CONST+TEMP   000072
        0002   *SIMPLE VAR   000047
        0004   *ARRAYS       000000
        0005   *BLANK        000000
        0006   TITLE    000001
        0007   TEMP     000001
        0010   CAP      000001
        0011   SOURCE   000001
        0012   COND     000001
        0013   PC1      000001
        0014   PC2      000001
        0015   KONST    000001
        0016   ARRAY    000001
        0017   FIXCON   000001
        0020   XSPACE   000003
        0021   DIMENS   000010

  EXTERNAL REFERENCES (BLOCK, NAME)

        0022   VARBL1
        0023   D1D1WM
        0024   PLYAWM
        0025   D2D1WM
        0026   VARBL2
        0027   OUTCAL
        0030   EXIT
        0031   NERR2$
        0032   EXP
        0033   NWDU$
        0034   NIO2$
        0035   NER10$

  STORAGE ASSIGNMENT FOR VARIABLES (BLOCK, TYPE, RELATIVE LOCATION, NAME)

   0016 R 000000 A        0010 R 000000 C        0002 R 000015 CKM      0017 R 000000 CON     0002 R 000021 C1
   0002 R 000022 C2       0002 R 000040 DD       0002 R 000036 DELTA    0002 R 000037 DN      0012 R 000000 G
   0002 R 000034 GV       0002 R 000030 G1       0002 R 000031 G2       0006 R 000000 H       0002 I 000004 I
   0002 I 000002 IE       0002 I 000042 JJ1      0002 I 000043 JJ2      0002 I 000012 J1      0002 I 000013 J2
   0015 I 000000 K        0017 I 000000 KON      0002 I 000044 L        0002 I 000017 LA      0002 I 000041 LAX
   0002 I 000011 LE       0002 I 000035 LEA      0002 I 000025 LG       0002 I 000020 LK      0021 I 000006 LSQ1
   0021 I 000007 LSQ2     0002 I 000026 LTA      0002 I 000005 L1       0021 I 000005 NAT     0021 I 000004 NCT
   0020 I 000000 NDIM     0021 I 000003 NGT      0002 I 000003 NLA      0021 I 000001 NNA     0002 I 000001 NNC
   0021 I 000000 NNO      0021 I 000002 NNT      0013 I 000000 NSQ1     0014 I 000000 NSQ2    0020 I 000001 NTH
   0002 I 000016 NTYPE    0020   000002 NX       0002 R 000000 PASS     0011 R 000000 Q       0002 R 000023 Q1
   0002 R 000024 Q2       0002 R 000045 SUMC     0002 R 000046 SUMCV    0007 R 000000 T       0002 R 000014 TCGM
   0002 R 000027 TM       0002 R 0000?7 TPRINT   0002 R 000006 TSTEP    0002 R 000010 TSUM    0002 R 000032 T1
```

```
CNEXPN,CNExPN

0002 R 000033 T2      0020 R 000002 X       0015 R 000000 XK      0001   000124 10L     0001   003514 1000L
0001   000326 1005L   0001   000347 1010L   0001   003352 1011G   0001   000365 1012L   0001   000420 1015L
0001   000426 1020L   0001   000466 1025L   0001   000512 1030L   0001   000533 1032L   0001   000571 1035L
0001   000577 1040L   0001   000642 1045L   0001   002105 105L    0001   002137 115L    0001   002150 120L
0001   002156 125L    0001   002164 130L    0001   002514 135L    0001   003262 140L    0001   003265 145L
0001   000133 15L     0001   003332 165L    0001   003343 175L    0001   003374 180L    0001   003405 185L
0001   003434 190L    0001   003437 195L    0001   000664 1998L   0001   000667 1999L   0001   000205 200G
0001   000672 2000L   0001   001234 2005L   0001   001241 2007L   0001   001261 2010L   0001   001264 2015L
0001   001302 2017L   0001   001336 2020L   0001   001344 2025L   0001   001404 2030L   0001   001411 2032L
0001   001434 2035L   0001   001437 2040L   0001   001460 2042L   0001   001517 2045L   0001   001525 2050L
0001   001570 2055L   0001   001616 2060L   0001   001622 2065L   0001   000224 212G    0001   000255 232G
0001   000145 25L     0001   001626 2998L   0001   001654 2999L   0001   000160 30L     0001   001657 3000L
0001   002602 3005L   0001   002607 3007L   0001   002627 3010L   0001   002632 3015L   0001   002650 3017L
0001   002704 3020L   0001   002712 3025L   0001   002752 3030L   0001   002757 3032L   0001   003002 3035L
0001   003005 3040L   0001   003026 3042L   0001   003065 3045L   0001   003073 3050L   0001   003136 3055L
0001   003164 3060L   0001   003170 3065L   0001   003174 3998L   0001   003222 3999L   0001   003225 4000L
0001   000737 4005L   0001   000744 4010L   0001   000745 4012L   0001   000764 4015L   0001   000765 4017L
0001   001002 4020L   0001   001016 4022L   0001   001033 4025L   0001   001041 4030L   0001   001077 4035L
0001   001113 4037L   0001   001130 4040L   0001   000236 45L     0001   001136 4998L   0001   001142 4999L
0001   000112 5L      0001   001145 50L     0001   001145 5000L   0001   002306 5005L   0001   002313 5010L
0001   002314 5012L   0001   002333 5015L   0001   002334 5017L   0001   002351 5020L   0001   002365 5022L
0001   002402 5025L   0001   002002 503G    0001   002410 5030L   0001   002446 5035L   0001   002462 5037L
0001   002477 5040L   0001   002111 535G    0001   001714 55L     0001   002213 573G    0001   002505 5998L
0001   002511 5999L   0001   001717 60L     0001   002514 6000L   0001   002234 602G    0001   001765 65L
0001   001774 85L     0000   000000 885F    0000   000011 886F    0000   000016 887F    0000   000023 888F
0000   000030 889F    0001   002023 90L     0001   002061 95L     0001   003456 995L    0001   003464 996L
0001   003472 997L    0001   003500 998L    0001   003507 999L

00101     1*          SUBROUTINE CNExPN
00101     2*    C     EXPLICIT EXPONENTIAL DIFFERENCING ROUTINE FOR SINDA   FORTRAN V
00101     3*    C     THE SHORT PSEUDO-COMPUTE SEQUENCE IS REQUIRED
00103     4*          INCLUDE COMM,LIST
00104     4*          COMMON /TITLE/H(1) /TEMP/T(1) /CAP/C(1) /SOURCE/Q(1) /COND/G(1)
00105     4*          COMMON /PC1/NSQ1(1) /PC2/NSQ2(1) /KONST/K(1) /ARRAY/A(1)
00106     4*          COMMON /FIXCON/KON(1) /XSPACE/NDIM,NTH,X(1)
00107     4*          COMMON /DIMENS/ NND,NNA,NNT,NGT,NCT,NAT,LSQ1,LSQ2
00110     4*          DIMENSION CON(1),XK(1),NX(1)
00111     4*          EQUIVALENCE (KON(1),CON(1)),(K(1),XK(1)),(X(1),NX(1))
00112     4*          END
00113     5*          INCLUDE DEFF,LIST
00113     5*    C*************  CONTROL CONSTANT DEFINITIONS AND NAMES  ******,*********
00113     5*    C     CONTROL CONSTANT 1 CONTAINS THE NEW PROBLEM TIME              (TIMEN)
00113     5*    C     CONTROL CONSTANT 2 CONTAINS THE TIME STEP USED               (DTIMEU)
00113     5*    C     CONTROL CONSTANT 3 CONTAINS THE PROBLEM STOP TIME            (TIMEND)
00113     5*    C     CONTROL CONSTANT 4 CONTAINS THE TIME STEP FACTOR,EXPLICIT (CSGFAC)
00113     5*    C     CC5 IS THE INPUT NUMBER OF ITERATION DO LOOPS, INTEGER      (NLOOP)
00113     5*    C     CC6 CONTAINS THE DIFFUSION TEMPERATURE CHANGE ALLOWED      (DTMPCA)
00113     5*    C     CC7 CONTAINS THE OUTPUT EACH ITERATION SWITCH              (OPEITR)
00113     5*    C     CC8 CONTAINS THE MAXIMUM ALLOWED TIME STEP                 (DTIMEH)
00113     5*    C     CC9 CONTAINS THE NEW ARITHMETIC TEMP. DAMPING FACTOR        (DAMPA)
00113     5*    C     CC10 CONTAINS THE NEW DIFFUSION TEMP. DAMPING FACTOR        (DAMPD)
00113     5*    C     CC11 CONTAINS THE MAXIMUM ALLOWED ARITHMETIC TEMP. CHANGE (ATMPCA)
00113     5*    C     CC12 CONTAINS THE BACKUP SWITCH CHECKED AFTER VARIABLES   (BACKUP)
00113     5*    C     CC13 CONTAINS THE PRESENT TIME OR PROBLEM START TIME        (TIMEO)
00113     5*    C     CC14 CONTAINS THE MEAN TIME BETWEEN AN ITERATION            (TIMEM)
00113     5*    C     CC15 CONTAINS THE DIFFUSION TEMPERATURE CHANGE CALCULATED (DTMPCC)
00113     5*    C     CC16 CONTAINS ARITHMETIC TEMPERATURE CHANGE CALCULATED    (ATMPCC)
```

CNEXPN.CNEXPN

```
00113     5*   C     CONTROL CONSTANT 17 IS RESERVED FOR THE C/SG MINIMUM      (CSGMIN)
00113     5*   C     CONTROL CONSTANT 18 CONTAINS THE OUTPUT INTERVAL          (OUTPUT)
00113     5*   C     CC19 CONTAINS THE ARITHMETIC RELAXATION CRITERIA ALLOWED  (ARLXCA)
00113     5*   C     CC20 CONTAINS THE NUMBER OF RELAXATION LOOPS USED,INTEGER (LOOPCT)
00113     5*   C     CC21 CONTAINS THE MINIMUM ALLOWED TIME STEP               (DTIMEL)
00113     5*   C     CC22 IS FOR THE INPUT TIME STEP IMPLICIT                  (DTIMEI)
00113     5*   C     CC23 CONTAINS THE C/SG MAXIMUM                            (CSGMAX)
00113     5*   C     CC24 CONTAINS THE C/SG RANGE ALLOWED                      (CSGRAL)
00113     5*   C     CC25 CONTAINS THE C/SG RANGE CALCULATED                   (CSGRCL)
00113     5*   C     CC26 CONTAINS THE DIFFUSION RELAXATION CRITERIA ALLOWED   (DRLXCA)
00113     5*   C     CC27 CONTAINS THE DIFFUSION RELAXATION CHANGE CALCULATED  (DRLXCC)
00113     5*   C     CC28 CONTAINS THE LINE COUNTER, INTEGER                   (LINECT)
00113     5*   C     CC29 CONTAINS THE PAGE COUNTER, INTEGER                   (PAGECT)
00113     5*   C     CC30 CONTAINS ARITHMETIC RELAXATION CHANGE CALCULATED     (ARLXCC)
00113     5*   C     CC31 IS INDICATOR, 0=THERMAL SPCS,1=THERMAL LPCS,2=GENERAL (LSPCS)
00113     5*   C     CC32 CONTAINS THE ENERGY BALANCE OF THE SYSTEM, IN - OUT  (ENGBAL)
00113     5*   C     CC33 CONTAINS THE DESIRED ENERGY BALANCE, USER INPUT      (BALENG)
00113     5*   C     CC34 CONTAINS THE NOCOPY SWITCH FOR MATRIX USERS          (NOCOPY)
00113     5*   C     CC35 CONTAINS RELATIVE NODE NUMBER OF CSGMIN
00113     5*   C     CC36 CONTAINS RELATIVE NODE NUMBER OF DTMPCC
00113     5*   C     CC37 CONTAINS RELATIVE NODE NUMBER OF ARLXCC
00113     5*   C     CC38 CONTAINS RELATIVE NODE NUMBER OF ATMPCC
00113     5*   C     CC39-40-41-42-43 CONTAIN DUMMY INTEGER CONSTANTS   (I-J-K-L-MTEST)
00113     5*   C     CC44-45-46-47-48 CONTAIN DUMMY FLOATING CONSTANTS  (R-S-T-U-VTEST)
00113     5*   C     CC49 IS THE QUASI-LINEARIZATION INTERVAL FOR CINDSM       (LAXFAC)
00113     5*   C     CC50 IS NOT USED AT PRESENT
00114     5*         END
00115     6*         IF(CON(4).LE.0.0) CON(4) = 1.0
00117     7*         IF(KON(5).LE.0) KON(5) = 1
00121     8*         IF(CON(6).LE.0.) CON(6) = 1.E+8
00123     9*         IF(CON(8).LE.0.) CON(8) = 1.E+8
00125    10*         IF(CON(9).LE.0.) CON(9) = 1.0
00127    11*         IF(CON(11).LE.0.) CON(11) = 1.E+8
00131    12*         IF(CON(18).LE.0.) GO TO 999
00133    13*         IF(CON(19).LE.0.) CON(19) = 1.E+8
00135    14*         IF(KON(31).NE.0) GO TO 995
00137    15*         PASS = -1.0
00140    16*         NNC = NND+NNA
00141    17*         IE = NTH
00142    18*         NLA = NDIM
00143    19*         NTH = NTH+NNC
00144    20*         NDIM = NDIM-NNC
00144    21*   C     CHECK FOR EXTRA LOCATIONS FOR CALCULATED NODES
00145    22*         I = NLA-NNC
00146    23*         IF(I.LT.0) GO TO 998
00150    24*         L1 = NND+1
00151    25*         TSTEP = CON(18)
00152    26*         TPRINT = CON(13)
00152    27*   C     INITALIZE TIME SUM BETWEEN OUTPUT INTERVALS
00153    28*       5 TSUM = 0.0
00153    29*   C     DOES OLD TIME PLUS THE OUTPUT INTERVAL EXCEED THE STOP TIME
00154    30*         IF(CON(13)+CON(18).LE.CON(3)) GO TO 10
00154    31*   C     DONT EXCEED IT
00156    32*         CON(18) = CON(3)-CON(13)
00156    33*   C     IS THE TIME STEP LARGER THAN ALLOWED
00157    34*      10 IF(TSTEP.LE.CON(8)) GO TO 15
00161    35*         TSTEP = CON(8)
00161    36*   C     DOES THE TIME SUM PLUS THE TIME STEP EXCEED OUTPUT INTERVAL
00162    37*      15 IF(TSUM+TSTEP-CON(18)) 25,30,20
00162    38*   C     DONT EXCEED IT
```

```
00165   39*        20 TSTEP = CON(18)-TSUM
00166   40*           GO TO 30
00166   41*     C     DOES TIME SUM PLUS TWO TIME STEPS EXCEED OUTPUT INTERVAL
00167   42*        25 IF(TSUM+2.0*TSTEP.LE.CON(18)) GO TO 30
00167   43*     C     APPROACH THE OUTPUT INTERVAL GRADUALLY
00171   44*           TSTEP = (CON(18)-TSUM)/2.0
00171   45*     C     STORE DELTA TIME STEP IN THE CONSTANTS
00172   46*        30 CON(2) = TSTEP
00172   47*     C     IS THE TIME STEP USED LESS THAN THE TIME STEP ALLOWED
00173   48*           IF(TSTEP.LT.CON(21)) GO TO 997
00173   49*     C     CALCULATE THE NEW TIME
00175   50*           CON(1) = TPRINT+TSUM+TSTEP
00175   51*     C     COMPUTE THE MEAN TIME BETWEEN ITERATIONS
00176   52*           CON(14) = (CON(1)+CON(13))/2.0
00176   53*     C     ZERO OUT ALL SOURCE LOCATIONS AND EXTRA LOCATIONS
00177   54*           DO 35 I = 1,NND
00202   55*           LE = IE+I
00203   56*           X(LE) = 0.0
00204   57*           Q(I) = 0.0
00205   58*        35 CONTINUE
00205   59*     C     SHIFT THE ARITHMETIC TEMPERATURES INTO THE EXTRA LOCATIONS
00207   60*           IF(NNA.LE.0) GO TO 45
00211   61*           DO 40 I = L1,NNC
00214   62*           Q(I) = 0.0
00215   63*           LE = IE+I
00216   64*           X(LE) = T(I)
00217   65*        40 CONTINUE
00221   66*        45 KON(12) = 0
00222   67*           CALL VARBL1
00223   68*           IF(KON(12).NE.0) GO TO 10
00225   69*           J1 = 0
00226   70*           J2 = 1
00227   71*           TCGM = 0.0
00230   72*           CKM = 1.E+8
00230   73*     C     CALCULATE Q SUM AND G SUM
00231   74*           DO 85 I = 1,NND
00234   75*           LE = IE+I
00235   76*           INCLUDE VARC,LIST
00236   76*           IF(FLD(1,1,NSQ1(J1+1)).EQ.0) GO TO 2000
00240   76*           NTYPE = FLD(0,5,NSQ2(J2))                                *NEW
00241   76*           LA = FLD(5,17,NSQ2(J2))                                  *NEW
00242   76*           LK = FLD(22,14,NSQ2(J2))                                 *NEW
00243   76*           GO TO (1005,1010,1015,1020,1025,1030,1035,1040,1045), NTYPE   **-3
00244   76*      1005 CALL D1D1WM(T(I),A(LA),XK(LK),C(I))
00245   76*           GO TO 1999
00246   76*      1010 CALL D1D1WM(T(I),A(LA),XK(LK),C1)
00247   76*      1012 J2 = J2+1
00250   76*           LA = FLD(5,17,NSQ2(J2))                                  *NEW
00251   76*           LK = FLD(22,14,NSQ2(J2))                                 *NEW
00252   76*           CALL D1D1WM(T(I),A(LA),XK(LK),C2)                        **-2
00253   76*           GO TO 1998
00254   76*      1015 C1 = XK(LK)*XK(LA)
00255   76*           GO TO 1012
00256   76*      1020 CALL D1D1WM(T(I),A(LA),XK(LK),C1)
00257   76*           J2 = J2+1
00260   76*           LA = FLD(5,17,NSQ2(J2))                                  *NEW
00261   76*           LK = FLD(22,14,NSQ2(J2))                                 *NEW
00262   76*           C2 = XK(LK)*XK(LA)                                       **-2
00263   76*           GO TO 1998
00264   76*      1025 CALL PLYAWM(A(LA),T(I),A(LA+1),XK(LK),C(I))
```

CNEXPN,CNEXPN

```
76*         GO TO 1999
76*    1030 CALL PLYAWM(A(LA),T(I),A(LA+1),XK(LK),C1)
76*    1032 J2 = J2+1
76*         LA = FLD(5,17,NSQ2(J2))                                    *NEW
76*         LK = FLD(22,14,NSQ2(J2))                                   *NEW
76*         CALL PLYAWM(A(LA),T(I),A(LA+1),XK(LK),C2)                  **-2
76*         GO TO 1998
76*    1035 C1 = XK(LK)*XK(LA)
76*         GO TO 1032
76*    1040 CALL PLYAWM(A(LA),T(I),A(LA+1),XK(LK),C1)
76*         J2 = J2+1
76*         LA = FLD(5,17,NSQ2(J2))                                    *NEW
76*         LK = FLD(22,14,NSQ2(J2))                                   *NEW
76*         C2 = XK(LK)*XK(LA)                                         **-2
76*         GO TO 1998
76*    1045 CALL D2D1WM(T(I),CON(14),A(LA),XK(LK),C(I))
76*         GO TO 1999
76*    1998 C(I) = C1+C2
76*    1999 J2 = J2+1
76*    2000 CONTINUE
76*         END
77*         INCLUDE VARQ,LIST
77*         IF(FLD(4,1,NSQ1(J1+1)).EQ.0) GO TO 5000
77*         NTYPE = FLD(0,5,NSQ2(J2))                                  *NEW
77*         LA = FLD(5,17,NSQ2(J2))                                    *NEW
77*         LK = FLD(22,14,NSQ2(J2))                                   *NEW
77*         GO TO (4005,4010,4015,4020,4025,4030,4035,4040,4030), NTYPE **-3
77*    4005 Q(I) = XK(LK)+Q(I)
77*         GO TO 4999
77*    4010 Q1 = 0.0
77*    4012 CALL D1D1WM(T(I),A(LA),XK(LK),Q2)
77*         GO TO 4998
77*    4015 Q1 = 0.0
77*    4017 CALL D1D1WM(CON(14),A(LA),XK(LK),Q2)
77*         GO TO 4998
77*    4020 CALL D1D1WM(CON(14),A(LA),XK(LK),Q1)
77*    4022 J2 = J2+1
77*         LA = FLD(5,17,NSQ2(J2))                                    *NEW
77*         LK = FLD(22,14,NSQ2(J2))                                   *NEW
77*         GO TO 4017                                                 **-2
77*    4025 Q1 = XK(LK)*XK(LA)
77*         GO TO 4022
77*    4030 CALL D1D1WM(CON(14),A(LA),XK(LK),Q1)
77*         J2 = J2+1
77*         LA = FLD(5,17,NSQ2(J2))                                    *NEW
77*         LK = FLD(22,14,NSQ2(J2))                                   *NEW
77*         Q2 = XK(LK)*XK(LA)                                         **-2
77*         GO TO 4998
77*    4035 CALL D1D1WM(CON(14),A(LA),XK(LK),Q1)
77*    4037 J2 = J2+1
77*         LA = FLD(5,17,NSQ2(J2))                                    *NEW
77*         LK = FLD(22,14,NSQ2(J2))                                   *NEW
77*         GO TO 4012                                                 **-2
77*    4040 Q1 = XK(LK)*XK(LA)
77*         GO TO 4037
77*    4998 Q(I) = Q1+Q2+Q(I)
77*    4999 J2 = J2+1
77*    5000 CONTINUE
77*         END
78*      50 J1 = J1+1
```

CNEXPN.CNEXPN

```
      LG = FLD(5,16,NSQ1(J1))
      IF(LG.EQ.0) GO TO 85
      LTA = FLD(22,14,NSQ1(J1))
      INCLUDE VARG,LIST
C     CHECK FOR RADIATION CONDUCTOR
      IF(FLD(2,1,NSQ1(J1)).EQ.0) GO TO 3000
      NTYPE = FLD(0,5,NSQ2(J2))                                       *NEW
      LA = FLD(5,17,NSQ2(J2))                                         *NEW
      LK = FLD(22,14,NSQ2(J2))                                        *NEW
      GOTO(2005,2010,2015,2020,2025,2030,2035,2040,2045,2050,2055,    **-3
     $     2060,2065), NTYPE
 2005 TM = (T(I)+T(LTA))/2.0
 2007 CALL D1D1WM(TM,A(LA),XK(LK),G(LG))
      GO TO 2999
 2010 TM = T(I)
      GO TO 2007
 2015 CALL D1D1WM(T(I),A(LA),XK(LK),G1)
 2017 J2 = J2+1
      LA = FLD(5,17,NSQ2(J2))                                         *NEW
      LK = FLD(22,14,NSQ2(J2))                                        *NEW
      CALL D1D1WM(T(LTA),A(LA),XK(LK),G2)                             **-2
      GO TO 2998
 2020 G1 = XK(LK)*XK(LA)
      GO TO 2017
 2025 CALL D1D1WM(T(I),A(LA),XK(LK),G1)
      J2 = J2+1
      LA = FLD(5,17,NSQ2(J2))                                         *NEW
      LK = FLD(22,14,NSQ2(J2))                                        *NEW
      G2 = XK(LK)*XK(LA)                                              **-2
      GO TO 2998
 2030 TM = (T(I)+T(LTA))/2.0
 2032 CALL PLYAWM(A(LA),TM,A(LA+1),XK(LK),G(LG))
      GO TO 2999
 2035 TM = T(I)
      GO TO 2032
 2040 CALL PLYAWM(A(LA),T(I),A(LA+1),XK(LK),G1)
 2042 J2 = J2+1
      LA = FLD(5,17,NSQ2(J2))                                         *NEW
      LK = FLD(22,14,NSQ2(J2))                                        *NEW
      CALL PLYAWM(A(LA),T(LTA),A(LA+1),XK(LK),G2)                     **-2
      GO TO 2998
 2045 G1 = XK(LK)*XK(LA)
      GO TO 2042
 2050 CALL PLYAWM(A(LA),T(I),A(LA+1),XK(LK),G1)
      J2 = J2+1
      LA = FLD(5,17,NSQ2(J2))                                         *NEW
      LK = FLD(22,14,NSQ2(J2))                                        *NEW
      G2 = XK(LK)*XK(LA)                                              **-2
      GO TO 2998
 2055 TM = (T(I)+T(LTA))/2.0
      CALL D2D1WM(TM,CON(14),A(LA),XK(LK),G(LG))
      GO TO 2999
 2060 TM = T(LTA)
      GO TO 2007
 2065 TM = T(LTA)
      GO TO 2032
 2998 G(LG) = 1./(1./G1+1./G2)
      IF(FLD(3,1,NSQ1(J1)).EQ.1) G(LG) = G1*G2
 2999 J2 = J2+1
 3000 CONTINUE
```

CNEXPN,CNEXPN

```
00457            END
00460            IF(FLD(3,1,NSQ1(J1)).EQ.0) GO TO 55
00462            T1 = T(I)+460.0
00463            T2 = T(LTA)+460.0
00464            GV = G(LG)*(T1*T1+T2*T2)*(T1+T2)
00465            GO TO 60
00466         55 GV = G(LG)
00466     C      OBTAIN THE Q RATE THRU THE CONDUCTOR
00467         60 Q(I) = Q(I)+GV*T(LTA)
00467     C      SAVE SUMMATION OF CONDUCTORS
00470            X(LE) = X(LE)+GV
00470     C      CHECK FOR ADJOINING DIFFUSION NODE
00471            IF(LTA.GT.NND.OR.FLD(21,1,NSQ1(J1)).EQ.1) GO TO 65
00471     C      SAVE SUMMATION OF CONDUCTORS FOR ADJOINING NODE
00473            LEA = IE+LTA
00474            X(LEA) = X(LEA)+GV
00475            Q(LTA) = Q(LTA)+GV*T(I)
00475     C      CHECK FOR LAST CONDUCTOR
00476         65 IF(NSQ1(J1).GT.0) GO TO 50
00500         85 CONTINUE
00500     C      OBTAIN NEW DIFFUSION TEMPERATURES, DTMPCC AND CSGMIN
00502            DO 100 I = 1,NND
00505            LE = IE+I
00505     C      CALCULATE C/SK MINIMUM
00506            T1 = C(I)/X(LE)
00507            IF(T1.GE.CKM) GO TO 90
00511            CKM = T1
00512            KON(35) = I
00512     C      COMPUTE NEW TEMPERATURES USING CALCULATED SOURCE TERMS
00513         90 T2 = 1.0/EXP(TSTEP*X(LE)/C(I))
00514            T1 = (1.0-T2)*Q(I)/X(LE)+T2*T(I)
00514     C      CALCULATE THE ABSOLUTE VALUE TEMPERATURE CHANGE
00515            T2 = ABS(T1-T(I))
00515     C      SAVE THE LARGEST TEMPERATURE CHANGE
00516            IF(TCGM.GE.T2) GO TO 95
00520            TCGM = T2
00521            KON(36) = I
00521     C      STORE THE TEMPERATURES
00522         95 X(LE) = T(I)
00523            T(I) = T1
00524        100 CONTINUE
00526            CON(17) = CKM
00527            DELTA = CKM*CON(4)
00530            IF(CKM.LE.0.0) GO TO 996
00530     C      CHECK FOR FIRST PASS
00532            IF(PASS.GT.0.0) GO TO 115
00532     C      UNDO THE TEMPERATURE CALCULATIONS
00534        105 DO 110 I = 1,NNC
00537            LE = IE+I
00540            T(I) = X(LE)
00541        110 CONTINUE
00543            IF(PASS.GT.0.0) GO TO 15
00545            PASS = 1.0
00546            CON(1) = TPRINT
00547            CON(2) = 0.0
00550            TSTEP = DELTA*0.95
00551            GO TO 195
00551     C      IS THE TIME STEP USED LESS THAN THE TIME STEP CALCULATED
00552        115 IF(TSTEP.LE.DELTA) GO TO 130
00552     C      COMPUTE THE TIME STEP
```

CNEXPN.CNEXPN

```
00554  143*          TSTEP = DELTA*0.95
00555  144*          GO TO 105
00556  145*      120 TSTEP = 0.95*TSTEP*CON(6)/TCGM
00557  146*          GO TO 105
00560  147*      125 TSTEP = 0.95*TSTEP*CON(11)/TCGM
00561  148*          GO TO 105
00561  149*    C     SEE IF THE TEMPERATURE CHANGE WAS TOO LARGE
00562  150*      130 IF(TCGM.GT.CON(6)) GO TO 120
00562  151*    C     STORE THE MAXIMUM DIFFUSION TEMPERATURE CHANGE
00564  152*          CON(15) = TCGM
00564  153*    C     CHECK TO SEE IF THERE ARE ANY ARITHMETIC NODES
00565  154*          IF(NNA.LE.0) GO TO 185
00565  155*    C     COMPUTE ARITHMETIC TEMPERATURES BY SUCCESSIVE POINT OVER-RELAX
00567  156*          DN = CON(9)
00570  157*          DD = 1.0-DN
00571  158*          LAX = KON(5)
00572  159*          DO 170 I = 1,LAX
00575  160*          JJ1 = J1
00576  161*          JJ2 = J2
00577  162*          TCGM = 0.0
00600  163*          KON(20) = I
00601  164*          DO 165 L = L1,NNC
00604  165*          SUMC = 0.0
00605  166*          SUMCV = 0.0
00606  167*          IF(I.GT.1) GO TO 6000
00610  168*          INCLUDE VRQ2,LIST
00611  168*          IF(FLD(4,1,NSQ1(JJ1+1)).EQ.0) GO TO 6000
00613  168*          NTYPE = FLD(0,5,NSQ2(JJ2))                                      *NEW
00614  168*          LA = FLD(5,17,NSQ2(JJ2))                                        *NEW
00615  168*          LK = FLD(22,14,NSQ2(JJ2))                                       *NEW
00616  168*          GO TO (5005,5010,5015,5020,5025,5030,5035,5040,5030), NTYPE     **-3
00617  168*     5005 Q(L) = XK(LK)+Q(L)
00620  168*          GO TO 5999
00621  168*     5010 Q1 = 0.0
00622  168*     5012 CALL D1D1WM(T(L),A(LA),XK(LK),Q2)
00623  168*          GO TO 5998
00624  168*     5015 Q1 = 0.0
00625  168*     5017 CALL D1D1WM(CON(14),A(LA),XK(LK),Q2)
00626  168*          GO TO 5998
00627  168*     5020 CALL D1D1WM(CON(14),A(LA),XK(LK),Q1)
00630  168*     5022 JJ2 = JJ2+1
00631  168*          LA = FLD(5,17,NSQ2(JJ2))                                        *NEW
00632  168*          LK = FLD(22,14,NSQ2(JJ2))                                       *NEW
00633  168*          GO TO 5017                                                      **-2
00634  168*     5025 Q1 = XK(LK)*XK(LA)
00635  168*          GO TO 5022
00636  168*     5030 CALL D1D1WM(CON(14),A(LA),XK(LK),Q1)
00637  168*          JJ2 = JJ2+1
00640  168*          LA = FLD(5,17,NSQ2(JJ2))                                        *NEW
00641  168*          LK = FLD(22,14,NSQ2(JJ2))                                       *NEW
00642  168*          Q2 = XK(LK)*XK(LA)                                              **-2
00643  168*          GO TO 5998
00644  168*     5035 CALL D1D1WM(CON(14),A(LA),XK(LK),Q1)
00645  168*     5037 JJ2 = JJ2+1
00646  168*          LA = FLD(5,17,NSQ2(JJ2))                                        *NEW
00647  168*          LK = FLD(22,14,NSQ2(JJ2))                                       *NEW
00650  168*          GO TO 5012                                                      **-2
00651  168*     5040 Q1 = XK(LK)*XK(LA)
00652  168*          GO TO 5037
00653  168*     5998 Q(L) = Q1+Q2+Q(L)
```

CNEXPN.CNEXPN

```
168*        5999 JJ2 = JJ2+1
168*        6000 CONTINUE
168*             END
169*         135 JJ1 = JJ1+1
170*             LG = FLD(5,16,NSQ1(JJ1))
171*             LTA = FLD(22,14,NSQ1(JJ1))
172*             IF(I.GT.1) GO TO 4000
173*             INCLUDE VRG2,LIST
174*      C      CHECK FOR RADIATION CONDUCTOR
174*             IF(FLD(2,1,NSQ1(JJ1)).EQ.0) GO TO 4000
174*             NTYPE = FLD(0,5,NSQ2(JJ2))                                          *NEW
174*             LA = FLD(5,17,NSQ2(JJ2))                                            *NEW
174*             LK = FLD(22,14,NSQ2(JJ2))                                           *NEW
174*             GOTO(3005,3010,3015,3020,3025,3030,3035,3040,3045,3050,3055,        **-3
174*           $      3060,3065), NTYPE
174*        3005 TM = (T(L)+T(LTA))/2.0
174*        3007 CALL D1D1WM(TM,A(LA),XK(LK),G(LG))
174*             GO TO 3999
174*        3010 TM = T(L)
174*             GO TO 3007
174*        3015 CALL D1D1WM(T(L),A(LA),XK(LK),G1)
174*        3017 JJ2 = JJ2+1
174*             LA = FLD(5,17,NSQ2(JJ2))                                            *NEW
174*             LK = FLD(22,14,NSQ2(JJ2))                                           *NEW
174*             CALL D1D1WM(T(LTA),A(LA),XK(LK),G2)                                 **-2
174*             GO TO 3998
174*        3020 G1 = XK(LK)*XK(LA)
174*             GO TO 3017
174*        3025 CALL D1D1WM(T(L),A(LA),XK(LK),G1)
174*             JJ2 = JJ2+1
174*             LA = FLD(5,17,NSQ2(JJ2))                                            *NEW
174*             LK = FLD(22,14,NSQ2(JJ2))                                           *NEW
174*             G2 = XK(LK)*XK(LA)                                                  **-2
174*             GO TO 3998
174*        3030 TM = (T(L)+T(LTA))/2.0
174*        3032 CALL PLYAWM(A(LA),TM,A(LA+1),XK(LK),G(LG))
174*             GO TO 3999
174*        3035 TM = T(L)
174*             GO TO 3032
174*        3040 CALL PLYAWM(A(LA),T(L),A(LA+1),XK(LK),G1)
174*        3042 JJ2 = JJ2+1
174*             LA = FLD(5,17,NSQ2(JJ2))                                            *NEW
174*             LK = FLD(22,14,NSQ2(JJ2))                                           *NEW
174*             CALL PLYAWM(A(LA),T(LTA),A(LA+1),XK(LK),G2)                         **-2
174*             GO TO 3998
174*        3045 G1 = XK(LK)*XK(LA)
174*             GO TO 3042
174*        3050 CALL PLYAWM(A(LA),T(L),A(LA+1),XK(LK),G1)
174*             JJ2 = JJ2+1
174*             LA = FLD(5,17,NSQ2(JJ2))                                            *NEW
174*             LK = FLD(22,14,NSQ2(JJ2))                                           *NEW
174*             G2 = XK(LK)*XK(LA)                                                  **-2
174*             GO TO 3998
174*        3055 TM = (T(L)+T(LTA))/2.0
174*             CALL D2D1WM(TM,CON(14),A(LA),XK(LK),G(LG))
174*             GO TO 3999
174*        3060 TM = T(LTA)
174*             GO TO 3007
174*        3065 TM = T(LTA)
174*             GO TO 3032
```

/ CNEXPN.CNEXPN

```
00750      3998 G(LG) = 1./(1./G1+1./G2)
00751           IF(FLD(3,1,NSQ1(JJ1)).EQ.1) G(LG) = G1*G2
00753      3999 JJ2 = JJ2+1
00754      4000 CONTINUE
00755           END
00756           IF(FLD(3,1,NSQ1(JJ1)).EQ.0) GO TO 140
00760           T1 = T(L)+460.0
00761           T2 = T(LTA)+460.0
00762           GV = G(LG)*(T1*T1+T2*T2)*(T1+T2)
00763           GO TO 145
00764       140 GV = G(LG)
00765       145 SUMC = SUMC+GV
00766           SUMCV = SUMCV+GV*T(LTA)
00766     C     CHECK FOR LAST CONDUCTOR
00767           IF(NSQ1(JJ1).GT.0) GO TO 135
00771           T2 = DD*T(L)+DN*(SUMCV+Q(L))/SUMC
00771     C     OBTAIN THE CALCULATED TEMPERATURE DIFFERENCE
00772           T1 = ABS(T(L)-T2)
00772     C     STORE THE NEW TEMPERATURE
00773           T(L) = T2
00773     C     SAVE THE MAXIMUM ARITHMETIC RELAXATION CHANGE
00774           IF(TCGM.GE.T1) GO TO 165
00776           TCGM = T1
00777           KON(37) = L
01000       165 CONTINUE
01000     C     SEE IF RELAXATION CRITERIA WAS MET
01002           IF(TCGM.LE.CON(19)) GO TO 175
01004       170 CONTINUE
01004     C     STORE THE MAXIMUM ARITHMETIC RELAXATION CHANGE
01006       175 CON(30) = TCGM
01006     C     COMPUTE THE ARITHMETIC TEMPERATURE CHANGE
01007           TCGM = 0.0
01010           DO 180 I = L1,NNC
01013           LE = IE+I
01014           T1 = ABS(T(I)-X(LE))
01015           IF(T1.LT.TCGM) GO TO 180
01017           TCGM = T1
01020           KON(38) = I
01021       180 CONTINUE
01021     C     SEE IF ATMPCA WAS SATISFIED
01023           IF(TCGM.GT.CON(11)) GO TO 125
01025           CON(16) = TCGM
01026       185 KON(12) = 0
01027           CALL VARBL2
01027     C     CHECK THE BACKUP SWITCH
01030           IF(KON(12).NE.0) GO TO 105
01030     C     ADVANCE TIME
01032           CON(13) = CON(1)
01033           TSUM = TSUM+TSTEP
01034           TSTEP = DELTA*0.95
01034     C     CHECK FOR TIME TO PRINT
01035           IF(TSUM.GE.CON(18)) GO TO 190
01035     C     CHECK FOR PRINT EVERY ITERATION
01037           IF(KON(7).EQ.0) GO TO 10
01041           CALL OUTCAL
01042           GO TO 10
01042     C     TRY TO EVEN THE OUTPUT INTERVALS
01043       190 TPRINT = TPRINT+TSUM
01044       195 CALL OUTCAL
01044     C     IS TIME GREATER THAN END COMPUTE TIME
```

```
                          CNEXPN.CNEXPN

01045  230*         IF(CON(1)*1.000001.LT.CON(3)) GO TO 5
01047  231*         NTH = IE
01050  232*         NDIM = NLA
01051  233*         RETURN
01052  234*     995 WRITE(6,885)
01054  235*         GO TO 1000
01055  236*     996 WRITE(6,886)
01057  237*         GO TO 1000
01060  238*     997 WRITE(6,887)
01062  239*         GO TO 1000
01063  240*     998 WRITE(6,888) I
01066  241*         GO TO 1000
01067  242*     999 WRITE(6,889)
01071  243*    1000 CALL OUTCAL
01072  244*         CALL EXIT
01073  245*     885 FORMAT(46H CNEXPN REQUIRES SHORT PSEUDO-COMPUTE SEQUENCE)
01074  246*     886 FORMAT(24H CSGMIN ZERO OR NEGATIVE)
01075  247*     887 FORMAT(20H TIME STEP TOO SMALL)
01076  248*     888 FORMAT(I8,20H LOCATIONS AVAILABLE)
01077  249*     889 FORMAT(19H NO OUTPUT INTERVAL)
01100  250*         END

     END OF UNIVAC 1108 FORTRAN V COMPILATION.     0 *DIAGNOSTIC* MESSAGE(S)
   CNEXPN           SYMBOLIC
   CNEXPN   CODE    RELOCATABLE
```

```
QI   FOR,*   CNDUFR                                                          18 FEB 71
UNIVAC 1108 FORTRAN V ATHENA VERSION 131K-10D   CREATED ON 20 AUG 70
THIS COMPILATION WAS DONE ON 09 JUN 70 AT 23:15:00

   SUBROUTINE CNDUFR    ENTRY POINT 003620

   STORAGE USED (BLOCK, NAME, LENGTH)

         0001   *CODE         003635
         0000   *CONST+TEMP   000100
         0002   *SIMPLE VAR   000054
         0004   *ARRAYS       000000
         0005   *BLANK        000000
         0006   TITLE   000001
         0007   TEMP    000001
         0010   CAP     000001
         0011   SOURCE  000001
         0012   COND    000001
         0013   PC1     000001
         0014   PC2     000001
         0015   KONST   000001
         0016   ARRAY   000001
         0017   FIXCON  000001
         0020   XSPACE  000003
         0021   DIMENS  000010

   EXTERNAL REFERENCES (BLOCK, NAME)

         0022   VARBL1
         0023   D1D1WM
         0024   PLYAWM
         0025   D2D1WM
         0026   VARBL2
         0027   OUTCAL
         0030   EXIT
         0031   NERR2$
         0032   NWDU$
         0033   NIO2$
         0034   NER10$

   STORAGE ASSIGNMENT FOR VARIABLES (BLOCK, TYPE, RELATIVE LOCATION, NAME)

    0016 R 000000 A        0010 R 000000 C        0002 R 000017 CKM      0017 R 000000 CON      0002 R 000023 C1
    0002 R 000024 C2       0002 R 000045 DD       0002 R 000043 DELTA    0002 R 000044 DN       0002 R 000040 DT1
    0002 R 000041 DT2      0012 R 000000 G        0002 R 000036 GV       0002 R 000032 G1       0002 R 000033 G2
    0006 R 000000 H        0002 I 000005 I        0002 I 000002 IE       0002 I 000003 IEM      0002 I 000047 JJ1
    0002 I 000050 JJ2      0002 I 000014 J1       0002 I 000015 J2       0015 I 000000 K        0017 I 000000 KON
    0002 I 000051 L        0002 I 000021 LA       0002 I 000046 LAX      0002 I 000013 LE       0002 I 000037 LEA
    0002 I 000042 LEH      0002 I 000027 LG       0002 I 000022 LK       0021 I 000006 LSQ1     0021 I 000007 LSQ2
    0002 I 000030 LTA      0002 I 000006 L1       0021 I 000005 NAT      0021 I 000004 NCT      0020 I 000000 NDIM
    0021 I 000003 NGT      0002 I 000004 NLA      0021 I 000001 NNA      0002 I 000001 NNC      0021 I 000000 NND
    0021 I 000002 NNT      0013 I 000000 NSQ1     0014 I 000000 NSQ2     0020 I 000001 NTH      0002 I 000020 NTYPE
    0020   000002 NX       0002 R 000000 PASS     0011 R 000000 Q        0002 R 000025 Q1       0002 R 000026 Q2
    0002 R 000052 SUMC     0002 R 000053 SUMCV    0007 R 000000 T        0002 R 000016 TCGM     0002 R 000031 T"
    0002 R 000011 TPRINT   0002 R 000007 TSTEP    0002 R 000010 TSTEPO   0002 R 000012 TSUM     0002 R 000034 T1
```

CNDUFR

```
0002 R 000035 T2       0020 R 000002 X        0015 R 000000 XK       0001   000132 10L       0001   003574 1000L
0001   000344 1005L    0001   000365 1010L    0001   000403 1012L    0001   000436 1015L     0001   000444 1020L
0001   003407 1023G    0001   000504 1025L    0001   000530 1030L    0001   000551 1032L     0001   000607 1035L
0001   000615 1040L    0001   000660 1045L    0001   002125 105L     0001   003462 1050G     0001   002174 115L
0001   002205 120L     0001   002213 125L     0001   002221 130L     0001   002551 135L      0001   003317 140L
0001   003322 145L     0001   000141 15L      0001   003367 165L     0001   003400 175L      0001   003431 180L
0001   003442 185L     0001   003514 190L     0001   003517 195L     0001   000702 1998L     0001   000705 1999L
0001   000710 2000L    0001   001250 2005L    0001   001255 2007L    0001   001275 2010L     0001   001300 2015L
0001   001316 2017L    0001   000213 2026     0001   001352 2020L    0001   001360 2025L     0001   001420 2030L
0001   001425 2032L    0001   001450 2035L    0001   001453 2040L    0001   001474 2042L     0001   001533 2045L
0001   001541 2050L    0001   001604 2055L    0001   001632 2060L    0001   001636 2065L     0001   000232 214G
0001   000273 234G     0001   000153 25L      0001   001642 2998L    0001   001670 2999L     0001   000166 30L
0001   001673 3000L    0001   002637 3005L    0001   002644 3007L    0001   002664 3010L     0001   002667 3015L
0001   002705 3017L    0001   002741 3020L    0001   002747 3025L    0001   003007 3030L     0001   003014 3032L
0001   003037 3035L    0001   003042 3040L    0001   003063 3042L    0001   003122 3045L     0001   003130 3050L
0001   003173 3055L    0001   003221 3060L    0001   003225 3065L    0001   003231 3998L     0001   003257 3999L
0001   003262 4000L    0001   000753 4005L    0001   000760 4010L    0001   000761 4012L     0001   001000 4015L
0001   001001 4017L    0001   001016 4020L    0001   001032 4022L    0001   001047 4025L     0001   001055 4030L
0001   001113 4035L    0001   001127 4037L    0001   001144 4040L    0001   000244 45L       0001   001152 4998L
0001   001156 4999L    0001   000120 5L       0001   001161 50L      0001   001161 5000L     0001   002343 5005L
0001   002350 5010L    0001   002351 5012L    0001   002370 5015L    0001   002371 5017L     0001   002406 5020L
0001   002422 5022L    0001   002437 5025L    0001   002445 5030L    0001   002503 5035L     0001   002517 5037L
0001   002534 5040L    0001   002020 507G     0001   002131 541G     0001   001724 55L       0001   002162 556G
0001   002542 5998L    0001   002546 5999L    0001   001727 60L      0001   002551 6000L     0001   002250 605G
0001   002271 614G     0001   001775 65L      0000   000000 885F     0000   000011 886F      0000   000016 887F
0000   000023 888F     0000   000030 889F     0001   002044 90L      0001   002101 95L       0001   003536 995L
0001   003544 996L     0001   003552 997L     0001   003560 998L     0001   003567 999L
```

```
00101    1*        SUBROUTINE CNDUFR
00101    2*   C    EXPLICIT DUFORT-FRANKEL EXECUTION SUBROUTINE FOR SINDA F-V
00101    3*   C    THE SHORT PSEUDO-COMPUTE SEQUENCE IS REQUIRED
00103    4*        INCLUDE COMM,LIST
00104    4*        COMMON /TITLE/H(1) /TEMP/T(1) /CAP/C(1) /SOURCE/Q(1) /COND/G(1)
00105    4*        COMMON /PC1/NSQ1(1) /PC2/NSQ2(1) /KONST/K(1) /ARRAY/A(1)
00106    4*        COMMON /FIXCON/KON(1) /XSPACE/NDIM,NTH,X(1)
00107    4*        COMMON /DIMENS/ NND,NNA,NNT,NGT,NCT,NAT,LSQ1,LSQ2
00110    4*        DIMENSION CON(1),XK(1),NX(1)
00111    4*        EQUIVALENCE (KON(1),CON(1)),(K(1),XK(1)),(X(1),NX(1))
00112    4*        END
00113    5*        INCLUDE DEFF,LIST
00113    5*   C**************  CONTROL CONSTANT DEFINITIONS AND NAMES  ******.*********
00113    5*   C    CONTROL CONSTANT 1 CONTAINS THE NEW PROBLEM TIME            (TIMEN)
00113    5*   C    CONTROL CONSTANT 2 CONTAINS THE TIME STEP USED             (DTIMEU)
00113    5*   C    CONTROL CONSTANT 3 CONTAINS THE PROBLEM STOP TIME          (TIMEND)
00113    5*   C    CONTROL CONSTANT 4 CONTAINS THE TIME STEP FACTOR,EXPLICIT (CSGFAC)
00113    5*   C    CC5 IS THE INPUT NUMBER OF ITERATION DO LOOPS, INTEGER     (NLOOP)
00113    5*   C    CC6 CONTAINS THE DIFFUSION TEMPERATURE CHANGE ALLOWED     (DTMPCA)
00113    5*   C    CC7 CONTAINS THE OUTPUT EACH ITERATION SWITCH             (OPEITR)
00113    5*   C    CC8 CONTAINS THE MAXIMUM ALLOWED TIME STEP                (DTIMEH)
00113    5*   C    CC9 CONTAINS THE NEW ARITHMETIC TEMP. DAMPING FACTOR       (DAMPA)
00113    5*   C    CC10 CONTAINS THE NEW DIFFUSION TEMP. DAMPING FACTOR       (DAMPD)
00113    5*   C    CC11 CONTAINS THE MAXIMUM ALLOWED ARITHMETIC TEMP. CHANGE (ATMPCA)
00113    5*   C    CC12 CONTAINS THE BACKUP SWITCH CHECKED AFTER VARIABLES   (BACKUP)
00113    5*   C    CC13 CONTAINS THE PRESENT TIME OR PROBLEM START TIME       (TIMEO)
00113    5*   C    CC14 CONTAINS THE MEAN TIME BETWEEN AN ITERATION           (TIMEM)
00113    5*   C    CC15 CONTAINS THE DIFFUSION TEMPERATURE CHANGE CALCULATED (DTMPCC)
00113    5*   C    CC16 CONTAINS ARITHMETIC TEMPERATURE CHANGF CALCULATED    (ATMPCC)
```

```
C     CONTROL CONSTANT 17 IS RESERVED FOR THE C/SG MINIMUM  .   (CSGMIN)
C     CONTROL CONSTANT 18 CONTAINS THE OUTPUT INTERVAL          (OUTPUT)
C     CC19 CONTAINS THE ARITHMETIC RELAXATION CRITERIA ALLOWED  (ARLXCA)
C     CC20 CONTAINS THE NUMBER OF RELAXATION LOOPS USED,INTEGER (LOOPCT)
C     CC21 CONTAINS THE MINIMUM ALLOWED TIME STEP               (DTIMEL)
C     CC22 IS FOR THE INPUT TIME STEP IMPLICIT                  (DTIMEI)
C     CC23 CONTAINS THE C/SG MAXIMUM                            (CSGMAX)
C     CC24 CONTAINS THE C/SG RANGE ALLOWED                      (CSGRAL)
C     CC25 CONTAINS THE C/SG RANGE CALCULATED                   (CSGRCL)
C     CC26 CONTAINS THE DIFFUSION RELAXATION CRITERIA ALLOWED   (DRLXCA)
C     CC27 CONTAINS THE DIFFUSION RELAXATION CHANGE CALCULATED  (DRLXCC)
C     CC28 CONTAINS THE LINE COUNTER, INTEGER                   (LINECT)
C     CC29 CONTAINS THE PAGE COUNTER, INTEGER                   (PAGECT)
C     CC30 CONTAINS ARITHMETIC RELAXATION CHANGE CALCULATED     (ARLXCC)
C     CC31 IS INDICATOR, 0=THERMAL SPCS,1=THERMAL LPCS,2=GENERAL (LSPCS)
C     CC32 CONTAINS THE ENERGY BALANCE OF THE SYSTEM, IN - OUT  (ENGBAL)
C     CC33 CONTAINS THE DESIRED ENERGY BALANCE, USER INPUT      (BALENG)
C     CC34 CONTAINS THE NOCOPY SWITCH FOR MATRIX USERS          (NOCOPY)
C     CC35 CONTAINS RELATIVE NODE NUMBER OF CSGMIN
C     CC36 CONTAINS RELATIVE NODE NUMBER OF DTMPCC
C     CC37 CONTAINS RELATIVE NODE NUMBER OF ARLXCC
C     CC38 CONTAINS RELATIVE NODE NUMBER OF ATMPCC
C     CC39-40-41-42-43 CONTAIN DUMMY INTEGER CONSTANTS    (I-J-K-L-MTEST)
C     CC44-45-46-47-48 CONTAIN DUMMY FLOATING CONSTANTS   (R-S-T-U-VTEST)
C     CC49 IS THE QUASI-LINEARIZATION INTERVAL FOR CINDSM       (LAXFAC)
C     CC50 IS NOT USED AT PRESENT
      END
      IF(CON(4).LT.1.0) CON(4) = 1.0
      IF(KON(5).LE.0) KON(5) = 1
      IF(CON(6).LE.0.) CON(6) = 1.E+8
      IF(CON(8).LE.0.) CON(8) = 1.E+8
      IF(CON(9).LE.0.) CON(9) = 1.0
      IF(CON(11).LE.0.) CON(11) = 1.E+8
      IF(CON(18).LE.0.) GO TO 999
      IF(CON(19).LE.0.) CON(19) = 1.E+8
      IF(KON(31).NE.0) GO TO 995
      PASS = -1.0
      NNC = NND+NNA
      IE = NTH
      IEH = NTH+NNC
      NLA = NDIM
      NTH = NTH+NNC+NND
      NDIM = NDIM-NNC-NND
C     CHECK FOR EXTRA LOCATIONS FOR CALCULATED NODES
      I = NLA-NNC-NND
      IF(I.LT.0) GO TO 998
      L1 = NND+1
      TSTEP = CON(18)
      TSTEPO = 0.0
      TPRINT = CON(13)
C     INITALIZE TIME SUM BETWEEN OUTPUT INTERVALS
    5 TSUM = 0.0
C     DOES OLD TIME PLUS THE OUTPUT INTERVAL EXCEED THE STOP TIME
      IF(CON(13)+CON(18).LE.CON(3)) GO TO 10
C     DONT EXCEED IT
      CON(18) = CON(3)-CON(13)
C     IS THE TIME STEP LARGER THAN ALLOWED
   10 IF(TSTEP.LE.CON(8)) GO TO 15
      TSTEP = CON(8)
C     DOES THE TIME SUM PLUS THE TIME STEP EXCEED OUTPUT INTERVAL
```

CNDUFR

```
00164   39*        15 IF(TSUM+TSTEP-CON(18)) 25,30,20
00164   40*   C       DONT EXCEED IT
00167   41*        20 TSTEP = CON(18)-TSUM
00170   42*           GO TO 30
00170   43*   C       DOES TIME SUM PLUS TWO TIME STEPS EXCEED OUTPUT INTERVAL
00171   44*        25 IF(TSUM+2.0*TSTEP.LE.CON(18)) GO TO 30
00171   45*   C       APPROACH THE OUTPUT INTERVAL GRADUALLY
00173   46*           TSTEP = (CON(18)-TSUM)/2.0
00173   47*   C       STORE DELTA TIME STEP IN THE CONSTANTS
00174   48*        30 CON(2) = TSTEP
00174   49*   C       IS THE TIME STEP USED LESS THAN THE TIME STEP ALLOWED
00175   50*           IF(TSTEP.LT.CON(21)) GO TO 997
00175   51*   C       CALCULATE THE NEW TIME
00177   52*           CON(1) = TPRINT+TSUM+TSTEP
00177   53*   C       COMPUTE THE MEAN TIME BETWEEN ITERATIONS
00200   54*           CON(14) = (CON(1)+CON(13))/2.0
00200   55*   C       ZERO OUT ALL SOURCE LOCATIONS AND EXTRA LOCATIONS
00201   56*           DO 35 I = 1,NND
00204   57*           LE = IE+I
00205   58*           X(LE) = 0.0
00206   59*           Q(I) = 0.0
00207   60*        35 CONTINUE
00207   61*   C       SHIFT THE ARITHMETIC TEMPERATURES INTO THE EXTRA LOCATIONS
00211   62*           IF(NNA.LE.0) GO TO 45
00213   63*           DO 40 I = L1,NNC
00216   64*           Q(I) = 0.0
00217   65*           LE = IE+I
00220   66*           X(LE) = T(I)
00221   67*        40 CONTINUE
00223   68*        45 KON(12) = 0
00224   69*           CALL VARBL1
00225   70*           IF(KON(12).NE.0) GO TO 10
00227   71*           J1 = 0
00230   72*           J2 = 1
00231   73*           TCGM = 0.0
00232   74*           CKM = 1.E+10
00232   75*   C       CALCULATE Q SUM AND G SUM
00233   76*           DO 85 I = 1,NND
00236   77*           LE = IE+I
00237   78*           INCLUDE VARC,LIST
00240   78*           IF(FLD(1,1,NSQ1(J1+1)).EQ.0) GO TO 2000
00242   78*           NTYPE = FLD(0,5,NSQ2(J2))                                  *NEW
00243   78*           LA = FLD(5,17,NSQ2(J2))                                    *NEW
00244   78*           LK = FLD(22,14,NSQ2(J2))                                   *NEW
00245   78*           GO TO (1005,1010,1015,1020,1025,1030,1035,1040,1045), NTYPE **-3
00246   78*      1005 CALL D1D1WM(T(I),A(LA),XK(LK),C(I))
00247   78*           GO TO 1999
00250   78*      1010 CALL D1D1WM(T(I),A(LA),XK(LK),C1)
00251   78*      1012 J2 = J2+1
00252   78*           LA = FLD(5,17,NSQ2(J2))                                    *NEW
00253   78*           LK = FLD(22,14,NSQ2(J2))                                   *NEW
00254   78*           CALL D1D1WM(T(I),A(LA),XK(LK),C2)                          **-2
00255   78*           GO TO 1998
00256   78*      1015 C1 = XK(LK)*XK(LA)
00257   78*           GO TO 1012
00260   78*      1020 CALL D1D1WM(T(I),A(LA),XK(LK),C1)
00261   78*           J2 = J2+1
00262   78*           LA = FLD(5,17,NSQ2(J2))                                    *NEW
00263   78*           LK = FLD(22,14,NSQ2(J2))                                   *NEW
00264   78*           C2 = XK(LK)*XK(LA)                                         **-2
```

CNDUFR

```
78*        GO TO 1998
78*   1025 CALL PLYAWM(A(LA),T(I),A(LA+1),XK(LK),C(I))
78*        GO TO 1999
78*   1030 CALL PLYAWM(A(LA),T(I),A(LA+1),XK(LK),C1)
78*   1032 J2 = J2+1
78*        LA = FLD(5,17,NSQ2(J2))                                      *NEW
78*        LK = FLD(22,14,NSQ2(J2))                                     *NEW
78*        CALL PLYAWM(A(LA),T(I),A(LA+1),XK(LK),C2)                    **-2
78*        GO TO 1998
78*   1035 C1 = XK(LK)*XK(LA)
78*        GO TO 1032
78*   1040 CALL PLYAWM(A(LA),T(I),A(LA+1),XK(LK),C1)
78*        J2 = J2+1
78*        LA = FLD(5,17,NSQ2(J2))                                      *NEW
78*        LK = FLD(22,14,NSQ2(J2))                                     *NEW
78*        C2 = XK(LK)*XK(LA)                                           **-2
78*        GO TO 1998
78*   1045 CALL D2D1WM(T(I),CON(14),A(LA),XK(LK),C(I))
78*        GO TO 1999
78*   1998 C(I) = C1+C2
78*   1999 J2 = J2+1
78*   2000 CONTINUE
78*        END
79*        INCLUDE VARQ,LIST
79*        IF(FLD(4,1,NSQ1(J1+1)).EQ.0) GO TO 5000
79*        NTYPE = FLD(0,5,NSQ2(J2))                                    *NEW
79*        LA = FLD(5,17,NSQ2(J2))                                      *NEW
79*        LK = FLD(22,14,NSQ2(J2))                                     *NEW
79*        GO TO (4005,4010,4015,4020,4025,4030,4035,4040,4030), NTYPE  **-3
79*   4005 Q(I) = XK(LK)+Q(I)
79*        GO TO 4999
79*   4010 Q1 = 0.0
79*   4012 CALL D1D1WM(T(I),A(LA),XK(LK),Q2)
79*        GO TO 4998
79*   4015 Q1 = 0.0
79*   4017 CALL D1D1WM(CON(14),A(LA),XK(LK),Q2)
79*        GO TO 4998
79*   4020 CALL D1D1WM(CON(14),A(LA),XK(LK),Q1)
79*   4022 J2 = J2+1
79*        LA = FLD(5,17,NSQ2(J2))                                      *NEW
79*        LK = FLD(22,14,NSQ2(J2))                                     *NEW
79*        GO TO 4017                                                   **-2
79*   4025 Q1 = XK(LK)*XK(LA)
79*        GO TO 4022
79*   4030 CALL D1D1WM(CON(14),A(LA),XK(LK),Q1)
79*        J2 = J2+1
79*        LA = FLD(5,17,NSQ2(J2))                                      *NEW
79*        LK = FLD(22,14,NSQ2(J2))                                     *NEW
79*        Q2 = XK(LK)*XK(LA)                                           **-2
79*        GO TO 4998
79*   4035 CALL D1D1WM(CON(14),A(LA),XK(LK),Q1)
79*   4037 J2 = J2+1
79*        LA = FLD(5,17,NSQ2(J2))                                      *NEW
79*        LK = FLD(22,14,NSQ2(J2))                                     *NEW
79*        GO TO 4012                                                   **-2
79*   4040 Q1 = XK(LK)*XK(LA)
79*        GO TO 4037
79*   4998 Q(I) = Q1+Q2+Q(I)
79*   4999 J2 = J2+1
79*   5000 CONTINUE
```

```
79*          END
80*       50 J1 = J1+1
81*          LG = FLD(5,16,NSQ1(J1))
82*          IF(LG.EQ.0) GO TO 65
83*          LTA = FLD(22,14,NSQ1(J1))
84*          INCLUDE VARG,LIST
85*     C    CHECK FOR PADIATION CONDUCTOR
85*          IF(FLD(2,1,NSQ1(J1)).EQ.0) GO TO 3000
85*          NTYPE = FLD(0,5,NSQ2(J2))                                        *NEW
85*          LA = FLD(5,17,NSQ2(J2))                                          *NEW
85*          LK = FLD(22,14,NSQ2(J2))                                         *NEW
85*          GOTO(2005,2010,2015,2020,2025,2030,2035,2040,2045,2050,2055,     **-3
85*         $     2060,2065), NTYPE
85*     2005 TM = (T(I)+T(LTA))/2.0
85*     2007 CALL D1D1WM(TM,A(LA),XK(LK),G(LG))
85*          GO TO 2999
85*     2010 TM = T(I)
85*          GO TO 2007
85*     2015 CALL D1D1WM(T(I),A(LA),XK(LK),G1)
85*     2017 J2 = J2+1
85*          LA = FLD(5,17,NSQ2(J2))                                          *NEW
85*          LK = FLD(22,14,NSQ2(J2))                                         *NEW
85*          CALL D1D1WM(T(LTA),A(LA),XK(LK),G2)                              **-2
85*          GO TO 2998
85*     2020 G1 = XK(LK)*XK(LA)
85*          GO TO 2017
85*     2025 CALL D1D1WM(T(I),A(LA),XK(LK),G1)
85*          J2 = J2+1
85*          LA = FLD(5,17,NSQ2(J2))                                          *NEW
85*          LK = FLD(22,14,NSQ2(J2))                                         *NEW
85*          G2 = XK(LK)*XK(LA)                                               **-2
85*          GO TO 2998
85*     2030 TM = (T(I)+T(LTA))/2.0
85*     2032 CALL PLYAWM(A(LA),TM,A(LA+1),XK(LK),G(LG))
85*          GO TO 2999
85*     2035 TM = T(I)
85*          GO TO 2032
85*     2040 CALL PLYAWM(A(LA),T(I),A(LA+1),XK(LK),G1)
85*     2042 J2 = J2+1
85*          LA = FLD(5,17,NSQ2(J2))                                          *NEW
85*          LK = FLD(22,14,NSQ2(J2))                                         *NEW
85*          CALL PLYAWM(A(LA),T(LTA),A(LA+1),XK(LK),G2)                      **-2
85*          GO TO 2998
85*     2045 G1 = XK(LK)*XK(LA)
85*          GO TO 2042
85*     2050 CALL PLYAWM(A(LA),T(I),A(LA+1),XK(LK),G1)
85*          J2 = J2+1
85*          LA = FLD(5,17,NSQ2(J2))                                          *NEW
85*          LK = FLD(22,14,NSQ2(J2))                                         *NEW
85*          G2 = XK(LK)*XK(LA)                                               **-2
85*          GO TO 2998
85*     2055 TM = (T(I)+T(LTA))/2.0
85*          CALL D2D1WM(TM,CON(14),A(LA),XK(LK),G(LG))
85*          GO TO 2999
85*     2060 TM = T(LTA)
85*          GO TO 2007
85*     2065 TM = T(LTA)
85*          GO TO 2032
85*     2998 G(LG) = 1./(1./G1+1./G2)
85*          IF(FLD(3,1,NSQ1(J1)).EQ.1) G(LG) = G1*G2
```

```
00457   2999 J2 = J2+1
00460   3000 CONTINUE
00461        END
00462        IF(FLD(3,1,NSQ1(J1)).EQ.0) GO TO 55
00464        T1 = T(I)+460.0
00465        T2 = T(LTA)+460.0
00466        GV = G(LG)*(T1*T1+T2*T2)*(T1+T2)
00467        GO TO 60
00470     55 GV = G(LG)
00470 C      OBTAIN THE Q PLUS SUMMATION G*TA TERM
00471     60 Q(I) = Q(I)+GV*T(LTA)
00471 C      SAVE SUMMATION OF CONDUCTORS
00472        X(LE) = X(LE)+GV
00472 C      CHECK FOR ADJOINING DIFFUSION NODE OR ONE-WAY CONDUCTOR
00473        IF(LTA.GT.NND.OR.FLD(21,1,NSQ1(J1)).EQ.1) GO TO 65
00473 C      SAVE SUMMATION OF CONDUCTORS FOR ADJOINING NODE
00475        LEA = IE+LTA
00476        X(LEA) = X(LEA)+GV
00477        Q(LTA) = Q(LTA)+GV*T(I)
00477 C      CHECK FOR LAST CONDUCTOR
00500     65 IF(NSQ1(J1).GT.0) GO TO 50
00502     85 CONTINUE
00504        DT1 = TSTEP/(TSTEP+TSTEPO)
00505        DT2 = TSTEPO/(TSTEP+TSTEPO)
00505 C      OBTAIN NEW DIFFUSION TEMPERATURES, DTMPCC AND CSGMIN
00506        DO 100 I = 1,NND
00511        LE = IE+I
00512        LEH = IEH+I
00512 C      CALCULATE C/SK MINIMUM
00513        T1 = C(I)/X(LE)
00514        IF(T1.GE.CKM) GO TO 90
00516        CKM = T1
00517        KON(35) = I
00517 C      COMPUTE NEW TEMPERATURES USING CALCULATED SOURCE TERMS
00520     90 T1=(DT1*X(LEH)*(C(I)/TSTEP-X(LE))+Q(I))/(C(I)*(1.-DT2)/TSTEP+DT2*
00520       $X(LE))
00520 C      CALCULATE THE ABSOLUTE VALUE TEMPERATURE CHANGE
00521        T2 = ABS(T(I)-T1)
00521 C      SAVE THE LARGEST TEMPERATURE CHANGE
00522        IF(TCGM.GE.T2) GO TO 95
00524        TCGM = T2
00525        KON(36) = I
00525 C      STORE THE TEMPERATURES
00526     95 X(LE) = T(I)
00527        T(I) = T1
00530    100 CONTINUE
00532        CON(17) = CKM
00533        DELTA = CKM*CON(4)
00534        IF(CKM.LE.0.0) GO TO 996
00534 C      CHECK FOR FIRST PASS
00536        IF(PASS.GT.0.0) GO TO 115
00536 C      UNDO THE TEMPERATURE CALCULATIONS
00540    105 DO 110 I = 1,NNC
00543        LE = IE+I
00544        T(I) = X(LE)
00545    110 CONTINUE
00547        IF(PASS.GT.0.0) GO TO 15
00551        PASS = 1.0
00552        CON(1) = TPRINT
00553        CON(2) = 0.0
```

```
      TSTEP = CKM*0.9
      DO 112 I = 1,NND
      LEH = IEH+I
  112 X(LEH) = T(I)
      GO TO 195
C     IS THE TIME STEP USED LESS THAN THE TIME STEP CALCULATED
  115 IF(TSTEP.LE.DELTA) GO TO 130
C     COMPUTE THE TIME STEP
      TSTEP = DELTA*0.95
      GO TO 105
  120 TSTEP = 0.95*TSTEP*CON(6)/TCGM
      GO TO 105
  125 TSTEP = 0.95*TSTEP*CON(11)/TCGM
      GO TO 105
C     SEE IF THE TEMPERATURE CHANGE WAS TOO LARGE
  130 IF(TCGM.GT.CON(6)) GO TO 120
C     STORE THE MAXIMUM DIFFUSION TEMPERATURE CHANGE
      CON(15) = TCGM
C     CHECK TO SEE IF THERE ARE ANY ARITHMETIC NODES
      IF(NNA.LE.0) GO TO 185
C     COMPUTE ARITHMETIC TEMPERATURES BY SUCCESSIVE POINT OVER-RELAX
      DN = CON(9)
      DD = 1.0-DN
      LAX = KON(5)
      DO 170 I = 1,LAX
      JJ1 = J1
      JJ2 = J2
      TCGM = 0.0
      KON(20) = I
      DO 165 L = L1,NNC
      SUMC = 0.0
      SUMCV = 0.0
      IF(I.GT.1) GO TO 6000
      INCLUDE VRQ2,LIST
      IF(FLD(4,1,NSQ1(JJ1+1)).EQ.0) GO TO 6000
      NTYPE = FLD(0,5,NSQ2(JJ2))                                        *NEW
      LA = FLD(5,17,NSQ2(JJ2))                                          *NEW
      LK = FLD(22,14,NSQ2(JJ2))                                         *NEW
      GO TO (5005,5010,5015,5020,5025,5030,5035,5040,5030), NTYPE       **-3
 5005 Q(L) = XK(LK)+Q(L)
      GO TO 5999
 5010 Q1 = 0.0
 5012 CALL D1D1WM(T(L),A(LA),XK(LK),Q2)
      GO TO 5998
 5015 Q1 = 0.0
 5017 CALL D1D1WM(CON(14),A(LA),XK(LK),Q2)
      GO TO 5998
 5020 CALL D1D1WM(CON(14),A(LA),XK(LK),Q1)
 5022 JJ2 = JJ2+1
      LA = FLD(5,17,NSQ2(JJ2))                                          *NEW
      LK = FLD(22,14,NSQ2(JJ2))                                         *NEW
      GO TO 5017                                                        **-2
 5025 Q1 = XK(LK)*XK(LA)
      GO TO 5022
 5030 CALL D1D1WM(CON(14),A(LA),XK(LK),Q1)
      JJ2 = JJ2+1
      LA = FLD(5,17,NSQ2(JJ2))                                          *NEW
      LK = FLD(22,14,NSQ2(JJ2))                                         *NEW
      Q2 = XK(LK)*XK(LA)                                                **-2
      GO TO 5998
```

```
176*     5035 CALL D1D1WM(CON(14),A(LA),XK(LK),Q1)
176*     5037 JJ2 = JJ2+1
176*          LA = FLD(5,17,NSQ2(JJ2))                                   *NEW
176*          LK = FLD(22,14,NSQ2(JJ2))                                  *NEW
176*          GO TO 5012                                                 **-2
176*     5040 Q1 = XK(LK)*XK(LA)
176*          GO TO 5037
176*     5998 Q(L) = Q1+Q2+Q(L)
176*     5999 JJ2 = JJ2+1
176*     6000 CONTINUE
176*          END
177*      135 JJ1 = JJ1+1
178*          LG = FLD(5,16,NSQ1(JJ1))
179*          LTA = FLD(22,14,NSQ1(JJ1))
180*          IF(I.GT.1) GO TO 4000
181*          INCLUDE VRG2,LIST
182*   C      CHECK FOR RADIATION CONDUCTOR
182*          IF(FLD(2,1,NSQ1(JJ1)).EQ.0) GO TO 4000
182*          NTYPE = FLD(0,5,NSQ2(JJ2))                                 *NEW
182*          LA = FLD(5,17,NSQ2(JJ2))                                   *NEW
182*          LK = FLD(22,14,NSQ2(JJ2))                                  *NEW
182*          GOTO(3005,3010,3015,3020,3025,3030,3035,3040,3045,3050,3055,  **-3
182*        $      3060,3065), NTYPE
182*     3005 TM = (T(L)+T(LTA))/2.0
182*     3007 CALL D1D1WM(TM,A(LA),XK(LK),G(LG))
182*          GO TO 3999
182*     3010 TM = T(L)
182*          GO TO 3007
182*     3015 CALL D1D1WM(T(L),A(LA),XK(LK),G1)
182*     3017 JJ2 = JJ2+1
182*          LA = FLD(5,17,NSQ2(JJ2))                                   *NEW
182*          LK = FLD(22,14,NSQ2(JJ2))                                  *NEW
182*          CALL D1D1WM(T(LTA),A(LA),XK(LK),G2)                        **-2
182*          GO TO 3998
182*     3020 G1 = XK(LK)*XK(LA)
182*          GO TO 3017
182*     3025 CALL D1D1WM(T(L),A(LA),XK(LK),G1)
182*          JJ2 = JJ2+1
182*          LA = FLD(5,17,NSQ2(JJ2))                                   *NEW
182*          LK = FLD(22,14,NSQ2(JJ2))                                  *NEW
182*          G2 = XK(LK)*XK(LA)                                         **-2
182*          GO TO 3998
182*     3030 TM = (T(L)+T(LTA))/2.0
182*     3032 CALL PLYAWM(A(LA),TM,A(LA+1),XK(LK),G(LG))
182*          GO TO 3999
182*     3035 TM = T(L)
182*          GO TO 3032
182*     3040 CALL PLYAWM(A(LA),T(L),A(LA+1),XK(LK),G1)
182*     3042 JJ2 = JJ2+1
182*          LA = FLD(5,17,NSQ2(JJ2))                                   *NEW
182*          LK = FLD(22,14,NSQ2(JJ2))                                  *NEW
182*          CALL PLYAWM(A(LA),T(LTA),A(LA+1),XK(LK),G2)                **-2
182*          GO TO 3998
182*     3045 G1 = XK(LK)*XK(LA)
182*          GO TO 3042
182*     3050 CALL PLYAWM(A(LA),T(L),A(LA+1),XK(LK),G1)
182*          JJ2 = JJ2+1
182*          LA = FLD(5,17,NSQ2(JJ2))                                   *NEW
182*          LK = FLD(22,14,NSQ2(JJ2))                                  *NEW
182*          G2 = XK(LK)*XK(LA)                                         **-2
```

CNDUFR

```
00752  182*           GO TO 3998
00753  182*      3055 TM = (T(L)+T(LTA))/2.0
00754  182*           CALL D2D1WM(TM,CON(14),A(LA),XK(LK),G(LG))
00755  182*           GO TO 3999
00756  182*      3060 TM = T(LTA)
00757  182*           GO TO 3007
00760  182*      3065 TM = T(LTA)
00761  182*           GO TO 3032
00762  182*      3998 G(LG) = 1./(1./G1+1./G2)
00763  182*           IF(FLD(3,1,NSQ1(JJ1)).EQ.1) G(LG) = G1*G2
00765  182*      3999 JJ2 = JJ2+1
00766  182*      4000 CONTINUE
00767  182*           END
00770  183*           IF(FLD(3,1,NSQ1(JJ1)).EQ.0) GO TO 140
00772  184*           T1 = T(L)+460.0
00773  185*           T2 = T(LTA)+460.0
00774  186*           GV = G(LG)*(T1*T1+T2*T2)*(T1+T2)
00775  187*           GO TO 145
00776  188*       140 GV = G(LG)
00777  189*       145 SUMC = SUMC+GV
01000  190*           SUMCV = SUMCV+GV*T(LTA)
01000  191*   C       CHECK FOR LAST CONDUCTOR
01001  192*           IF(NSQ1(JJ1).GT.0) GO TO 135
01003  193*           T2 = DD*T(L)+DN*(SUMCV+Q(L))/SUMC
01003  194*   C       OBTAIN THE CALCULATED TEMPERATURE DIFFERENCE
01004  195*           T1 = ABS(T(L)-T2)
01004  196*   C       STORE THE NEW TEMPERATURE
01005  197*           T(L) = T2
01005  198*   C       SAVE THE MAXIMUM ARITHMETIC RELAXATION CHANGE
01006  199*           IF(TCGM.GE.T1) GO TO 165
01010  200*           TCGM = T1
01011  201*           KON(37) = L
01012  202*       165 CONTINUE
01012  203*   C       SEE IF RELAXATION CRITERIA WAS MET .
01014  204*           IF(TCGM.LE.CON(19)) GO TO 175
01016  205*       170 CONTINUE
01016  206*   C       STORE THE MAXIMUM ARITHMETIC RELAXATION CHANGE
01020  207*       175 CON(30) = TCGM
01020  208*   C       COMPUTE THE ARITHMETIC TEMPERATURE CHANGE
01021  209*           TCGM = 0.0
01022  210*           DO 180 I = L1,NNC
01025  211*           LE = IE+I
01026  212*           T1 = ABS(T(I)-X(LE))
01027  213*           IF(T1.LT.TCGM) GO TO 180
01031  214*           TCGM = T1
01032  215*           KON(38) = I
01033  216*       180 CONTINUE
01033  217*   C       SEE IF ATMPCA WAS SATISFIED
01035  218*           IF(TCGM.GT.CON(11)) GO TO 125
01037  219*           CON(16) = TCGM
01040  220*       185 KON(12) = 0
01041  221*           CALL VARBL2
01041  222*   C       CHECK THE BACKUP SWITCH
01042  223*           IF(KON(12).NE.0) GO TO 105
01042  224*   C       ADVANCE TIME
01044  225*           CON(13) = CON(1)
01045  226*           TSUM = TSUM+TSTEP
01046  227*           TSTEPO = TSTEP
01047  228*           DO 200 I = 1,NND
01052  229*           LE = IE+I
```

```
01053            LEH = IEH+I
01054        200 X(LEH) = X(LE)
01056            TSTEP = DELTA*0.95
01056      C     CHECK FOR TIME TO PRINT
01057            IF(TSUM.GE.CON(18)) GO TO 190
01057      C     CHECK FOR PRINT EVERY ITERATION
01061            IF(KON(7).EQ.0) GO TO 10
01063            CALL OUTCAL
01064            GO TO 10
01064      C     TRY TO EVEN THE OUTPUT INTERVALS
01065        190 TPRINT = TPRINT+TSUM
01066        195 CALL OUTCAL
01066      C     IS TIME GREATER THAN END COMPUTE TIME
01067            IF(CON(1)*1.000001.LT.CON(3)) GO TO 5
01071            NTH = IE
01072            NDIM = NLA
01073            RETURN
01074        995 WRITE(6,885)
01076            GO TO 1000
01077        996 WRITE(6,886)
01101            GO TO 1000
01102        997 WRITE(6,887)
01104            GO TO 1000
01105        998 WRITE(6,888) I
01110            GO TO 1000
01111        999 WRITE(6,889)
01113       1000 CALL OUTCAL
01114            CALL EXIT
01115        885 FORMAT(46H CNDUFR REQUIRES SHORT PSEUDO-COMPUTE SEQUENCE)
01116        886 FORMAT(24H CSGMIN ZERO OR NEGATIVE)
01117        887 FORMAT(20H TIME STEP TOO SMALL)
01120        888 FORMAT(I8,20H LOCATIONS AVAILABLE)
01121        889 FORMAT(19H NO OUTPUT INTERVAL)
01122            END

    END OF UNIVAC 1108 FORTRAN V COMPILATION.     0 *DIAGNOSTIC* MESSAGE(S)
   CNDUFR  CODE    RELOCATABLE
```

CNQUIK

```
@I   FOR**   CNQUIK                                                                                  18 FEB 71
UNIVAC 1108 FORTRAN V ATHENA VERSION 131K-10D   CREATED ON 20 AUG 70
THIS COMPILATION WAS DONE ON 09 JUN 70 AT 23:15:06

 SUBROUTINE CNQUIK    ENTRY POINT 003643

 STORAGE USED (BLOCK, NAME, LENGTH)

       0001    *CODE          003660
       0000    *CONST+TEMP    000100
       0002    *SIMPLE VAR    000054
       0004    *ARRAYS        000000
       0005    *BLANK         000000
       0006    TITLE    000001
       0007    TEMP     000001
       0010    CAP      000001
       0011    SOURCE   000001
       0012    COND     000001
       0013    PC1      000001
       0014    PC2      000001
       0015    KONST    000001
       0016    ARRAY    000001
       0017    FIXCON   000001
       0020    XSPACE   000003
       0021    DIMLNS   000010

 EXTERNAL REFERENCES (BLOCK, NAME)

       0022    VARBL1
       0023    D1D1WM
       0024    PLYAWM
       0025    D2D1WM
       0026    VARBL2
       0027    OUTCAL
       0030    EXIT
       0031    NERR2$
       0032    EXP
       0033    NWDU$
       0034    NIO2$
       0035    NER10$

 STORAGE ASSIGNMENT FOR VARIABLES (BLOCK, TYPE, RELATIVE LOCATION, NAME)

  0016 R 000000 A        0010 R 000000 C        0002 R 000017 CKM      0017 R 000000 CON      0002 R 000023 C1
  0002 R 000024 C2       0002 R 000045 DD       0002 R 000043 DELTA    0002 R 000044 DN       0002 R 000040 CT1
  0002 R 000041 DT2      0012 R 000000 G        0002 R 000036 GV       0002 R 000032 G1       0002 R 000033 G2
  0006 R 000000 H        0002 I 000005 I        0002 I 000002 IE       0002 I 000003 IEH      0002 I 000047 JJ1
  0002 I 000050 JJ2      0002 I 000014 J1       0002 I 000015 J2       0015 I 000000 K        0017 I 000000 KON
  0002 I 000051 L        0002 I 000021 LA       0002 I 000046 LAX      0002 I 000013 LE       0002 I 000037 LEA
  0002 I 000042 LEH      0002 I 000027 LG       0002 I 000022 LK       0021 I 000006 LSQ1     0021 I 000007 LSQ2
  0002 I 000030 LTA      0002 I 000006 L1       0021 I 000005 NAT      0021 I 000004 NCT      0020 I 000000 NDIM
  0021 I 000003 NGT      0002 I 000004 NLA      0021 I 000001 NNA      0002 I 000001 NNC      0021 I 000000 NND
  0021 I 000002 NNT      0013 I 000000 NSQ1     0014 I 000000 NSQ2     0020 I 000001 NTH      0002 I 000020 NTYPE
  0020   000002 NX       0002 R 000000 PASS     0011 R 000000 Q        0002 R 000025 Q1       0002 R 000026 Q2
  0002 R 000052 SUMC     0002 R 000053 SUMCV    0007 R 000000 T        0002 R 000016 TCGM     0002 R 000031 TM
```

CNQUIK

```
0002 R 000011 TPRINT   0002 R 000007 TSTEP    0002 R 000010 TSTEPO   0002 R 000012 TSUM     0002 R 000034 T1
0002 R 000035 T2       0020 R 000002 X        0015 R 000000 XK       0001   000132 10L      0001   003617 1000L
0001   000344 1005L    0001   000365 1010L    0001   000403 1012L    0001   000436 1015L    0011   000444 1020L
0001   003432 1025G    0001   000504 1025L    0001   000530 1030L    0001   000551 1032L    0001   000607 1035L
0001   000615 1040L    0001   000660 1045L    0001   002150 105L     0001   003505 1052G    0001   002217 115L
0001   002230 120L     0001   002236 125L     0001   002244 130L     0001   002574 135L     0001   003342 140L
0001   003345 145L     0001   000141 15L      0001   003412 165L     0001   003423 175L     0001   003454 180L
0001   003465 185L     0001   003537 190L     0001   003542 195L     0001   000702 1998L    0001   000705 1999L
0001   000710 2000L    0001   001250 2005L    0001   001255 2007L    0001   001275 2010L    0001   001300 2015L
0001   001316 2017L    0001   000213 202G     0001   001352 2020L    0001   001360 2025L    0001   001420 2030L
0001   001425 2032L    0001   001450 2035L    0001   001453 2040L    0001   001474 2042L    0001   001533 2045L
0001   001541 2050L    0001   001604 2055L    0001   001632 2060L    0001   001636 2065L    0001   000232 214G
0001   000273 234G     0001   000153 25L      0001   001642 2998L    0001   001670 2999L    0001   000166 30L
0001   001673 3000L    0001   002662 3005L    0001   002667 3007L    0001   002707 3010L    0001   002712 3015L
0001   002730 3017L    0001   002764 3020L    0001   002772 3025L    0001   003032 3030L    0001   003037 3032L
0001   003062 3035L    0001   003065 3040L    0001   003106 3042L    0001   003145 3045L    0001   003153 3050L
0001   003216 3055L    0001   003244 3060L    0001   003250 3065L    0001   003254 3998L    0001   003302 3999L
0001   003305 4000L    0001   000753 4005L    0001   000760 4010L    0001   000761 4012L    0001   001000 4015L
0001   001001 4017L    0001   001016 4020L    0001   001032 4022L    0001   001047 4025L    0001   001055 4030L
0001   001113 4035L    0001   001127 4037L    0001   001144 4040L    0001   000244 45L      0001   001152 4998L
0001   001156 4999L    0001   000120 5L       0001   001161 50L      0001   001161 5000L    0001   002366 5005L
0001   002373 5010L    0001   002374 5012L    0001   002413 5015L    0001   002414 5017L    0001   002431 5020L
0001   002445 5022L    0001   002462 5025L    0001   002470 5030L    0001   002526 5035L    0001   002542 5037L
0001   002557 5040L    0001   002020 507G     0001   002154 543G     0001   001724 55L      0001   002205 560G
0001   002565 5998L    0001   002571 5999L    0001   001727 60L      0001   002574 6000L    0001   002273 607G
0001   002314 616G     0001   001775 65L      0000   000000 885F     0000   000011 886F     0000   000016 887F
0000   000023 888F     0000   000030 889F     0001   002044 90L      0001   002124 95L      0001   003561 995L
0001   003567 996L     0001   003575 997L     0001   003603 998L     0001   003612 999L
```

```
00101     1*           SUBROUTINE CNQUIK
00101     2*     C     COMBINED EXPONENTIAL PREDICTION AND DUFORT-FRANKEL SINDA ROUTINE
00101     3*     C     THE SHORT PSEUDO-COMPUTE SEQUENCE IS REQUIRED
00103     4*           INCLUDE COMM,LIST
00104     4*           COMMON /TITLE/H(1) /TEMP/T(1) /CAP/C(1) /SOURCE/Q(1) /COND/G(1)
00105     4*           COMMON /PC1/NSQ1(1) /PC2/NSQ2(1) /KONST/K(1) /ARRAY/A(1)
00106     4*           COMMON /FIXCON/KON(1) /XSPACE/NDIM,NTH,X(1)
00107     4*           COMMON /DIMENS/ NND,NNA,NNT,NGT,NCT,NAT,LSQ1,LSQ2
00110     4*           DIMENSION CON(1),XK(1),NX(1)
00111     4*           EQUIVALENCE (KON(1),CON(1)),(K(1),XK(1)),(X(1),NX(1))
00112     4*           END
00113     5*           INCLUDE DEFF,LIST
00113     5*     C*************  CONTROL CONSTANT DEFINITIONS AND NAMES  ****************
00113     5*     C     CONTROL CONSTANT 1 CONTAINS THE NEW PROBLEM TIME               (TIMEN)
00113     5*     C     CONTROL CONSTANT 2 CONTAINS THE TIME STEP USED                (DTIMEU)
00113     5*     C     CONTROL CONSTANT 3 CONTAINS THE PROBLEM STOP TIME             (TIMEND)
00113     5*     C     CONTROL CONSTANT 4 CONTAINS THE TIME STEP FACTOR,EXPLICIT (CSGFAC)
00113     5*     C     CC5 IS THE INPUT NUMBER OF ITERATION DO LOOPS, INTEGER        (NLOOP)
00113     5*     C     CC6 CONTAINS THE DIFFUSION TEMPERATURE CHANGE ALLOWED        (DTMPCA)
00113     5*     C     CC7 CONTAINS THE OUTPUT EACH ITERATION SWITCH                (OPEITR)
00113     5*     C     CC8 CONTAINS THE MAXIMUM ALLOWED TIME STEP                   (DTIMEH)
00113     5*     C     CC9 CONTAINS THE NEW ARITHMETIC TEMP. DAMPING FACTOR          (DAMPA)
00113     5*     C     CC10 CONTAINS THE NEW DIFFUSION TEMP. DAMPING FACTOR          (DAMPD)
00113     5*     C     CC11 CONTAINS THE MAXIMUM ALLOWED ARITHMETIC TEMP. CHANGE (ATMPCA)
00113     5*     C     CC12 CONTAINS THE BACKUP SWITCH CHECKED AFTER VARIABLES      (BACKUP)
00113     5*     C     CC13 CONTAINS THE PRESENT TIME OR PROBLEM START TIME          (TIMEO)
00113     5*     C     CC14 CONTAINS THE MEAN TIME BETWEEN AN ITERATION              (TIMEM)
00113     5*     C     CC15 CONTAINS THE DIFFUSION TEMPERATURE CHANGE CALCULATED (DTMPCC)
```

CNQUIK

```
00113     5*   C      CC16 CONTAINS ARITHMETIC TEMPERATURE CHANGE CALCULATED     (ATMPCC)
00113     5*   C      CONTROL CONSTANT 17 IS RESERVED FOR THE C/SG MINIMUM       (CSGMIN)
00113     5*   C      CONTROL CONSTANT 18 CONTAINS THE OUTPUT INTERVAL           (OUTPUT)
00113     5*   C      CC19 CONTAINS THE ARITHMETIC RELAXATION CRITERIA ALLOWED   (ARLXCA)
00113     5*   C      CC20 CONTAINS THE NUMBER OF RELAXATION LOOPS USED,INTEGER (LOOPCT)
00113     5*   C      CC21 CONTAINS THE MINIMUM ALLOWED TIME STEP                (DTIMEL)
00113     5*   C      CC22 IS FOR THE INPUT TIME STEP IMPLICIT                   (DTIMEI)
00113     5*   C      CC23 CONTAINS THE C/SG MAXIMUM                             (CSGMAX)
00113     5*   C      CC24 CONTAINS THE C/SG RANGE ALLOWED                       (CSGRAL)
00113     5*   C      CC25 CONTAINS THE C/SG RANGE CALCULATED                    (CSGRCL)
00113     5*   C      CC26 CONTAINS THE DIFFUSION RELAXATION CRITERIA ALLOWED    (DRLXCA)
00113     5*   C      CC27 CONTAINS THE DIFFUSION RELAXATION CHANGE CALCULATED   (DRLXCC)
00113     5*   C      CC28 CONTAINS THE LINE COUNTER, INTEGER                    (LINECT)
00113     5*   C      CC29 CONTAINS THE PAGE COUNTER, INTEGER                    (PAGECT)
00113     5*   C      CC30 CONTAINS ARITHMETIC RELAXATION CHANGE CALCULATED      (ARLXCC)
00113     5*   C      CC31 IS INDICATOR, 0=THERMAL SPCS,1=THERMAL LPCS,2=GENERAL (LSPCS)
00113     5*   C      CC32 CONTAINS THE ENERGY BALANCE OF THE SYSTEM, IN - OUT   (ENGBAL)
00113     5*   C      CC33 CONTAINS THE DESIRED ENERGY BALANCE, USER INPUT       (BALENG)
00113     5*   C      CC34 CONTAINS THE NOCOPY SWITCH FOR MATRIX USERS           (NOCOPY)
00113     5*   C      CC35 CONTAINS RELATIVE NODE NUMBER OF CSGMIN
00113     5*   C      CC36 CONTAINS RELATIVE NODE NUMBER OF DTMPCC
00113     5*   C      CC37 CONTAINS RELATIVE NODE NUMBER OF ARLXCC
00113     5*   C      CC38 CONTAINS RELATIVE NODE NUMBER OF ATMPCC
00113     5*   C      CC39-40-41-42-43 CONTAIN DUMMY INTEGER CONSTANTS    (I-J-K-L-MTEST)
00113     5*   C      CC44-45-46-47-48 CONTAIN DUMMY FLOATING CONSTANTS  (R-S-T-U-VTEST)
00113     5*   C      CC49 IS THE QUASI-LINEARIZATION INTERVAL FOR CINDSM        (LAXFAC)
00113     5*   C      CC50 IS NOT USED AT PRESENT
00114     5*          END
00115     6*          IF(CON(4).LT.1.0) CON(4) = 1.0
00117     7*          IF(KON(5).LE.0) KON(5) = 1
00121     8*          IF(CON(6).LE.0.) CON(6) = 1.E+8
00123     9*          IF(CON(8).LE.0.) CON(8) = 1.E+8
00125    10*          IF(CON(9).LE.0.) CON(9) = 1.0
00127    11*          IF(CON(11).LE.0.) CON(11) = 1.E+8
00131    12*          IF(CON(18).LE.0.) GO TO 999
00133    13*          IF(CON(19).LE.0.) CON(19) = 1.E+8
00135    14*          IF(KON(31).NE.0) GO TO 995
00137    15*          PASS = -1.0
00140    16*          NNC = NND+NNA
00141    17*          IE = NTH
00142    18*          IEH = NTH+NNC
00143    19*          NLA = NDIM
00144    20*          NTH = NTH+NNC+NND
00145    21*          NDIM = NDIM-NNC-NND
00145    22*   C      CHECK FOR EXTRA LOCATIONS FOR CALCULATED NODES
00146    23*          I = NLA-NNC-NND
00147    24*          IF(I.LT.0) GO TO 998
00151    25*          L1 = NND+1
00152    26*          TSTEP = CON(18)
00153    27*          TSTEPO = 0.0
00154    28*          TPRINT = CON(13)
00154    29*   C      INITALIZE TIME SUM BETWEEN OUTPUT INTERVALS
00155    30*        5 TSUM = 0.0
00155    31*   C      DOES OLD TIME PLUS THE OUTPUT INTERVAL EXCEED THE STOP TIME
00156    32*          IF(CON(13)+CON(18).LE.CON(3)) GO TO 10
00156    33*   C      DONT EXCEED IT
00160    34*          CON(18) = CON(3)-CON(13)
00160    35*   C      IS THE TIME STEP LARGER THAN ALLOWED
00161    36*       10 IF(TSTEP.LE.CON(8)) GO TO 15
00163    37*          TSTEP = CON(8)
```

```
00163   38*   C       DOES THE TIME SUM PLUS THE TIME STEP EXCEED OUTPUT INTERVAL
00164   39*        15 IF(TSUM+TSTEP-CON(18)) 25,30,20
00164   40*   C       DONT EXCEED IT
00167   41*        20 TSTEP = CON(18)-TSUM
00170   42*           GO TO 30
00170   43*   C       DOES TIME SUM PLUS TWO TIME STEPS EXCEED OUTPUT INTERVAL
00171   44*        25 IF(TSUM+2.0*TSTEP.LE.CON(18)) GO TO 30
00171   45*   C       APPROACH THE OUTPUT INTERVAL GRADUALLY
00173   46*           TSTEP = (CON(18)-TSUM)/2.0
00173   47*   C       STORE DELTA TIME STEP IN THE CONSTANTS
00174   48*        30 CON(2) = TSTEP
00174   49*   C       IS THE TIME STEP USED LESS THAN THE TIME STEP ALLOWED
00175   50*           IF(TSTEP.LT.CON(21)) GO TO 997
00175   51*   C       CALCULATE THE NEW TIME
00177   52*           CON(1) = TPRINT+TSUM+TSTEP
00177   53*   C       COMPUTE THE MEAN TIME BETWEEN ITERATIONS
00200   54*           CON(14) = (CON(1)+CON(13))/2.0
00200   55*   C       ZERO OUT ALL SOURCE LOCATIONS AND EXTRA LOCATIONS
00201   56*           DO 35 I = 1,NND
00204   57*           LE = IE+I
00205   58*           X(LE) = 0.0
00206   59*           Q(I) = 0.0
00207   60*        35 CONTINUE
00207   61*   C       SHIFT THE ARITHMETIC TEMPERATURES INTO THE EXTRA LOCATIONS
00211   62*           IF(NNA.LE.0) GO TO 45
00213   63*           DO 40 I = L1,NNC
00216   64*           Q(I) = 0.0
00217   65*           LE = IE+I
00220   66*           X(LE) = T(I)
00221   67*        40 CONTINUE
00223   68*        45 KON(12) = 0
00224   69*           CALL VARBL1
00225   70*           IF(KON(12).NE.0) GO TO 10
00227   71*           J1 = 0
00230   72*           J2 = 1
00231   73*           TCGM = 0.0
00232   74*           CKM = 1.E+10
00232   75*   C       CALCULATE Q SUM AND G SUM
00233   76*           DO 85 I = 1,NND
00236   77*           LE = IE+I
00237   78*           INCLUDE VARC,LIST
00240   78*           IF(FLD(1,1,NSQ1(J1+1)).EQ.0) GO TO 2000
00242   78*           NTYPE = FLD(0,5,NSQ2(J2))                                    *NEW
00243   78*           LA = FLD(5,17,NSQ2(J2))                                      *NEW
00244   78*           LK = FLD(22,14,NSQ2(J2))                                     *NEW
00245   78*           GO TO (1005,1010,1015,1020,1025,1030,1035,1040,1045), NTYPE  **-3
00246   78*      1005 CALL D1D1WM(T(I),A(LA),XK(LK),C(I))
00247   78*           GO TO 1999
00250   78*      1010 CALL D1D1WM(T(I),A(LA),XK(LK),C1)
00251   78*      1012 J2 = J2+1
00252   78*           LA = FLD(5,17,NSQ2(J2))                                      *NEW
00253   78*           LK = FLD(22,14,NSQ2(J2))                                     *NEW
00254   78*           CALL D1D1WM(T(I),A(LA),XK(LK),C2)                            **-2
00255   78*           GO TO 1998
00256   78*      1015 C1 = XK(LK)*XK(LA)
00257   78*           GO TO 1012
00260   78*      1020 CALL D1D1WM(T(I),A(LA),XK(LK),C1)
00261   78*           J2 = J2+1
00262   78*           LA = FLD(5,17,NSQ2(J2))                                      *NEW
00263   78*           LK = FLD(22,14,NSQ2(J2))                                     *NEW
```

CNQUIK

```
78*          C2 = XK(LK)*XK(LA)                                          **-2
78*          GO TO 1998
78*     1025 CALL PLYAWM(A(LA),T(I),A(LA+1),XK(LK),C(I))
78*          GO TO 1999
78*     1030 CALL PLYAWM(A(LA),T(I),A(LA+1),XK(LK),C1)
78*     1032 J2 = J2+1
78*          LA = FLD(5,17,NSQ2(J2))                                     *NEW
78*          LK = FLD(22,14,NSQ2(J2))                                    *NEW
78*          CALL PLYAWM(A(LA),T(I),A(LA+1),XK(LK),C2)                   **-2
78*          GO TO 1998
78*     1035 C1 = XK(LK)*XK(LA)
78*          GO TO 1032
78*     1040 CALL PLYAWM(A(LA),T(I),A(LA+1),XK(LK),C1)
78*          J2 = J2+1
78*          LA = FLD(5,17,NSQ2(J2))                                     *NEW
78*          LK = FLD(22,14,NSQ2(J2))                                    *NEW
78*          C2 = XK(LK)*XK(LA)                                          **-2
78*          GO TO 1998
78*     1045 CALL D2D1WM(T(I),CON(14),A(LA),XK(LK),C(I))
78*          GO TO 1999
78*     1998 C(I) = C1+C2
78*     1999 J2 = J2+1
78*     2000 CONTINUE
78*          END
79*          INCLUDE VARQ,LIST
79*          IF(FLD(4,1,NSQ1(J1+1)).EQ.0) GO TO 5000
79*          NTYPE = FLD(0,5,NSQ2(J2))                                   *NEW
79*          LA = FLD(5,17,NSQ2(J2))                                     *NEW
79*          LK = FLD(22,14,NSQ2(J2))                                    *NEW
79*          GO TO (4005,4010,4015,4020,4025,4030,4035,4040,4030), NTYPE **-3
79*     4005 Q(I) = XK(LK)+Q(I)
79*          GO TO 4999
79*     4010 Q1 = 0.0
79*     4012 CALL D1D1WM(T(I),A(LA),XK(LK),Q2)
79*          GO TO 4998
79*     4015 Q1 = 0.0
79*     4017 CALL D1D1WM(CON(14),A(LA),XK(LK),Q2)
79*          GO TO 4998
79*     4020 CALL D1D1WM(CON(14),A(LA),XK(LK),Q1)
79*     4022 J2 = J2+1
79*          LA = FLD(5,17,NSQ2(J2))                                     *NEW
79*          LK = FLD(22,14,NSQ2(J2))                                    *NEW
79*          GO TO 4017                                                  **-2
79*     4025 Q1 = XK(LK)*XK(LA)
79*          GO TO 4022
79*     4030 CALL D1D1WM(CON(14),A(LA),XK(LK),Q1)
79*          J2 = J2+1
79*          LA = FLD(5,17,NSQ2(J2))                                     *NEW
79*          LK = FLD(22,14,NSQ2(J2))                                    *NEW
79*          Q2 = XK(LK)*XK(LA)                                          **-2
79*          GO TO 4998
79*     4035 CALL D1D1WM(CON(14),A(LA),XK(LK),Q1)
79*     4037 J2 = J2+1
79*          LA = FLD(5,17,NSQ2(J2))                                     *NEW
79*          LK = FLD(22,14,NSQ2(J2))                                    *NEW
79*          GO TO 4012                                                  **-2
79*     4040 Q1 = XK(LK)*XK(LA)
79*          GO TO 4037
79*     4998 Q(I) = Q1+Q2+Q(I)
79*     4999 J2 = J2+1
```

.CNQUIK

```
79*        5000 CONTINUE
79*             END
80*          50 J1 = J1+1
81*             LG = FLD(5,16,NSQ1(J1))
82*             IF(LG.EQ.0) GO TO 65
83*             LTA = FLD(22,14,NSQ1(J1))
84*             INCLUDE VARG,LIST
85*       C     CHECK FOR RADIATION CONDUCTOR
85*             IF(FLD(2,1,NSQ1(J1)).EQ.0) GO TO 3000
85*             NTYPE = FLD(0,5,NSQ2(J2))                                                   *NEW
85*             LA = FLD(5,17,NSQ2(J2))                                                     *NEW
85*             LK = FLD(22,14,NSQ2(J2))                                                    *NEW
85*             GOTO(2005,2010,2015,2020,2025,2030,2035,2040,2045,2050,2055,                **-3
85*            $     2060,2065), NTYPE
85*        2005 TM = (T(I)+T(LTA))/2.0
85*        2007 CALL D1D1WM(TM,A(LA),XK(LK),G(LG))
85*             GO TO 2999
85*        2010 TM = T(I)
85*             GO TO 2007
85*        2015 CALL D1D1WM(T(I),A(LA),XK(LK),G1)
85*        2017 J2 = J2+1
85*             LA = FLD(5,17,NSQ2(J2))                                                     *NEW
85*             LK = FLD(22,14,NSQ2(J2))                                                    *NEW
85*             CALL D1D1WM(T(LTA),A(LA),XK(LK),G2)                                         **-2
85*             GO TO 2998
85*        2020 G1 = XK(LK)*XK(LA)
85*             GO TO 2017
85*        2025 CALL D1D1WM(T(I),A(LA),XK(LK),G1)
85*             J2 = J2+1
85*             LA = FLD(5,17,NSQ2(J2))                                                     *NEW
85*             LK = FLD(22,14,NSQ2(J2))                                                    *NEW
85*             G2 = XK(LK)*XK(LA)                                                          **-2
85*             GO TO 2998
85*        2030 TM = (T(I)+T(LTA))/2.0
85*        2032 CALL PLYAWM(A(LA),TM,A(LA+1),XK(LK),G(LG))
85*             GO TO 2999
85*        2035 TM = T(I)
85*             GO TO 2032
85*        2040 CALL PLYAWM(A(LA),T(I),A(LA+1),XK(LK),G1)
85*        2042 J2 = J2+1
85*             LA = FLD(5,17,NSQ2(J2))                                                     *NEW
85*             LK = FLD(22,14,NSQ2(J2))                                                    *NEW
85*             CALL PLYAWM(A(LA),T(LTA),A(LA+1),XK(LK),G2)                                 **-2
85*             GO TO 2998
85*        2045 G1 = XK(LK)*XK(LA)
85*             GO TO 2042
85*        2050 CALL PLYAWM(A(LA),T(I),A(LA+1),XK(LK),G1)
85*             J2 = J2+1
85*             LA = FLD(5,17,NSQ2(J2))                                                     *NEW
85*             LK = FLD(22,14,NSQ2(J2))                                                    *NEW
85*             G2 = XK(LK)*XK(LA)                                                          **-2
85*             GO TO 2998
85*        2055 TM = (T(I)+T(LTA))/2.0
85*             CALL D2D1WM(TM,CON(14),A(LA),XK(LK),G(LG))
85*             GO TO 2999
85*        2060 TM = T(LTA)
85*             GO TO 2007
85*        2065 TM = T(LTA)
85*             GO TO 2032
85*        2998 G(LG) = 1./(1./G1+1./G2)
```

CNOUIK

```
      IF(FLD(3,1,NSQ1(J1)).EQ.1) G(LG) = G1*G2
 2999 J2 = J2+1
 3000 CONTINUE
      END
      IF(FLD(3,1,NSQ1(J1)).EQ.0) GO TO 55
      T1 = T(I)+460.0
      T2 = T(LTA)+460.0
      GV = G(LG)*(T1*T1+T2*T2)*(T1+T2)
      GO TO 60
   55 GV = G(LG)
C     OBTAIN THE Q PLUS SUMMATION G*TA TERM
   60 Q(I) = Q(I)+GV*T(LTA)
C     SAVE SUMMATION OF CONDUCTORS
      X(LE) = X(LE)+GV
C     CHECK FOR ADJOINING DIFFUSION NODE OR ONE-WAY CONDUCTOR
      IF(LTA.GT.NND.OR.FLD(21,1,NSQ1(J1)).EQ.1) GO TO 65
C     SAVE SUMMATION OF CONDUCTORS FOR ADJOINING NODE
      LEA = IE+LTA
      X(LEA) = X(LEA)+GV
      Q(LTA) = Q(LTA)+GV*T(I)
C     CHECK FOR LAST CONDUCTOR
   65 IF(NSQ1(J1).GT.0) GO TO 50
   85 CONTINUE
      DT1 = TSTEP/(TSTEP+TSTEPO)
      DT2 = TSTEPO/(TSTEP+TSTEPO)
C     OBTAIN NEW DIFFUSION TEMPERATURES, DTMPCC AND CSGMIN
      DO 100 I = 1,NND
      LE = IE+I
      LEH = IEH+I
C     CALCULATE C/SK MINIMUM
      T1 = C(I)/X(LE)
      IF(T1.GE.CKM) GO TO 90
      CKM = T1
      KON(35) = I
C     COMPUTE NEW TEMPERATURES USING CALCULATED SOURCE TERMS
   90 T1=(DT1*X(LEH)*(C(I)/TSTEP-X(LE))+Q(I))/(C(I)*(1.-DT2)/TSTEP+DT2*
     $X(LE))
      T2 = 1.0/EXP(TSTEP*X(LE)/C(I))
      T1 = (T1+(1.0-T2)*Q(I)/X(LE)+T2*T(I))*0.5
C     CALCULATE THE ABSOLUTE VALUE TEMPERATURE CHANGE
      T2 = ABS(T(I)-T1)
C     SAVE THE LARGEST TEMPERATURE CHANGE
      IF(TCGM.GE.T2) GO TO 95
      TCGM = T2
      KON(36) = I
C     STORE THE TEMPERATURES
   95 X(LE) = T(I)
      T(I) = T1
  100 CONTINUE
      CON(17) = CKM
      DELTA = CKM*CON(4)
      IF(CKM.LE.0.0) GO TO 996
C     CHECK FOR FIRST PASS
      IF(PASS.GT.0.0) GO TO 115
C     UNDO THE TEMPERATURE CALCULATIONS
  105 DO 110 I = 1,NNC
      LE = IE+I
      T(I) = X(LE)
  110 CONTINUE
      IF(PASS.GT.0.0) GO TO 15
```

```
00553  142*          PASS = 1.0
00554  143*          CON(1) = TPRINT
00555  144*          CON(2) = 0.0
00556  145*          TSTEP = CKM*0.9
00557  146*          DO 112 I = 1,NND
00562  147*          LEH = IEH+I
00563  148*      112 X(LEH) = T(I)
00565  149*          GO TO 195
00565  150*    C     IS THE TIME STEP USED LESS THAN THE TIME STEP CALCULATED
00566  151*      115 IF(TSTEP.LE.DELTA) GO TO 130
00566  152*    C     COMPUTE THE TIME STEP
00570  153*          TSTEP = DELTA*0.95
00571  154*          GO TO 105
00572  155*      120 TSTEP = 0.95*TSTEP*CON(6)/TCGM
00573  156*          GO TO 105
00574  157*      125 TSTEP = 0.95*TSTEP*CON(11)/TCGM
00575  158*          GO TO 105
00575  159*    C     SEE IF THE TEMPERATURE CHANGE WAS TOO LARGE
00576  160*      130 IF(TCGM.GT.CON(6)) GO TO 120
00576  161*    C     STORE THE MAXIMUM DIFFUSION TEMPERATURE CHANGE
00600  162*          CON(15) = TCGM
00600  163*    C     CHECK TO SEE IF THERE ARE ANY ARITHMETIC NODES
00601  164*          IF(NNA.LE.0) GO TO 185
00601  165*    C     COMPUTE ARITHMETIC TEMPERATURES BY SUCCESSIVE POINT OVER-RELAX
00603  166*          DN = CON(9)
00604  167*          DD = 1.0-DN
00605  168*          LAX = KON(5)
00606  169*          DO 170 I = 1,LAX
00611  170*          JJ1 = J1
00612  171*          JJ2 = J2
00613  172*          TCGM = 0.0
00614  173*          KON(20) = I
00615  174*          DO 165 L = L1,NNC
00620  175*          SUMC = 0.0
00621  176*          SUMCV = 0.0
00622  177*          IF(I.GT.1) GO TO 6000
00624  178*          INCLUDE VRQ2,LIST
00625  178*          IF(FLD(4,1,NSQ1(JJ1+1)).EQ.0) GO TO 6000
00627  178*          NTYPE = FLD(0,5,NSQ2(JJ2))                                    *NEW
00630  178*          LA = FLD(5,17,NSQ2(JJ2))                                      *NEW
00631  178*          LK = FLD(22,14,NSQ2(JJ2))                                     *NEW
00632  178*          GO TO (5005,5010,5015,5020,5025,5030,5035,5040,5030), NTYPE   **-3
00633  178*     5005 Q(L) = XK(LK)+Q(L)
00634  178*          GO TO 5999
00635  178*     5010 Q1 = 0.0
00636  178*     5012 CALL D1D1WM(T(L),A(LA),XK(LK),Q2)
00637  178*          GO TO 5998
00640  178*     5015 Q1 = 0.0
00641  178*     5017 CALL D1D1WM(CON(14),A(LA),XK(LK),Q2)
00642  178*          GO TO 5998
00643  178*     5020 CALL D1D1WM(CON(14),A(LA),XK(LK),Q1)
00644  178*     5022 JJ2 = JJ2+1
00645  178*          LA = FLD(5,17,NSQ2(JJ2))                                      *NEW
00646  178*          LK = FLD(22,14,NSQ2(JJ2))                                     *NEW
00647  178*          GO TO 5017                                                    **-2
00650  178*     5025 Q1 = XK(LK)*XK(LA)
00651  178*          GO TO 5022
00652  178*     5030 CALL D1D1WM(CON(14),A(LA),XK(LK),Q1)
00653  178*          JJ2 = JJ2+1
00654  178*          LA = FLD(5,17,NSQ2(JJ2))                                      *NEW
```

CNQUIK

```
      LK = FLD(22,14,NSQ2(JJ2))                                         *NEW
      Q2 = XK(LK)*XK(LA)                                                **-2
      GO TO 5998
 5035 CALL D1D1WM(CON(14),A(LA),XK(LK),Q1)
 5037 JJ2 = JJ2+1
      LA = FLD(5,17,NSQ2(JJ2))                                          *NEW
      LK = FLD(22,14,NSQ2(JJ2))                                         *NEW
      GO TO 5012                                                        **-2
 5040 Q1 = XK(LK)*XK(LA)
      GO TO 5037
 5998 Q(L) = Q1+Q2+Q(L)
 5999 JJ2 = JJ2+1
 6000 CONTINUE
      END
  135 JJ1 = JJ1+1
      LG = FLD(5,16,NSQ1(JJ1))
      LTA = FLD(22,14,NSQ1(JJ1))
      IF(I.GT.1) GO TO 4000
      INCLUDE VRG2.LIST
C     CHECK FOR RADIATION CONDUCTOR
      IF(FLD(2,1,NSQ1(JJ1)).EQ.0) GO TO 4000
      NTYPE = FLD(0,5,NSQ2(JJ2))                                        *NEW
      LA = FLD(5,17,NSQ2(JJ2))                                          *NEW
      LK = FLD(22,14,NSQ2(JJ2))                                         *NEW
      GOTO(3005,3010,3015,3020,3025,3030,3035,3040,3045,3050,3055,      **-3
     $     3060,3065), NTYPE
 3005 TM = (T(L)+T(LTA))/2.0
 3007 CALL D1D1WM(TM,A(LA),XK(LK),G(LG))
      GO TO 3999
 3010 TM = T(L)
      GO TO 3007
 3015 CALL D1D1WM(T(L),A(LA),XK(LK),G1)
 3017 JJ2 = JJ2+1
      LA = FLD(5,17,NSQ2(JJ2))                                          *NEW
      LK = FLD(22,14,NSQ2(JJ2))                                         *NEW
      CALL D1D1WM(T(LTA),A(LA),XK(LK),G2)                               **-2
      GO TO 3998
 3020 G1 = XK(LK)*XK(LA)
      GO TO 3017
 3025 CALL D1D1WM(T(L),A(LA),XK(LK),G1)
      JJ2 = JJ2+1
      LA = FLD(5,17,NSQ2(JJ2))                                          *NEW
      LK = FLD(22,14,NSQ2(JJ2))                                         *NEW
      G2 = XK(LK)*XK(LA)                                                **-2
      GO TO 3998
 3030 TM = (T(L)+T(LTA))/2.0
 3032 CALL PLYAWM(A(LA),TM,A(LA+1),XK(LK),G(LG))
      GO TO 3999
 3035 TM = T(L)
      GO TO 3032
 3040 CALL PLYAWM(A(LA),T(L),A(LA+1),XK(LK),G1)
 3042 JJ2 = JJ2+1
      LA = FLD(5,17,NSQ2(JJ2))                                          *NEW
      LK = FLD(22,14,NSQ2(JJ2))                                         *NEW
      CALL PLYAWM(A(LA),T(LTA),A(LA+1),XK(LK),G2)                       **-2
      GO TO 3998
 3045 G1 = XK(LK)*XK(LA)
      GO TO 3042
 3050 CALL PLYAWM(A(LA),T(L),A(LA+1),XK(LK),G1)
      JJ2 = JJ2+1
```

```
00751  184*           LA = FLD(5,17,NSQ2(JJ2))                                  *NEW
00752  184*           LK = FLD(22,14,NSQ2(JJ2))                                 *NEW
00753  184*           G2 = XK(LK)*XK(LA)                                        **-2
00754  184*           GO TO 3998
00755  184*      3055 TM = (T(L)+T(LTA))/2.0
00756  184*           CALL D2D1WM(TM,CON(14),A(LA),XK(LK),G(LG))
00757  184*           GO TO 3999
00760  184*      3060 TM = T(LTA)
00761  184*           GO TO 3007
00762  184*      3065 TM = T(LTA)
00763  184*           GO TO 3032
00764  184*      3998 G(LG) = 1./(1./G1+1./G2)
00765  184*           IF(FLD(3,1,NSQ1(JJ1)).EQ.1) G(LG) = G1*G2
00767  184*      3999 JJ2 = JJ2+1
00770  184*      4000 CONTINUE
00771  184*           END
00772  185*           IF(FLD(3,1,NSQ1(JJ1)).EQ.0) GO TO 140
00774  186*           T1 = T(L)+460.0
00775  187*           T2 = T(LTA)+460.0
00776  188*           GV = G(LG)*(T1*T1+T2*T2)*(T1+T2)
00777  189*           GO TO 145
01000  190*       140 GV = G(LG)
01001  191*       145 SUMC = SUMC+GV
01002  192*           SUMCV = SUMCV+GV*T(LTA)
01002  193*     C     CHECK FOR LAST CONDUCTOR
01003  194*           IF(NSQ1(JJ1).GT.0) GO TO 135
01005  195*           T2 = DD*T(L)+DN*(SUMCV+Q(L))/SUMC
01005  196*     C     OBTAIN THE CALCULATED TEMPERATURE DIFFERENCE
01006  197*           T1 = ABS(T(L)-T2)
01006  198*     C     STORE THE NEW TEMPERATURE
01007  199*           T(L) = T2
01007  200*     C     SAVE THE MAXIMUM ARITHMETIC RELAXATION CHANGE
01010  201*           IF(TCGM.GE.T1) GO TO 165
01012  202*           TCGM = T1
01013  203*           KON(37) = L
01014  204*       165 CONTINUE
01014  205*     C     SEE IF RELAXATION CRITERIA WAS MET
01016  206*           IF(TCGM.LE.CON(19)) GO TO 175
01020  207*       170 CONTINUE
01020  208*     C     STORE THE MAXIMUM ARITHMETIC RELAXATION CHANGE
01022  209*       175 CON(30) = TCGM
01022  210*     C     COMPUTE THE ARITHMETIC TEMPERATURE CHANGE
01023  211*           TCGM = 0.0
01024  212*           DO 180 I = L1,INC
01027  213*           LE = IE+I
01030  214*           T1 = ABS(T(I)-X(LE))
01031  215*           IF(T1.LT.TCGM) GO TO 180
01033  216*           TCGM = T1
01034  217*           KON(38) = I
01035  218*       180 CONTINUE
01035  219*     C     SEE IF ATMPCA WAS SATISFIED
01037  220*           IF(TCGM.GT.CON(11)) GO TO 125
01041  221*           CON(16) = TCGM
01042  222*       185 KON(12) = 0
01043  223*           CALL VARBL2
01043  224*     C     CHECK THE BACKUP SWITCH
01044  225*           IF(KON(12).NE.0) GO TO 105
01044  226*     C     ADVANCE TIME
01046  227*           CON(13) = CON(1)
01047  228*           TSUM = TSUM+TSTEP
```

CNQUIK

```
01050            TSTEPO = TSTEP
01051            DO 200 I = 1,NND
01054            LE = IE+I
01055            LEH = IEH+I
01056        200 X(LEH) = X(LE)
01060            TSTEP = DELTA*0.95
01060       C    CHECK FOR TIME TO PRINT
01061            IF(TSUM.GE.CON(18)) GO TO 190
01061       C    CHECK FOR PRINT EVERY ITERATION
01063            IF(KON(7).EQ.0) GO TO 10
01065            CALL OUTCAL
01066            GO TO 10
01066       C    TRY TO EVEN THE OUTPUT INTERVALS
01067        190 TPRINT = TPRINT+TSUM
01070        195 CALL OUTCAL
01070       C    IS TIME GREATER THAN END COMPUTE TIME
01071            IF(CON(1)*1.000001.LT.CON(3)) GO TO 5
01073            NTH = IE
01074            NDIM = NLA
01075            RETURN
01076        995 WRITE(6,885)
01100            GO TO 1000
01101        996 WRITE(6,886)
01103            GO TO 1000
01104        997 WRITE(6,887)
01106            GO TO 1000
01107        998 WRITE(6,888) I
01112            GO TO 1000
01113        999 WRITE(6,889)
01115       1000 CALL OUTCAL
01116            CALL EXIT
01117        885 FORMAT(46H CNQUIK REQUIRES SHORT PSEUDO-COMPUTE SEQUENCE)
01120        886 FORMAT(24H CSGMIN ZERO OR NEGATIVE)
01121        887 FORMAT(20H TIME STEP TOO SMALL)
01122        888 FORMAT(I8,20H LOCATIONS AVAILABLE)
01123        889 FORMAT(19H NO OUTPUT INTERVAL)
01124            END

     END OF UNIVAC 1108 FORTRAN V COMPILATION.      0 *DIAGNOSTIC* MESSAGE(S)
   CNQUIK  CODE    RELOCATABLE
```

B. COMPUTER LISTINGS OF SINDA IMPLICIT SOLUTION ROUTINES

```
                               CNBACK,CNBACK
QIW    FOR,*          CNBACK,CNBACK                                                          21 FEB 71
UNIVAC 1108 FORTRAN V ATHENA VERSION 131K-10D   CREATED ON 20 AUG 70
THIS COMPILATION WAS DONE ON 09 JUN 70 AT 14:00:11

   SUBROUTINE CNBACK     ENTRY POINT 004236

   STORAGE USED (BLOCK, NAME, LENGTH)

         0001    *CODE          004253
         0000    *CONST+TEMP    000125
         0002    *SIMPLE VAR    000067
         0004    *ARRAYS        000000
         0005    *BLANK         000000
         0006    TITLE    000001
         0007    TEMP     000001
         0010    CAP      000001
         0011    SOURCE   000001
         0012    COND     000001
         0013    PC1      000001
         0014    PC2      000001
         0015    KONST    000001
         0016    ARRAY    000001
         0017    FIXCON   000001
         0020    XSPACE   000003
         0021    DIMENS   000010


   EXTERNAL REFERENCES (BLOCK, NAME)

         0022    VARBL1
         0023    OUTCAL
         0024    D1D1WM
         0025    PLYAWM
         0026    D2D1WM
         0027    TOPLIN
         0030    VARBL2
         0031    EXIT
         0032    NERR2$
         0033    NWDU$
         0034    NIO2$
         0035    NER1U$


   STORAGE ASSIGNMENT FOR VARIABLES (BLOCK, TYPE, RELATIVE LOCATION, NAME)

    0016 R 000000 A          0002 R 000017 AA         0002 R 000016 AN         0010 R 000000 C          0017 R 000000 CON
    0002 R 000034 C1         0002 R 000035 C2         0002 R 000015 DD         0002 R 000014 DN         0012 R 000000 G
    0002 R 000044 GSUM       0002 R 000053 GV         0002 R 000050 G1         0002 R 000040 G2         0006 R 000000 H
    0002 I 000025 I          0002 I 000003 IE1        0002 I 000005 IE2        0002 I 000006 IE3        0002 I 000007 J
    0002 I 000030 JJ         0002 I 000060 JJ1        0002 I 000061 JJ2        0002 I 000021 J1         0002 I 000024 J2
    0015 I 000000 K          0002 I 000057 KK1        0002 I 000063 KK2        0017 I 000000 KON        0002 I 000020 K1
    0002 I 000062 L          0002 I 000032 LA         0002 I 000013 LAX        0002 I 000066 LE         0002 I 000026 LE1
    0002 I 000055 LE2        0002 I 000056 LE3        0002 I 000045 LG         0002 I 000033 LK         0021 I 000006 LS01
    0021 I 000007 LS02       0002 I 000046 LTA        0021 I 000005 NAT        0021 I 000004 NCT        0020 I 000000 NDIM
    0021 I 000003 NGT        0002 I 000002 NLA        0002 I 000001 NN         0021 I 000001 NNA        0002 I 000004 NNC
    0021 I 000000 NND        0021 I 000002 NNT        0013 I 000000 NSQ1       0014 I 000000 NSQ2       0020 I 000001 NTH
    0002 I 000031 NTYPE      0020   000002 NX         0002 R 000000 PASS       0011 R 000000 Q          0002 R 000043 QSUM
```

CNBACK,CNBACK

```
0002 R 000041 Q1        0002 R 000042 Q2        0002 R 000027 RC        0002 R 000022 RLXA      C002 R 000023 RLXD
0002 R 000036 R1        0002 R 000054 R2        0002 R 000037 S         0007 R 000000 T         D002 R 000065 TC6A
0002 R 000064 TCGD      0002 R 000047 TM        0002 R 000010 TPRINT    0002 R 000012 TSTEPN    C002 R 000011 TSUM
0002 R C00051 T1        0002 R 000052 T2        0020 R 000002 X         0015 R 000000 XK        0001   000162 10L
0001   002074 100L      0001   004212 1000L     0001   000420 1005L     0001   000441 1010L     0001   000457 1012L
0001   000512 1015L     0001   000520 1020L     0001   000560 1025L     0001   000604 1030L     0001   000625 1032L
0001   000663 1035L     0001   000671 1040L     0001   000734 1045L     0001   002121 105L      C001   003623 1067G
0001   002154 110L      0001   003674 1114G     0001   003741 1133G     0001   003764 1144G     0001   002172 115L
0001   004027 1163G     0001   002231 120L      0001   002242 125L      0001   002276 145L      0001   000171 15L
0001   002322 150L      0001   002345 155L      0001   002354 160L      0001   002367 165L      0001   002402 170L
0001   002446 175L      0001   002450 180L      0001   002767 185L      0001   003546 190L      0001   003551 195L
0001   000756 1998L     0001   000761 1999L     0001   000174 20L       0001   000764 2000L     0001   001335 2005L
0001   001342 2007L     0001   001362 2010L     0001   001365 2015L     0001   001403 2017L     0001   001437 2020L
0001   001445 2025L     0001   001505 2030L     0001   001512 2032L     0001   001535 2035L     0001   001540 2040L
0001   001561 2042L     0001   001620 2045L     0001   001626 2050L     0001   001671 2055L     0001   001717 2060L
0001   001723 2065L     0001   003615 215L      0001   000266 220G      0001   003634 220L      0001   003641 225L
0001   000310 232G      0001   000314 237G      0001   003645 240L      0001   003666 245L      0001   003721 250L
0001   003735 255L      0001   000352 260G      0001   003754 265L      0001   004006 270L      0001   004023 275L
0001   004052 285L      0001   004102 290L      0001   001727 2998L     0001   001755 2999L     0001   000206 30L
0001   001760 3000L     0001   003055 3005L     0001   003063 3007L     0001   003103 3010L     0001   003107 3015L
0001   003126 3017L     0001   003162 3020L     0001   003170 3025L     0001   003231 3030L     0001   003237 3032L
0001   003262 3035L     0001   003266 3040L     0001   003310 3042L     0001   003347 3045L     0001   003355 3050L
0001   003421 3055L     0001   003450 3060L     0001   003454 3065L     0001   000221 35L       0001   003460 3998L
0001   003506 3999L     0001   000233 40L       0001   003511 4000L     0001   001035 4005L     0001   001042 4010L
0001   001043 4012L     0001   001062 4015L     0001   001063 4017L     0001   001100 4020L     0001   001114 4022L
0001   001131 4025L     0001   001137 4030L     0001   001175 4035L     0001   001211 4037L     0001   001226 4040L
0001   000237 45L       0001   001234 4998L     0001   001240 4999L     0001   000150 5L        0001   001243 5000L
0001   002554 5005L     0001   002562 5010L     0001   002563 5012L     0001   002603 5015L     0001   002604 5017L
0001   002621 5020L     0001   002635 5022L     0001   002652 5025L     0001   002660 5030L     0001   002716 5035L
0001   002732 5037L     0001   002747 5040L     0001   002127 547G      0001   002163 563G      0001   002755 5998L
0001   002762 5999L     0001   000343 60L       0001   002765 6000L     0001   002407 640G      0001   002473 660G
0001   002501 665G      0001   001253 70L       0001   002024 75L       0001   002040 80L       0000   000000 880F
0000   000006 881F      0000   000017 882F      0000   000025 883F      0000   000032 884F      0000   000037 885F
0000   000042 886F      0000   000045 887F      0000   000050 888F      0000   000054 889F      0001   004124 99CL
0001   004132 991L      0001   004140 993L      0001   004146 994L      0001   004155 995L      0001   004163 996L
0001   004171 997L      0001   004177 998L      0001   004205 999L
```

```
00101    1*    C     SUBROUTINE CNBACK
00101    2*    C     IMPLICIT BACKWARD DIFFERENCING EXECUTION SUBROUTINE
00101    3*    C     THE LONG PSEUDO-COMPUTE SEQUENCE IS REQUIRED, SINDA  FORTRAN V
00101    4*    C     ALL NODES RECEIVE A SUCCESSIVE POINT ITERATION
00101    5*    C     RELAXATION CRITERIA MUST BE SPECIFIED
00101    6*    C     OVER-RELAXATION IS ALLOWED, THE DAMPENING FACTORS ARE ADDRESSABLE
00103    7*          INCLUDE COMM,LIST
00104    7*          COMMON /TITLE/H(1) /TEMP/T(1) /CAP/C(1) /SOURCE/Q(1) /COND/G(1)
00105    7*          COMMON /PC1/NSQ1(1) /PC2/NSQ2(1) /KONST/K(1) /ARRAY/A(1)
00106    7*          COMMON /FIXCON/KON(1) /XSPACE/NDIM,NTH,X(1)
00107    7*          COMMON /DIMENS/ NND,NNA,NNT,NGT,NCT,NAT,LSQ1,LSQ2
00110    7*          DIMENSION CON(1),XK(1),NX(1)
00111    7*          EQUIVALENCE (KON(1),CON(1)),(K(1),XK(1)),(X(1),NX(1))
00112    7*          END
00113    8*          INCLUDE DEFF,LIST
00113    8*    C************ CONTROL CONSTANT DEFINITIONS AND NAMES  ****************
00113    8*    C     CONTROL CONSTANT 1 CONTAINS THE NEW PROBLEM TIME       (TIMEN)
00113    8*    C     CONTROL CONSTANT 2 CONTAINS THE TIME STEP USED         (DTIMEU)
00113    8*    C     CONTROL CONSTANT 3 CONTAINS THE PROBLEM STOP TIME      (TIMEND)
00113    8*    C     CONTROL CONSTANT 4 CONTAINS THE TIME STEP FACTOR,EXPLICIT (CSGFAC)
```

CNBACK.CNBACK

```
00113     8*   C      CC5 IS THE INPUT NUMBER OF ITERATION DO LOOPS, INTEGER      (NLOOP)
00113     8*   C      CC6 CONTAINS THE DIFFUSION TEMPERATURE CHANGE ALLOWED      (DTMPCA)
00113     8*   C      CC7 CONTAINS THE OUTPUT EACH ITERATION SWITCH              (OPEITR)
00113     8*   C      CC8 CONTAINS THE MAXIMUM ALLOWED TIME STEP                 (DTIMEH)
00113     8*   C      CC9 CONTAINS THE NEW ARITHMETIC TEMP. DAMPING FACTOR        (DAMPA)
00113     8*   C      CC10 CONTAINS THE NEW DIFFUSION TEMP. DAMPING FACTOR        (DAMPD)
00113     8*   C      CC11 CONTAINS THE MAXIMUM ALLOWED ARITHMETIC TEMP. CHANGE (ATMPCA)
00113     8*   C      CC12 CONTAINS THE BACKUP SWITCH CHECKED AFTER VARIABLES   (BACKUP)
00113     8*   C      CC13 CONTAINS THE PRESENT TIME OR PROBLEM START TIME        (TIMEO)
00113     8*   C      CC14 CONTAINS THE MEAN TIME BETWEEN AN ITERATION            (TIMEM)
00113     8*   C      CC15 CONTAINS THE DIFFUSION TEMPERATURE CHANGE CALCULATED (DTMPCC)
00113     8*   C      CC16 CONTAINS ARITHMETIC TEMPERATURE CHANGE CALCULATED    (ATMPCC)
00113     8*   C      CONTROL CONSTANT 17 IS RESERVED FOR THE C/SG MINIMUM      (CSGMIN)
00113     8*   C      CONTROL CONSTANT 18 CONTAINS THE OUTPUT INTERVAL          (OUTPUT)
00113     8*   C      CC19 CONTAINS THE ARITHMETIC RELAXATION CRITERIA ALLOWED  (ARLXCA)
00113     8*   C      CC20 CONTAINS THE NUMBER OF RELAXATION LOOPS USED,INTEGER (LOOPCT)
00113     8*   C      CC21 CONTAINS THE MINIMUM ALLOWED TIME STEP               (DTIMEL)
00113     8*   C      CC22 IS FOR THE INPUT TIME STEP IMPLICIT                  (DTIMEI)
00113     8*   C      CC23 CONTAINS THE C/SG MAXIMUM                            (CSGMAX)
00113     8*   C      CC24 CONTAINS THE C/SG RANGE ALLOWED                      (CSGRAL)
00113     8*   C      CC25 CONTAINS THE C/SG RANGE CALCULATED                   (CSGRCL)
00113     8*   C      CC26 CONTAINS THE DIFFUSION RELAXATION CRITERIA ALLOWED   (DRLXCA)
00113     8*   C      CC27 CONTAINS THE DIFFUSION RELAXATION CHANGE CALCULATED  (DRLXCC)
00113     8*   C      CC28 CONTAINS THE LINE COUNTER, INTEGER                   (LINECT)
00113     8*   C      CC29 CONTAINS THE PAGE COUNTER, INTEGER                   (PAGECT)
00113     8*   C      CC30 CONTAINS ARITHMETIC RELAXATION CHANGE CALCULATED     (ARLXCC)
00113     8*   C      CC31 IS INDICATOR, 0=THERMAL SPCS,1=THERMAL LPCS,2=GENERAL (LSPCS)
00113     8*   C      CC32 CONTAINS THE ENERGY BALANCE OF THE SYSTEM, IN - OUT  (ENGBAL)
00113     8*   C      CC33 CONTAINS THE DESIRED ENERGY BALANCE, USER INPUT      (BALENG)
00113     8*   C      CC34 CONTAINS THE NOCOPY SWITCH FOR MATRIX USERS          (NOCOPY)
00113     8*   C      CC35 CONTAINS RELATIVE NODE NUMBER OF CSGMIN
00113     8*   C      CC36 CONTAINS RELATIVE NODE NUMBER OF DTMPCC
00113     8*   C      CC37 CONTAINS RELATIVE NODE NUMBER OF ARLXCC
00113     8*   C      CC38 CONTAINS RELATIVE NODE NUMBER OF ATMPCC
00113     8*   C      CC39-40-41-42-43 CONTAIN DUMMY INTEGER CONSTANTS   (I-J-K-L-MTEST)
00113     8*   C      CC44-45-46-47-48 CONTAIN DUMMY FLOATING CONSTANTS  (R-S-T-U-VTEST)
00113     8*   C      CC49 IS THE QUASI-LINEARIZATION INTERVAL FOR CINDSM       (LAXFAC)
00113     8*   C      CC50 IS NOT USED AT PRESENT
00114     8*          END
00115     9*          IF(KON(5).LE.0) GO TO 999
00117    10*          IF(CON(6).LE.0.) CON(6) = 1.E+8
00121    11*          IF(CON(8).LE.0.) CON(8) = 1.E+8
00123    12*          IF(CON(9).LE.0.) CON(9) = 1.0
00125    13*          IF(CON(10).LE.0.) CON(10) = 1.0
00127    14*          IF(CON(11).LE.0.) CON(11) = 1.E+8
00131    15*          IF(CON(3).LE.CON(13)) GO TO 990
00133    16*          IF(CON(18).LE.0.) GO TO 998
00135    17*          IF(NNA.GT.0.AND.CON(19).LE.0.) GO TO 997
00137    18*          IF(CON(22).LE.0.) GO TO 996
00141    19*          IF(NND.GT.0.AND.CON(26).LE.0.) GO TO 995
00143    20*          IF(KON(31).NE.1) GO TO 991
00145    21*          PASS = -1.0
00146    22*          NN = NND+1
00147    23*          NLA = NDIM
00150    24*          IE1 = NTH
00151    25*          NNC = NND+NNA
00152    26*          IE2 = NTH+NNT
00153    27*          IE3 = IE2+NND
00154    28*          J =  2*NND+NNT
00155    29*          NTH = NTH+J
```

CNBACK,CNBACK

```
00156          NDIM =  NDIM-J
00156    C     CHECK FOR EXTRA LOCATIONS FOR CALCULATED NODES
00157          IF(NDIM.LT.0) GO TO 994
00161          TPRINT = CON(13)
00161    C     INITALIZE TIME SUM BETWEEN OUTPUT INTERVALS
00162        5 TSUM = 0.0
00162    C     DOES OLD TIME PLUS THE OUTPUT INTERVAL EXCEED THE STOP TIME
00163          IF(CON(13)+CON(18).GT.CON(3)) CON(18) = CON(3)-CON(13)
00163    C     DONT EXCEED IT
00165       10 TSTEPN = CON(22)
00166          IF(TSTEPN.LE.CON(8)) GO TO 20
00170       15 TSTEPN = CON(8)
00171          GO TO 35
00171    C     DOES THE TIME SUM PLUS THE TIME STEP EXCEED OUTPUT INTERVAL
00172       20 IF(TSUM+TSTEPN-CON(18)) 30,35,25
00172    C     DONT EXCEED IT
00175       25 TSTEPN = CON(18)-TSUM
00176          GO TO 35
00176    C     DOES TIME SUM PLUS TWO TIME STEPS EXCEED OUTPUT INTERVAL
00177       30 IF(TSUM+2.0*TSTEPN.LE.CON(18)) GO TO 35
00177    C     APPROACH THE OUTPUT INTERVAL GRADUALLY
00201          TSTEPN = (CON(18)-TSUM)/2.0
00201    C     STORE DELTA TIME STEP IN THE CONSTANTS
00202       35 CON(2) = TSTEPN
00202    C     CALCULATE THE NEW TIME
00203          IF(PASS.GT.0.) GO TO 40
00205          CON(1) = TPRINT
00206          CON(2) = 0.0
00207          GO TO 45
00210       40 CON(1) = TPRINT+TSUM+TSTEPN
00210    C     COMPUTE THE MEAN TIME BETWEEN ITERATIONS
00211       45 CON(14) = (CON(1)+CON(13))/2.0
00212          LAX = KON(5)
00213          DN = CON(10)
00214          DD = 1.0-DN
00215          AN = CON(9)
00216          AA = 1.0-AN
00216    C     DO THE RELAXATION LOOP
00217          DO 240 K1 = 1,LAX
00222          KON(20) = K1
00223          J1 = 0
00224          RLXA = 0.0
00225          RLXD = 0.0
00226          IF(K1.GT.1) GO TO 110
00230          J2 = 1
00230    C     ZERO OUT ALL SOURCE LOCATIONS AND SHIFT TEMPERATURES
00231          DO 50 I = 1,NNC
00234       50 Q(I) = 0.0
00236          DO 55 I = 1,NNT
00241          LE1 = IE1+I
00242       55 X(LE1) = T(I)
00244          KON(12) = 0
00245          CALL VARBL1
00245    C     CHECK THE BACKUP SWITCH
00246          IF(KON(12).NE.0) GO TO 15
00246    C     CHECK FOR FIRST PASS
00250          IF(PASS.GE.0.) GO TO 60
00252          CALL OUTCAL
00253          PASS = 1.0
00254          GO TO 10
```

```
   60 RC = 1.E+8
      JJ = 0
C     CALCULATE FIRST PASS TEMPERATURES AND CSGMIN
      DO 105 I = 1,NND
      INCLUDE VARC,LIST
C     FOLD DELTAT INTO THE CAPACITANCES*
      IF(FLD(1,1,NSQ1(J1+1)).EQ.0) GO TO 2000
      NTYPE = FLD(0,5,NSQ2(J2))                                        *NEW
      LA = FLD(5,17,NSQ2(J2))                                          *NEW
      LK = FLD(22,14,NSQ2(J2))                                         *NEW
      GO TO (1005,1010,1015,1020,1025,1030,1035,1040,1045), NTYPE      **-3
 1005 CALL D1D1WM(T(I),A(LA),XK(LK),C(I))
      GO TO 1999
 1010 CALL D1D1WM(T(I),A(LA),XK(LK),C1)
 1012 J2 = J2+1
      LA = FLD(5,17,NSQ2(J2))                                          *NEW
      LK = FLD(22,14,NSQ2(J2))                                         *NEW
      CALL D1D1WM(T(I),A(LA),XK(LK),C2)                                **-2
      GO TO 1998
 1015 C1 = XK(LK)*XK(LA)
      GO TO 1012
 1020 CALL D1D1WM(T(I),A(LA),XK(LK),C1)
      J2 = J2+1
      LA = FLD(5,17,NSQ2(J2))                                          *NEW
      LK = FLD(22,14,NSQ2(J2))                                         *NEW
      C2 = XK(LK)*XK(LA)                                               **-2
      GO TO 1998
 1025 CALL PLYAWM(A(LA),T(I),A(LA+1),XK(LK),C(I))
      GO TO 1999
 1030 CALL PLYAWM(A(LA),T(I),A(LA+1),XK(LK),C1)
 1032 J2 = J2+1
      LA = FLD(5,17,NSQ2(J2))                                          *NEW
      LK = FLD(22,14,NSQ2(J2))                                         *NEW
      CALL PLYAWM(A(LA),T(I),A(LA+1),XK(LK),C2)                        **-2
      GO TO 1998
 1035 C1 = XK(LK)*XK(LA)
      GO TO 1032
 1040 CALL PLYAWM(A(LA),T(I),A(LA+1),XK(LK),C1)
      J2 = J2+1
      LA = FLD(5,17,NSQ2(J2))                                          *NEW
      LK = FLD(22,14,NSQ2(J2))                                         *NEW
      C2 = XK(LK)*XK(LA)                                               **-2
      GO TO 1998
 1045 CALL D2D1WM(T(I),CON(14),A(LA),XK(LK),C(I))
      GO TO 1999
 1998 C(I) = C1+C2
 1999 J2 = J2+1
 2000 CONTINUE
      END
      C(I) = C(I)/TSTEPN
      R1 = 0.0
      S  = 0.0
      G2 = 0.0
      INCLUDE VARQ,LIST
      IF(FLD(4,1,NSQ1(J1+1)).EQ.0) GO TO 5000
      NTYPE = FLD(0,5,NSQ2(J2))                                        *NEW
      LA = FLD(5,17,NSQ2(J2))                                          *NEW
      LK = FLD(22,14,NSQ2(J2))                                         *NEW
      GO TO (4005,4010,4015,4020,4025,4030,4035,4040,4030), NTYPE      **-3
 4005 Q(I) = XK(LK)+Q(I)
```

CNBACK,CNBACK

```
100*        GO TO 4999
100*   4010 Q1 = 0.0
100*   4012 CALL D1D1WM(T(I),A(LA),XK(LK),Q2)
100*        GO TO 4998
100*   4015 Q1 = 0.0
100*   4017 CALL D1D1WM(CON(14),A(LA),XK(LK),Q2)
100*        GO TO 4998
100*   4020 CALL D1D1WM(CON(14),A(LA),XK(LK),Q1)
100*   4022 J2 = J2+1
100*        LA = FLD(5,17,NSQ2(J2))                                         *NEW
100*        LK = FLD(22,14,NSQ2(J2))                                        *NEW
100*        GO TO 4017                                                      **-2
100*   4025 Q1 = XK(LK)*XK(LA)
100*        GO TO 4022
100*   4030 CALL D1D1WM(CON(14),A(LA),XK(LK),Q1)
100*        J2 = J2+1
100*        LA = FLD(5,17,NSQ2(J2))                                         *NEW
100*        LK = FLD(22,14,NSQ2(J2))                                        *NEW
100*        Q2 = XK(LK)*XK(LA)                                              **-2
100*        GO TO 4998
100*   4035 CALL D1D1WM(CON(14),A(LA),XK(LK),Q1)
100*   4037 J2 = J2+1
100*        LA = FLD(5,17,NSQ2(J2))                                         *NEW
100*        LK = FLD(22,14,NSQ2(J2))                                        *NEW
100*        GO TO 4012                                                      **-2
100*   4040 Q1 = XK(LK)*XK(LA)
100*        GO TO 4037
100*   4998 Q(I) = Q1+Q2+Q(I)
100*   4999 J2 = J2+1
100*   5000 CONTINUE
100*        END
101*        Q(I) = Q(I)+C(I)*(T(I)+460.0)
102*        QSUM = Q(I)
103*        GSUM = C(I)
104*     70 J1 = J1+1
105*        LG = FLD(5,16,NSQ1(J1))
106*        LTA = FLD(22,14,NSQ1(J1))
107*        INCLUDE VARG,LIST
107*        IF(FLD(2,1,NSQ1(J1)).EQ.0) GO TO 3000
107*        NTYPE = FLD(0,5,NSQ2(J2))                                       *NEW
107*        LA = FLD(5,17,NSQ2(J2))                                         *NEW
107*        LK = FLD(22,14,NSQ2(J2))                                        *NEW
107*        GOTO(2005,2010,2015,2020,2025,2030,2035,2040,2045,2050,2055,    **-3
107*       $     2060,2065), NTYPE
107*   2005 TM = (T(I)+T(LTA))/2.0
107*   2007 CALL D1D1WM(TM,A(LA),XK(LK),G(LG))
107*        GO TO 2999
107*   2010 TM = T(I)
107*        GO TO 2007
107*   2015 CALL D1D1WM(T(I),A(LA),XK(LK),G1)
107*   2017 J2 = J2+1
107*        LA = FLD(5,17,NSQ2(J2))                                         *NEW
107*        LK = FLD(22,14,NSQ2(J2))                                        *NEW
107*        CALL D1D1WM(T(LTA),A(LA),XK(LK),G2)                             **-2
107*        GO TO 2998
107*   2020 G1 = XK(LK)*XK(LA)
107*        GO TO 2017
107*   2025 CALL D1D1WM(T(I),A(LA),XK(LK),G1)
107*        J2 = J2+1
107*        LA = FLD(5,17,NSQ2(J2))                                         *NEW
```

CNBACK,CNBACK

```
00447  107*          LK = FLD(22,14,NSQ2(J2))                                    *NEW
00450  107*          G2 = XK(LK)*XK(LA)                                          **-2
00451  107*          GO TO 2998
00452  107*     2030 TM = (T(I)+T(LTA))/2.0
00453  107*     2032 CALL PLYAWM(A(LA),TM,A(LA+1),XK(LK),G(LG))
00454  107*          GO TO 2999
00455  107*     2035 TM = T(I)
00456  107*          GO TO 2032
00457  107*     2040 CALL PLYAWM(A(LA),T(I),A(LA+1),XK(LK),G1)
00460  107*     2042 J2 = J2+1
00461  107*          LA = FLD(5,17,NSQ2(J2))                                     *NEW
00462  107*          LK = FLD(22,14,NSQ2(J2))                                    *NEW
00463  107*          CALL PLYAWM(A(LA),T(LTA),A(LA+1),XK(LK),G2)                 **-2
00464  107*          GO TO 2998
00465  107*     2045 G1 = XK(LK)*XK(LA)
00466  107*          GO TO 2042
00467  107*     2050 CALL PLYAWM(A(LA),T(I),A(LA+1),XK(LK),G1)
00470  107*          J2 = J2+1
00471  107*          LA = FLD(5,17,NSQ2(J2))                                     *NEW
00472  107*          LK = FLD(22,14,NSQ2(J2))                                    *NEW
00473  107*          G2 = XK(LK)*XK(LA)                                          **-2
00474  107*          GO TO 2998
00475  107*     2055 TM = (T(I)+T(LTA))/2.0
00476  107*          CALL D2D1WM(TM,CON(14),A(LA),XK(LK),G(LG))
00477  107*          GO TO 2999
00500  107*     2060 TM = T(LTA)
00501  107*          GO TO 2007
00502  107*     2065 TM = T(LTA)
00503  107*          GO TO 2032
00504  107*     2998 G(LG) = 1./(1./G1+1./G2)
00505  107*          IF(FLD(3,1,NSQ1(J1)).EQ.1) G(LG) = G1*G2
00507  107*     2999 J2 = J2+1
00510  107*     3000 CONTINUE
00511  107*          END
00512  108*          T1 = T(I)+460.0
00513  109*          T2 = T(LTA)+460.0
00513  110*    C     CHECK FOR RADIATION CONDUCTOR
00514  111*          IF(FLD(3,1,NSQ1(J1)).EQ.0) GO TO 75
00516  112*          R1 = R1+G(LG)
00517  113*          QSUM = QSUM+G(LG)*T2**4
00520  114*          G2 = G2+G(LG)*(T1*T1+T2*T2)*(T1+T2)
00521  115*          GO TO 80
00522  116*       75 GV = G(LG)
00523  117*          G2 = G2+GV
00524  118*          GSUM = GSUM+GV
00525  119*          QSUM = QSUM+GV*T2
00525  120*    C     CHECK FOR LAST CONDUCTOR
00526  121*       80 IF(NSQ1(J1).GT.0) GO TO 70
00526  122*    C     DAMPEN RADIATION ON THIS NODE IF PRESENT
00530  123*          IF(R1.LE.0.) GO TO 100
00532  124*          R2 = R1*T1**4
00533  125*          T2 = (QSUM-R2)/GSUM
00534  126*          R1 = R1*T2**4
00535  127*          S = (R1+R2)/2.0
00535  128*    C     OBTAIN THE NEW TEMPERATURE
00536  129*      100 T(I) = (DN*((GSUM-S)/GSUM)+DD*T1)-460.0
00537  130*          R1 = C(I)/G2
00540  131*          IF(R1.GE.RC) GO TO 105
00542  132*          RC = R1
00543  133*          KON(35) = I
```

```
00544        105 CONTINUE
00544      C     CONVERT TEMPERATURES TO RANKINE
00546            DO 65 I = 1,NNT
00551            LE1 = IE1+I
00552            T(I) = T(I)+460.
00553         65 X(LE1) = X(LE1)+460.
00555            CON(17) = RC*TSTEPN
00556            IF(RC.LE.0.) GO TO 993
00560            GO TO 225
00560      C     NOW RELAX THE NETWORK BY SUCCESSIVE POINT AND EXTRAPOLATION
00561        110 JJ = JJ+1
00562            DO 165 I = 1,NND
00565            R1 = 0.0
00566            S = 0.0
00567            QSUM = Q(I)
00570            GSUM = C(I)
00571        115 J1 = J1+1
00572            LG = FLD(5,16,NSQ1(J1))
00573            LTA = FLD(22,14,NSQ1(J1))
00573      C     CHECK FOR RADIATION CONDUCTOR
00574            IF(FLD(3,1,NSQ1(J1)).EQ.0) GO TO 120
00576            R1 = R1+G(LG)
00577            QSUM = QSUM+G(LG)*T(LTA)**4
00600            GO TO 125
00601        120 GSUM = GSUM+G(LG)
00602            QSUM = QSUM+G(LG)*T(LTA)
00602      C     CHECK FOR LAST CONDUCTOR
00603        125 IF(NSQ1(J1).GT.0) GO TO 115
00603      C     DAMPEN RADIATION ON THIS NODE IF PRESENT
00605            IF(R1.LE.0.) GO TO 145
00607            R2 = R1*T(I)**4
00610            T2 = (QSUM-R2)/GSUM
00611            R1 = R1*T2**4
00612            S = (R1+R2)/2.0
00612      C     OBTAIN THE NEW TEMPERATURE
00613        145 T2 = DN*((QSUM-S)/GSUM)+DD*T(I)
00613      C     OBTAIN THE CALCULATED TEMPERATURE DIFFERENCE
00614            T1 = ABS(T(I)-T2)
00614      C     STORE THE NEW AND OLD TEMPERATURES
00615            GO TO (160,155,150), JJ
00616        150 LE2 = IE2+I
00617            LE3 = IE3+I
00620            R1 = T2-T(I)
00621            X(LE2) = T(I)
00622            X(LE3) = R1/(R1-X(LE3))
00623            GO TO 160
00624        155 LE3 = IE3+I
00625            X(LE3) = T2-T(I)
00626        160 T(I) = T2
00627            IF(RLXD.GE.T1) GO TO 165
00631            RLXD = T1
00632            KK1 = I
00633        165 CONTINUE
00635            GO TO (180,180,170), JJ
00635      C     PERFORM LINEAR EXTRAPOLATION ON THE ERROR FUNCTION CURVE
00636        170 JJ = 0
00637            DO 175 I = 1,NND
00642            LE3 = IE3+1
00642      C     SEE IF THE EXTRAPOLATION IS ALLOWABLE
00643            IF(X(LE3).GE.0.) GO TO 175
```

```
00643  194*   C      LIMIT THE EXTRAPOLATION
00645  195*          IF(X(LE3).LT.-10.) X(LE3) = -10.
00647  196*          LE2 = IE2+I
00650  197*          T(I) = X(LE3)*X(LE2)+(1.0-X(LE3))*T(I)
00651  198*      175 CONTINUE
00653  199*      180 IF(NNA.LE.0) GO TO 220
00655  200*          JJ1 = J1
00656  201*          JJ2 = J2
00657  202*          DO 230 I = 1,NNT
00662  203*      230 T(I) = T(I)-460.0
00664  204*          DO 215 I = NN,NNC
00667  205*          L = I
00670  206*          GSUM = 0.0
00671  207*          IF(K1.GT.2) GO TO 6000
00673  208*          INCLUDE VRQ2,LIST
00674  208*          IF(FLD(4,1,NSQ1(JJ1+1)).EQ.0) GO TO 6000
00676  208*          NTYPE = FLD(0,5,NSQ2(JJ2))                                          *NEW
00677  208*          LA = FLD(5,17,NSQ2(JJ2))                                            *NEW
00700  208*          LK = FLD(22,14,NSQ2(JJ2))                                           *NEW
00701  208*          GO TO (5005,5010,5015,5020,5025,5030,5035,5040,5030), NTYPE         **-3
00702  208*     5005 Q(L) = XK(LK)+Q(L)
00703  208*          GO TO 5999
00704  208*     5010 Q1 = 0.0
00705  208*     5012 CALL D1D1WM(T(L),A(LA),XK(LK),Q2)
00706  208*          GO TO 5998
00707  208*     5015 Q1 = 0.0
00710  208*     5017 CALL D1D1WM(CON(14),A(LA),XK(LK),Q2)
00711  208*          GO TO 5998
00712  208*     5020 CALL D1D1WM(CON(14),A(LA),XK(LK),Q1)
00713  208*     5022 JJ2 = JJ2+1
00714  208*          LA = FLD(5,17,NSQ2(JJ2))                                            *NEW
00715  208*          LK = FLD(22,14,NSQ2(JJ2))                                           *NEW
00716  208*          GO TO 5017                                                          **-2
00717  208*     5025 Q1 = XK(LK)*XK(LA)
00720  208*          GO TO 5022
00721  208*     5030 CALL D1D1WM(CON(14),A(LA),XK(LK),Q1)
00722  208*          JJ2 = JJ2+1
00723  208*          LA = FLD(5,17,NSQ2(JJ2))                                            *NEW
00724  208*          LK = FLD(22,14,NSQ2(JJ2))                                           *NEW
00725  208*          Q2 = XK(LK)*XK(LA)                                                  **-2
00726  208*          GO TO 5998
00727  208*     5035 CALL D1D1WM(CON(14),A(LA),XK(LK),G1)
00730  208*     5037 JJ2 = JJ2+1
00731  208*          LA = FLD(5,17,NSQ2(JJ2))                                            *NEW
00732  208*          LK = FLD(22,14,NSQ2(JJ2))                                           *NEW
00733  208*          GO TO 5012                                                          **-2
00734  208*     5040 Q1 = XK(LK)*XK(LA)
00735  208*          GO TO 5037
00736  208*     5998 Q(L) = Q1+Q2+Q(L)
00737  208*     5999 JJ2 = JJ2+1
00740  208*     6000 CONTINUE
00741  208*          END
00742  209*          QSUM = Q(I)
00743  210*      185 JJ1 = JJ1+1
00744  211*          LG = FLD(5,16,NSQ1(JJ1))
00745  212*          LTA = FLD(22,14,NSQ1(JJ1))
00746  213*          IF(K1.GT.2) GO TO 4000
00750  214*          INCLUDE VRG2,LIST
00751  214*          IF(FLD(2,1,NSQ1(JJ1)).EQ.0) GO TO 4000
00753  214*          NTYPE = FLD(0,5,NSQ2(JJ2))                                          *NEW
```

CNBACK,CNBACK

```
214*          LA = FLD(5,17,NSQ2(JJ2))                                        *NEW
214*          LK = FLD(22,14,NSQ2(JJ2))                                       *NEW
214*          GOTO(3005,3010,3015,3020,3025,3030,3035,3040,3045,3050,3055,    **-3
214*         $     3060,3065), NTYPE
214*     3005 TM = (T(L)+T(LTA))/2.0
214*     3007 CALL D1D1WM(TM,A(LA),XK(LK),G(LG))
214*          GO TO 3999
214*     3010 TM = T(L)
214*          GO TO 3007
214*     3015 CALL D1D1WM(T(L),A(LA),XK(LK),G1)
214*     3017 JJ2 = JJ2+1
214*          LA = FLD(5,17,NSQ2(JJ2))                                        *NEW
214*          LK = FLD(22,14,NSQ2(JJ2))                                       *NEW
214*          CALL D1D1WM(T(LTA),A(LA),XK(LK),G2)                             **-2
214*          GO TO 3998
214*     3020 G1 = XK(LK)*XK(LA)
214*          GO TO 3017
214*     3025 CALL D1D1WM(T(L),A(LA),XK(LK),G1)
214*          JJ2 = JJ2+1
214*          LA = FLD(5,17,NSQ2(JJ2))                                        *NEW
214*          LK = FLD(22,14,NSQ2(JJ2))                                       *NEW
214*          G2 = XK(LK)*XK(LA)                                              **-2
214*          GO TO 3998
214*     3030 TM = (T(L)+T(LTA))/2.0
214*     3032 CALL PLYAWM(A(LA),TM,A(LA+1),XK(LK),G(LG))
214*          GO TO 3999
214*     3035 TM = T(L)
214*          GO TO 3032
214*     3040 CALL PLYAWM(A(LA),T(L),A(LA+1),XK(LK),G1)
214*     3042 JJ2 = JJ2+1
214*          LA = FLD(5,17,NSQ2(JJ2))                                        *NEW
214*          LK = FLD(22,14,NSQ2(JJ2))                                       *NEW
214*          CALL PLYAWM(A(LA),T(LTA),A(LA+1),XK(LK),G2)                     **-2
214*          GO TO 3998
214*     3045 G1 = XK(LK)*XK(LA)
214*          GO TO 3042
214*     3050 CALL PLYAWM(A(LA),T(L),A(LA+1),XK(LK),G1)
214*          JJ2 = JJ2+1
214*          LA = FLD(5,17,NSQ2(JJ2))                                        *NEW
214*          LK = FLD(22,14,NSQ2(JJ2))                                       *NEW
214*          G2 = XK(LK)*XK(LA)                                              **-2
214*          GO TO 3998
214*     3055 TM = (T(L)+T(LTA))/2.0
214*          CALL D2D1WM(TM,CON(14),A(LA),XK(LK),G(LG))
214*          GO TO 3999
214*     3060 TM = T(LTA)
214*          GO TO 3007
214*     3065 TM = T(LTA)
214*          GO TO 3032
214*     3998 G(LG) = 1./(1./G1+1./G2)
214*          IF(FLD(3,1,NSQ1(JJ1)).EQ.1) G(LG) = G1*G2
214*     3999 JJ2 = JJ2+1
214*     4000 CONTINUE
214*          END
215*          T1 = T(I)+460.0
216*          T2 = T(LTA)+460.0
217*    C     CHECK FOR RADIATION CONDUCTOR
218*          IF(FLD(3,1,NSQ1(JJ1)).EQ.0) GO TO 190
219*          GV = G(LG)*(T1*T1+T2*T2)*(T1+T2)
220*          GO TO 195
```

```
01050        190 GV = G(LG)
01051        195 GSUM = GSUM+GV
01052            QSUM = QSUM+GV*T2
01052      C     CHECK FOR LAST CONDUCTOR
01053            IF(NSQ1(JJ1).GT.0) GO TO 185
01053      C     CALCULATE THE NEW TEMPERATURE
01055            T2 = AN*QSUM/GSUM+AA*T1
01056            T1 = ABS(T2-T1)
01057            T(I) = T2-460.0
01060            IF(RLXA.GE.T1) GO TO 215
01062            RLXA = T1
01063            KK2 = I
01064        215 CONTINUE
01066            DO 235 I = 1,NNT
01071        235 T(I) = T(I)+460.0
01071      C     SEE IF THE ARITHMETIC RELAXATION CRITERIA WAS MET
01073            IF(RLXA.GT.CON(19)) GO TO 225
01073      C     SEE IF THE DIFFUSION RELAXATION CRITERIA WAS MET
01075        220 IF(RLXD.LE.CON(26)) GO TO 245
01077        225 IF(KON(7).EQ.0) GO TO 240
01101            CALL OUTCAL
01102        240 CONTINUE
01104            IF(KON(28).GE.65) CALL TOPLIN
01106            WRITE(6,882)
01110            KON(28) = KON(28)+2
01110      C     SEE IF THE TEMPERATURE CHANGES WERE TOO LARGE
01111        245 TCGD = 0.0
01112            TCGA = 0.0
01113            DO 250 I = 1,NND
01116            LE = IE1+I
01117            C(I) = C(I)*TSTEPN
01120            T1 = ABS(T(I)-X(LE))
01121            IF(TCGD.GT.T1) GO TO 250
01123            TCGD = T1
01124            KON(36) = I
01125        250 CONTINUE
01127            IF(TCGD.LE.CON(6)) GO TO 265
01131            TSTEPN = 0.95*TSTEPN*CON(6)/TCGD
01132        255 DO 260 I = 1,NNT
01135            LE = IE1+I
01136        260 T(I) = X(LE)-460.0
01140            GO TO 30
01141        265 IF(NNA.LE.0) GO TO 275
01143            DO 270 I = NN,NNC
01146            LE = IE1+I
01147            T1 = ABS(T(I)-X(LE))
01150            IF(TCGA.GT.T1) GO TO 270
01152            TCGA = T1
01153            KON(38) = I
01154        270 CONTINUE
01156            IF(TCGA.LE.CON(11)) GO TO 275
01160            TSTEPN = 0.95*TSTEPN*CON(11)/TCGA
01161            GO TO 255
01161      C      CONVERT TEMPERATURES BACK TO FARENHEIT
01162        275 DO 280 I = 1,NNT
01165        280 T(I) = T(I)-460.0
01165      C     STORE THE TEMPERATURE AND RELAXATION CHANGES
01167            CON(15) = TCGD
01170            CON(16) = TCGA
01171            CON(27) = RLXD
```

CNBACK,CNBACK

```
01172           IF(RLXA.GT.RLXD) GO TO 285
01174           KK2 = KK1
01175           RLXA = RLXD
01176       285 KON(37) = KK2
01177           CON(30) = RLXA
01200           KON(12) = 0
01201           CALL VARBL2
01201     C     CHECK THE BACKUP SWITCH
01202           IF(KON(12).NE.0) GO TO 255
01202     C     ADVANCE TIME
01204           CON(13) = CON(1)
01205           TSUM = TSUM+TSTEPN
01205     C     CHECK FOR TIME TO PRINT
01206           IF(TSUM.GE.CON(18)) GO TO 290
01206     C     CHECK FOR PRINT EVERY ITERATION
01210           IF(KON(7).NE.0) CALL OUTCAL
01212           GO TO 10
01212     C     TRY TO EVEN THE OUTPUT INTERVALS
01213       290 TPRINT = TPRINT+TSUM
01214           CALL OUTCAL
01214     C     IS TIME GREATER THAN END COMPUTE TIME
01215           IF(CON(1)*1.000001.LT.CON(3)) GO TO 5
01217           NTH = IE1
01220           NDIM = NLA
01221           RETURN
01222       990 WRITE(6,880)
01224           GO TO 1000
01225       991 WRITE(6,881)
01227           GO TO 1000
01230       993 WRITE(6,883)
01232           GO TO 1000
01233       994 WRITE(6,884) NDIM
01236           GO TO 1000
01237       995 WRITE(6,885)
01241           GO TO 1000
01242       996 WRITE(6,886)
01244           GO TO 1000
01245       997 WRITE(6,887)
01247           GO TO 1000
01250       998 WRITE(6,888)
01252           GO TO 1000
01253       999 WRITE(6,889)
01255      1000 CALL OUTCAL
01256           CALL EXIT
01257       880 FORMAT(29H TRANSIENT TIME NOT SPECIFIED)
01260       881 FORMAT(45H CNBACK REQUIRES LONG PSEUDO-COMPUTE SEQUENCE)
01261       882 FORMAT(29H RELAXIATION CRITERIA NOT MET)
01262       883 FORMAT(24H CSGMIN ZERO OR NEGATIVE)
01263       884 FORMAT(I8,20H LOCATIONS AVAILABLE)
01264       885 FORMAT(10H NO DRLXCA)
01265       886 FORMAT(10H NO DTIMEI)
01266       887 FORMAT(10H NO ARLXCA)
01267       888 FORMAT(19H NO OUTPUT INTERVAL)
01270       889 FORMAT(9H NO NLOOP)
01271           END

  END OF UNIVAC 1108 FORTRAN V COMPILATION.      0 *DIAGNOSTIC* MESSAGE(S)
 CNBACK          SYMBOLIC                         12 JAN 70  23:06:44   0  42603742   14   335  (DELETED)
 CNBACK   CODE   RELOCATABLE                      12 JAN 70  23:06:44   1  42615064   84     1  (DELETED)
                                                                        0  42615210   14   226
```

```
CNFWBK,CNFWBK

QIW   FOR,*        CNFWBK,CNFWBK                                                          21 FEB 71
UNIVAC 1108 FORTRAN V ATHENA VERSION 131K-10D   CREATED ON 20 AUG 70
THIS COMPILATION WAS DONE ON 09 JUN 70 AT 14:00:43

   SUBROUTINE CNFWBK    ENTRY POINT 004271

   STORAGE USED (BLOCK, NAME, LENGTH)

         0001    *CODE          004307
         0000    *CONST+TEMP    000127
         0002    *SIMPLE VAR    000071
         0004    *ARRAYS        000000
         0005    *BLANK         000000
         0006    TITLE    000001
         0007    TEMP     000001
         0010    CAP      000001
         0011    SOURCE   000001
         0012    COND     000001
         0013    PC1      000001
         0014    PC2      000001
         0015    KONST    000001
         0016    ARRAY    000001
         0017    FIXCON   000001
         0020    XSPACE   000003
         0021    DIMENS   000010

   EXTERNAL REFERENCES (BLOCK, NAME)

         0022    VARBL1
         0023    OUTCAL
         0024    D1D1WM
         0025    PLYAWM
         0026    D2D1WM
         0027    TOPLIN
         0030    VARBL2
         0031    EXIT
         0032    NERR2$
         0033    NWDU$
         0034    NI02$
         0035    NER10$

   STORAGE ASSIGNMENT FOR VARIABLES (BLOCK, TYPE, RELATIVE LOCATION, NAME)

    0016 R 000000 A       0002 R 000017 AA      0002 R 000016 AN      0010 R 000000 C       0017 R 000030 CON
    0002 R 000035 C1      0002 R 000036 C2      0002 R 000015 DD      0002 R 000014 DN      0012 R 000000 G
    0002 R 000045 GSUM    0002 R 000055 GV      0002 R 000052 G1      0002 R 000041 G2      0006 R 000000 H
    0002 I 000026 I       0002 I 000003 IE1     0002 I 000005 IE2     0002 I 000006 IE3     0002 I 000007 J
    0002 I 000031 JJ      0002 I 000062 JJ1     0002 I 000063 JJ2     0002 I 000022 J1      0002 I 000025 J2
    0015 I 000000 K       0002 I 000061 KK1     0002 I 000065 KK2     0017 I 000000 KON     0002 I 000021 K1
    0002 I 000064 L       0002 I 000033 LA      0002 I 000013 LAX     0002 I 000070 LE      0002 I 000027 LE1
    0002 I 000057 LE2     0002 I 000060 LE3     0002 I 000046 LG      0002 I 000034 LK      0021 I 000006 LSQ1
    0021 I 000007 LSQ2    0002 I 000047 LTA     0002 I 000050 LTAE    0021 I 000005 NAT     0021 I 000004 NCT
    0020 I 000000 NDIM    0021 I 000003 NGT     0002 I 000002 NLA     0002 I 000001 NN      0021 I 000001 NNA
    0002 I 000004 NNC     0021 I 000000 NND     0021 I 000002 NNT     0013 I 000000 NSQ1    0014 I 000000 NSQ2
    0020 I 000001 NTH     0002 I 000032 NTYPE   0020   000002 NX      0002 R 000000 PASS    0011 R 000000 Q
```

```
CNFWBK,CNFWBK

0002 R 000044 QSUM      0002 R 000042 Q1        0002 R 000043 Q2        0002 R 000030 RC        0002 R 000023 RLXA
0002 R 000024 RLXD      0002 R 000037 R1        0002 R 000056 R2        0002 R 000040 S         0007 R 000000 T
0002 R 000067 TCGA      0002 R 000066 TCGD      0002 R 000051 TM        0002 R 000010 TPRINT    0002 R 000020 TSTEP
0002 R 000012 TSTEPN    0002 R 000011 TSUM      0002 R 000053 T1        0002 R 000054 T2        0020 R 000002 X
0015 R 000000 XK        0001   000162 10L       0001   002126 100L      0001   004244 1000L     0001   000423 1005L
0001   000444 1010L     0001   000462 1012L     0001   000515 1015L     0001   000523 1020L     0001   000563 1025L
0001   000607 1030L     0001   000630 1032L     0001   000666 1035L     0001   000674 1040L     0001   000737 1045L
0001   002153 105L      0001   003655 1073G     0001   002206 110L      0001   003726 1120G     0001   003773 1137G
0001   002224 115L      0001   004016 1150G     0001   004061 1167G     0001   002263 120L      0001   002274 125L
0001   002330 145L      0001   000171 15L       0001   002314 150L      0001   002377 155L      0001   002406 160L
0001   002421 165L      0001   002434 170L      0001   002500 175L      0001   002502 180L      0001   003021 185L
0001   003600 190L      0001   003603 195L      0001   000761 1998L     0001   000764 1999L     0001   000174 20L
0001   000767 2000L     0001   001344 2005L     0001   001351 2007L     0001   001371 2010L     0001   001374 2015L
0001   001412 2017L     0001   001446 2020L     0001   001454 2025L     0001   001514 2030L     0001   001521 2032L
0001   001544 2035L     0001   001547 2040L     0001   001570 2042L     0001   001627 2045L     0001   001635 2050L
0001   001700 2055L     0001   001726 2060L     0001   001732 2065L     0001   003647 215L      0001   003666 220L
0001   000271 221G      0001   003673 225L      0001   000313 233G      0001   000317 240G      0001   003677 240L
0001   003720 245L      0001   003753 250L      0001   003767 255L      0001   000355 261G      0001   004006 265L
0001   004040 270L      0001   004055 275L      0001   004104 285L      0001   004134 290L      0001   001736 2998L
0001   001764 2999L     0001   000206 30L       0001   001767 3000L     0001   003107 3005L     0001   003115 3007L
0001   003135 3010L     0001   003141 3015L     0001   003160 3017L     0001   003214 3020L     0001   003222 3025L
0001   003263 3030L     0001   003271 3032L     0001   003314 3035L     0001   003320 3040L     0001   003342 3042L
0001   003401 3045L     0001   003407 3050L     0001   003453 3055L     0001   003502 3060L     0001   003506 3065L
0001   000221 35L       0001   003512 3998L     0001   003540 3999L     0001   000233 40L       0001   003543 4000L
0001   001040 4005L     0001   001045 4010L     0001   001046 4012L     0001   001065 4015L     0001   001066 4017L
0001   001103 4020L     0001   001117 4022L     0001   001134 4025L     0001   001142 4030L     0001   001200 4035L
0001   001214 4037L     0001   001231 4040L     0001   000237 45L       0001   001237 4998L     0001   001243 4999L
0001   000150 5L        0001   001246 5000L     0001   002605 5005L     0001   002613 5010L     0001   002614 5012L
0001   002634 5015L     0001   002635 5017L     0001   002652 5020L     0001   002666 5022L     0001   002703 5025L
0001   002711 5030L     0001   002747 5035L     0001   002763 5037L     0001   003000 5040L     0001   002161 553G
0001   002215 567G      0001   003006 5998L     0001   003013 5999L     0001   000346 60L       0001   003016 6000L
0001   002441 644G      0001   002525 664G      0001   002533 671G      0001   001260 70L       0001   002047 75L
0001   002072 80L       0000   000000 880F      0000   000006 881F      0000   000017 882F      0000   000025 883F
0000   000032 884F      0000   000037 885F      0000   000042 886F      0000   000045 887F      0000   000050 888F
0000   000054 889F      0001   004156 990L      0001   004164 991L      0001   004172 993L      0001   004200 994L
0001   004207 995L      0001   004215 996L      0001   004223 997L      0001   004231 998L      0001   004237 999L
```

```
00101    1*         SUBROUTINE CNFWBK
00101    2*   C     IMPLICIT FORWARD-BACKWARD DIFFERENCING EXECUTION SUBROUTINE
00101    3*   C     THE LONG PSEUDO-COMPUTE SEQUENCE IS REQUIRED, SINDA  FORTRAN V
00101    4*   C     ALL NODES RECEIVE A SUCCESSIVE POINT ITERATION
00101    5*   C     RELAXATION CRITERIA MUST BE SPECIFIED
00101    6*   C     OVER-RELAXATION IS ALLOWED, THE DAMPENING FACTORS ARE ADDRESSABLE
00103    7*         INCLUDE COMM,LIST
00104    7*         COMMON /TITLE/H(1) /TEMP/T(1) /CAP/C(1) /SOURCE/Q(1) /COND/G(1)
00105    7*         COMMON /PC1/NSQ1(1) /PC2/NSQ2(1) /KONST/K(1) /ARRAY/A(1)
00106    7*         COMMON /FIXCON/KON(1) /XSPACE/NDIM,NTH,X(1)
00107    7*         COMMON /DIMENS/ NND,NNA,NNT,NGT,NCT,NAT,LSQ1,LSQ2
00110    7*         DIMENSION CON(1),XK(1),NX(1)
00111    7*         EQUIVALENCE (KON(1),CON(1)),(K(1),XK(1)),(X(1),NX(1))
00112    7*         END
00113    8*         INCLUDE DEFF,LIST
00113    8*   C*************  CONTROL CONSTANT DEFINITIONS AND NAMES  ****************
00113    8*   C     CONTROL CONSTANT 1 CONTAINS THE NEW PROBLEM TIME            (TIMEN)
00113    8*   C     CONTROL CONSTANT 2 CONTAINS THE TIME STEP USED             (DTIMEU)
00113    8*   C     CONTROL CONSTANT 3 CONTAINS THE PROBLEM STOP TIME          (TIMEND)
00113    8*   C     CONTROL CONSTANT 4 CONTAINS THE TIME STEP FACTOR,EXPLICIT (CSGFAC)
```

```
CNFWBK,CNFWBK
00113     8*   C      CC5 IS THE INPUT NUMBER OF ITERATION DO LOOPS, INTEGER      (NLOOP)
00113     8*   C      CC6 CONTAINS THE DIFFUSION TEMPERATURE CHANGE ALLOWED      (DTMPCA)
00113     8*   C      CC7 CONTAINS THE OUTPUT EACH ITERATION SWITCH              (OPEITR)
00113     8*   C      CC8 CONTAINS THE MAXIMUM ALLOWED TIME STEP                 (DTIMEH)
00113     8*   C      CC9 CONTAINS THE NEW ARITHMETIC TEMP. DAMPING FACTOR        (DAMPA)
00113     8*   C      CC10 CONTAINS THE NEW DIFFUSION TEMP. DAMPING FACTOR        (DAMPD)
00113     8*   C      CC11 CONTAINS THE MAXIMUM ALLOWED ARITHMETIC TEMP. CHANGE (ATMPCA)
00113     8*   C      CC12 CONTAINS THE BACKUP SWITCH CHECKED AFTER VARIABLES    (BACKUP)
00113     8*   C      CC13 CONTAINS THE PRESENT TIME OR PROBLEM START TIME        (TIMEO)
00113     8*   C      CC14 CONTAINS THE MEAN TIME BETWEEN AN ITERATION            (TIMEM)
00113     8*   C      CC15 CONTAINS THE DIFFUSION TEMPERATURE CHANGE CALCULATED (DTMPCC)
00113     8*   C      CC16 CONTAINS ARITHMETIC TEMPERATURE CHANGE CALCULATED     (ATMPCC)
00113     8*   C      CONTROL CONSTANT 17 IS RESERVED FOR THE C/SG MINIMUM       (CSGMIN)
00113     8*   C      CONTROL CONSTANT 18 CONTAINS THE OUTPUT INTERVAL           (OUTPUT)
00113     8*   C      CC19 CONTAINS THE ARITHMETIC RELAXATION CRITERIA ALLOWED   (ARLXCA)
00113     8*   C      CC20 CONTAINS THE NUMBER OF RELAXATION LOOPS USED,INTEGER (LOOPCT)
00113     8*   C      CC21 CONTAINS THE MINIMUM ALLOWED TIME STEP                (DTIMEL)
00113     8*   C      CC22 IS FOR THE INPUT TIME STEP IMPLICIT                   (DTIMEI)
00113     8*   C      CC23 CONTAINS THE C/SG MAXIMUM                             (CSGMAX)
00113     8*   C      CC24 CONTAINS THE C/SG RANGE ALLOWED                       (CSGRAL)
00113     8*   C      CC25 CONTAINS THE C/SG RANGE CALCULATED                    (CSGRCL)
00113     8*   C      CC26 CONTAINS THE DIFFUSION RELAXATION CRITERIA ALLOWED    (DRLXCA)
00113     8*   C      CC27 CONTAINS THE DIFFUSION RELAXATION CHANGE CALCULATED   (DRLXCC)
00113     8*   C      CC28 CONTAINS THE LINE COUNTER, INTEGER                    (LINECT)
00113     8*   C      CC29 CONTAINS THE PAGE COUNTER, INTEGER                    (PAGECT)
00113     8*   C      CC30 CONTAINS ARITHMETIC RELAXATION CHANGE CALCULATED      (ARLXCC)
00113     8*   C      CC31 IS INDICATOR, 0=THERMAL SPCS,1=THERMAL LPCS,2=GENERAL (LSPCS)
00113     8*   C      CC32 CONTAINS THE ENERGY BALANCE OF THE SYSTEM, IN - OUT   (ENGBAL)
00113     8*   C      CC33 CONTAINS THE DESIRED ENERGY BALANCE, USER INPUT       (BALENG)
00113     8*   C      CC34 CONTAINS THE NOCOPY SWITCH FOR MATRIX USERS           (NOCOPY)
00113     8*   C      CC35 CONTAINS RELATIVE NODE NUMBER OF CSGMIN
00113     8*   C      CC36 CONTAINS RELATIVE NODE NUMBER OF DTMPCC
00113     8*   C      CC37 CONTAINS RELATIVE NODE NUMBER OF ARLXCC
00113     8*   C      CC38 CONTAINS RELATIVE NODE NUMBER OF ATMPCC
00113     8*   C      CC39-40-41-42-43 CONTAIN DUMMY INTEGER CONSTANTS    (I-J-K-L-MTEST)
00113     8*   C      CC44-45-46-47-48 CONTAIN DUMMY FLOATING CONSTANTS   (R-S-T-U-VTEST)
00113     8*   C      CC49 IS THE QUASI-LINEARIZATION INTERVAL FOR CINDSM        (LAXFAC)
00113     8*   C      CC50 IS NOT USED AT PRESENT
00114     8*          END
00115     9*          IF(KON(5).LE.0) GO TO 999
00117    10*          IF(CON(6).LE.0.) CON(6) = 1.E+8
00121    11*          IF(CON(8).LE.0.) CON(8) = 1.E+8
00123    12*          IF(CON(9).LE.0.) CON(9) = 1.0
00125    13*          IF(CON(10).LE.0.) CON(10) = 1.0
00127    14*          IF(CON(11).LE.0.) CON(11) = 1.E+8
00131    15*          IF(CON(3).LE.CON(13)) GO TO 990
00133    16*          IF(CON(18).LE.0.) GO TO 998
00135    17*          IF(NNA.GT.0.AND.CON(19).LE.0.) GO TO 997
00137    18*          IF(CON(22).LE.0.) GO TO 996
00141    19*          IF(NND.GT.0.AND.CON(26).LE.0.) GO TO 995
00143    20*          IF(KON(31).NE.1) GO TO 991
00145    21*          PASS = -1.0
00146    22*          NN = NND+1
00147    23*          NLA = NDIM
00150    24*          IE1 = NTH
00151    25*          NNC = NND+NNA
00152    26*          IE2 = NTH+NNT
00153    27*          IE3 = IE2+NND
00154    28*          J =  2*NND+NNT
00155    29*          NTH = NTH+J
```

```
00156         NDIM =  NDIM-J
00156   C     CHECK FOR EXTRA LOCATIONS FOR CALCULATED NODES
00157         IF(NDIM.LT.0) GO TO 994
00161         TPRINT = CON(13)
00161   C     INITALIZE TIME SUM BETWEEN OUTPUT INTERVALS
00162       5 TSUM = 0.0
00162   C     DOES OLD TIME PLUS THE OUTPUT INTERVAL EXCEED THE STOP TIME
00163         IF(CON(13)+CON(18).GT.CON(3)) CON(18) = CON(3)-CON(13)
00163   C     DONT EXCEED IT
00165      10 TSTEPN = CON(22)
00166         IF(TSTEPN.LE.CON(8)) GO TO 20
00170      15 TSTEPN = CON(8)
00171         GO TO 35
00171   C     DOES THE TIME SUM PLUS THE TIME STEP EXCEED OUTPUT INTERVAL
00172      20 IF(TSUM+TSTEPN-CON(18)) 30,35,25
00172   C     DONT EXCEED IT
00175      25 TSTEPN = CON(18)-TSUM
00176         GO TO 35
00176   C     DOES TIME SUM PLUS TWO TIME STEPS EXCEED OUTPUT INTERVAL
00177      30 IF(TSUM+2.0*TSTEPN.LE.CON(18)) GO TO 35
00177   C     APPROACH THE OUTPUT INTERVAL GRADUALLY
00201         TSTEPN = (CON(18)-TSUM)/2.0
00201   C     STORE DELTA TIME STEP IN THE CONSTANTS
00202      35 CON(2) = TSTEPN
00202   C     CALCULATE THE NEW TIME
00203         IF(PASS.GT.0.) GO TO 40
00205         CON(1) = TPRINT
00206         CON(2) = 0.0
00207         GO TO 45
00210      40 CON(1) = TPRINT+TSUM+TSTEPN
00210   C     COMPUTE THE MEAN TIME BETWEEN ITERATIONS
00211      45 CON(14) = (CON(1)+CON(13))/2.0
00212         LAX = KON(5)
00213         DN = CON(10)
00214         DD = 1.0-DN
00215         AN = CON(9)
00216         AA = 1.0-AN
00217         TSTEP = TSTEPN/2.0
00217   C     DO THE RELAXATION LOOP
00220         DO 240 K1 = 1,LAX
00223         KON(20) = K1
00224         J1 = 0
00225         RLXA = 0.0
00226         RLXD = 0.0
00227         IF(K1.GT.1) GO TO 110
00231         J2 = 1
00231   C     ZERO OUT ALL SOURCE LOCATIONS AND SHIFT TEMPERATURES
00232         DO 50 I = 1,NNC
00235      50 Q(I) = 0.0
00237         DO 55 I = 1,NNT
00242         LE1 = IE1+I
00243      55 X(LE1) = T(I)
00245         KON(12) = 0
00246         CALL VARBL1
00246   C     CHECK THE BACKUP SWITCH
00247         IF(KON(12).NE.0) GO TO 15
00247   C     CHECK FOR FIRST PASS
00251         IF(PASS.GE.0.) GO TO 60
00253         CALL OUTCAL
00254         PASS = 1.0
```

```
      GO TO 10
   60 RC = 1.E+8
      JJ = 0
C     CALCULATE FIRST PASS TEMPERATURES AND CSGMIN
      DO 105 I = 1,NND
      INCLUDE VARC,LIST
C     FOLD DELTAT INTO THE CAPACITANCES
      IF(FLD(1,1,NSQ1(J1+1)).EQ.0) GO TO 2000
      NTYPE = FLD(0,5,NSQ2(J2))                                          *NEW
      LA = FLD(5,17,NSQ2(J2))                                            *NEW
      LK = FLD(22,14,NSQ2(J2))                                           *NEW
      GO TO (1005,1010,1015,1020,1025,1030,1035,1040,1045), NTYPE        **-3
 1005 CALL D1D1WM(T(I),A(LA),XK(LK),C(I))
      GO TO 1999
 1010 CALL D1D1WM(T(I),A(LA),XK(LK),C1)
 1012 J2 = J2+1
      LA = FLD(5,17,NSQ2(J2))                                            *NEW
      LK = FLD(22,14,NSQ2(J2))                                           *NEW
      CALL D1D1WM(T(I),A(LA),XK(LK),C2)                                  **-2
      GO TO 1998
 1015 C1 = XK(LK)*XK(LA)
      GO TO 1012
 1020 CALL D1D1WM(T(I),A(LA),XK(LK),C1)
      J2 = J2+1
      LA = FLD(5,17,NSQ2(J2))                                            *NEW
      LK = FLD(22,14,NSQ2(J2))                                           *NEW
      C2 = XK(LK)*XK(LA)                                                 **-2
      GO TO 1998
 1025 CALL PLYAWM(A(LA),T(I),A(LA+1),XK(LK),C(I))
      GO TO 1999
 1030 CALL PLYAWM(A(LA),T(I),A(LA+1),XK(LK),C1)
 1032 J2 = J2+1
      LA = FLD(5,17,NSQ2(J2))                                            *NEW
      LK = FLD(22,14,NSQ2(J2))                                           *NEW
      CALL PLYAWM(A(LA),T(I),A(LA+1),XK(LK),C2)                          **-2
      GO TO 1998
 1035 C1 = XK(LK)*XK(LA)
      GO TO 1032
 1040 CALL PLYAWM(A(LA),T(I),A(LA+1),XK(LK),C1)
      J2 = J2+1
      LA = FLD(5,17,NSQ2(J2))                                            *NEW
      LK = FLD(22,14,NSQ2(J2))                                           *NEW
      C2 = XK(LK)*XK(LA)                                                 **-2
      GO TO 1998
 1045 CALL D2D1WM(T(I),CON(14),A(LA),XK(LK),C(I))
      GO TO 1999
 1998 C(I) = C1+C2
 1999 J2 = J2+1
 2000 CONTINUE
      END
      C(I) = C(I)/TSTEP
      R1 = 0.0
      S = 0.0
      G2 = 0.0
      INCLUDE VARQ,LIST
      IF(FLD(4,1,NSQ1(J1+1)).EQ.0) GO TO 5000
      NTYPE = FLD(0,5,NSQ2(J2))                                          *NEW
      LA = FLD(5,17,NSQ2(J2))                                            *NEW
      LK = FLD(22,14,NSQ2(J2))                                           *NEW
      GO TO (4005,4010,4015,4020,4025,4030,4035,4040,4030), NTYPE        **-3
```

CNFWBK,CNFWBK

```
101*   4005 Q(I) = XK(LK)+Q(I)
101*        GO TO 4999
101*   4010 Q1 = 0.0
101*   4012 CALL D1D1WM(T(I),A(LA),XK(LK),Q2)
101*        GO TO 4998
101*   4015 Q1 = 0.0
101*   4017 CALL D1D1WM(CON(14),A(LA),XK(LK),Q2)
101*        GO TO 4998
101*   4020 CALL D1D1WM(CON(14),A(LA),XK(LK),Q1)
101*   4022 J2 = J2+1
101*        LA = FLD(5,17,NSQ2(J2))                                        *NEW
101*        LK = FLD(22,14,NSQ2(J2))                                       *NEW
101*        GO TO 4017                                                     **-2
101*   4025 Q1 = XK(LK)*XK(LA)
101*        GO TO 4022
101*   4030 CALL D1D1WM(CON(14),A(LA),XK(LK),Q1)
101*        J2 = J2+1
101*        LA = FLD(5,17,NSQ2(J2))                                        *NEW
101*        LK = FLD(22,14,NSQ2(J2))                                       *NEW
101*        Q2 = XK(LK)*XK(LA)                                             **-2
101*        GO TO 4998
101*   4035 CALL D1D1WM(CON(14),A(LA),XK(LK),Q1)
101*   4037 J2 = J2+1
101*        LA = FLD(5,17,NSQ2(J2))                                        *NEW
101*        LK = FLD(22,14,NSQ2(J2))                                       *NEW
101*        GO TO 4012                                                     **-2
101*   4040 Q1 = XK(LK)*XK(LA)
101*        GO TO 4037
101*   4998 Q(I) = Q1+Q2+Q(I)
101*   4999 J2 = J2+1
101*   5000 CONTINUE
101*        END
102*        Q(I) = 2.0*Q(I)+C(I)*(T(I)+460.0)
103*        QSUM = Q(I)
104*        GSUM = C(I)
105*     70 J1 = J1+1
106*        LG = FLD(5,16,NSQ1(J1))
107*        LTA = FLD(22,14,NSQ1(J1))
108*        LTAE = LTA+IE1
109*        INCLUDE VARG,LIST
109*        IF(FLD(2,1,NSQ1(J1)).EQ.0) GO TO 3000
109*        NTYPE = FLD(0,5,NSQ2(J2))                                      *NEW
109*        LA = FLD(5,17,NSQ2(J2))                                        *NEW
109*        LK = FLD(22,14,NSQ2(J2))                                       *NEW
109*        GOTO(2005,2010,2015,2020,2025,2030,2035,2040,2045,2050,2055,   **-3
109*      $      2060,2065), NTYPE
109*   2005 TM = (T(I)+T(LTA))/2.0
109*   2007 CALL D1D1WM(TM,A(LA),XK(LK),G(LG))
109*        GO TO 2999
109*   2010 TM = T(I)
109*        GO TO 2007
109*   2015 CALL D1D1WM(T(I),A(LA),XK(LK),G1)
109*   2017 J2 = J2+1
109*        LA = FLD(5,17,NSQ2(J2))                                        *NEW
109*        LK = FLD(22,14,NSQ2(J2))                                       *NEW
109*        CALL D1D1WM(T(LTA),A(LA),XK(LK),G2)                            **-2
109*        GO TO 2998
109*   2020 G1 = XK(LK)*XK(LA)
109*        GO TO 2017
109*   2025 CALL D1D1WM(T(I),A(LA),XK(LK),G1)
```

CNFWBK,CNFWBK

```
00447  109*        J2 = J2+1
00450  109*        LA = FLD(5,17,NSQ2(J2))                                   *NEW
00451  109*        LK = FLD(22,14,NSQ2(J2))                                  *NEW
00452  109*        G2 = XK(LK)*XK(LA)                                        **-2
00453  109*        GO TO 2998
00454  109*   2030 TM = (T(I)+T(LTA))/2.0
00455  109*   2032 CALL PLYAWM(A(LA),TM,A(LA+1),XK(LK),G(LG))
00456  109*        GO TO 2999
00457  109*   2035 TM = T(I)
00460  109*        GO TO 2032
00461  109*   2040 CALL PLYAWM(A(LA),T(I),A(LA+1),XK(LK),G1)
00462  109*   2042 J2 = J2+1
00463  109*        LA = FLD(5,17,NSQ2(J2))                                   *NEW
00464  109*        LK = FLD(22,14,NSQ2(J2))                                  *NEW
00465  109*        CALL PLYAWM(A(LA),T(LTA),A(LA+1),XK(LK),G2)               **-2
00466  109*        GO TO 2998
00467  109*   2045 G1 = XK(LK)*XK(LA)
00470  109*        GO TO 2042
00471  109*   2050 CALL PLYAWM(A(LA),T(I),A(LA+1),XK(LK),G1)
00472  109*        J2 = J2+1
00473  109*        LA = FLD(5,17,NSQ2(J2))                                   *NEW
00474  109*        LK = FLD(22,14,NSQ2(J2))                                  *NEW
00475  109*        G2 = XK(LK)*XK(LA)                                        **-2
00476  109*        GO TO 2998
00477  109*   2055 TM = (T(I)+T(LTA))/2.0
00500  109*        CALL D2D1WM(TM,CON(14),A(LA),XK(LK),G(LG))
00501  109*        GO TO 2999
00502  109*   2060 TM = T(LTA)
00503  109*        GO TO 2007
00504  109*   2065 TM = T(LTA)
00505  109*        GO TO 2032
00506  109*   2998 G(LG) = 1./(1./G1+1./G2)
00507  109*        IF(FLD(3,1,NSQ1(J1)).EQ.1) G(LG) = G1*G2
00511  109*   2999 J2 = J2+1
00512  109*   3000 CONTINUE
00513  109*        END
00514  110*        T1 = T(I)+460.0
00515  111*        T2 = T(LTA)+460.0
00515  112*   C    CHECK FOR RADIATION CONDUCTOR
00516  113*        IF(FLD(3,1,NSQ1(J1)).EQ.0) GO TO 75
00520  114*        R1 = R1+G(LG)
00521  115*        QSUM = QSUM+G(LG)*T2**4
00522  116*        G2 = G2+G(LG)*(T1*T1+T2*T2)*(T1+T2)
00523  117*        Q(I) = Q(I)+G(LG)*((X(LTAE)+460.0)**4-T1**4)
00524  118*        GO TO 80
00525  119*     75 GV = G(LG)
00526  120*        Q(I) = Q(I)+GV*(X(LTAE)-T(I))
00527  121*        G2 = G2+GV
00530  122*        GSUM = GSUM+GV
00531  123*        QSUM = QSUM+GV*T2
00531  124*   C    CHECK FOR LAST CONDUCTOR
00532  125*     80 IF(NSQ1(J1).GT.0) GO TO 70
00532  126*   C    DAMPEN RADIATION ON THIS NODE IF PRESENT
00534  127*        IF(R1.LE.0.) GO TO 100
00536  128*        R2 = R1*T1**4
00537  129*        T2 = (QSUM-R2)/GSUM
00540  130*        R1 = R1*T2**4
00541  131*        S = (R1+R2)/2.0
00541  132*   C    OBTAIN THE NEW TEMPERATURE
00542  133*    100 T(I) = (DN*((GSUM-S)/GSUM)+DD*T1)-460.0
```

CNFWBK,CNFWRK

```
00543            R1 = C(I)/G2
00544            IF(R1.GE.RC) GO TO 105
00546            RC = R1
00547            KON(35) = I
00550        105 CONTINUE
00550     C      CONVERT TEMPERATURES TO RANKINE •
00552            DO 65 I = 1,NNT
00555            LE1 = IE1+I
00556            T(I) = T(I)+460.
00557         65 X(LE1) = X(LE1)+460.
00561            CON(17) = RC*TSTEP
00562            IF(RC.LE.0.) GO TO 993
00564            GO TO 225
00564     C      NOW RELAX THE NETWORK BY SUCCESSIVE POINT AND EXTRAPOLATION
00565        110 JJ = JJ+1
00566            DO 165 I = 1,NND
00571            R1 = 0.0
00572            S = 0.0
00573            QSUM = Q(I)
00574            GSUM = C(I)
00575        115 J1 = J1+1
00576            LG = FLD(5,16,NSQ1(J1))
00577            LTA = FLD(22,14,NSQ1(J1))
00577     C      CHECK FOR RADIATION CONDUCTOR
00600            IF(FLD(3,1,NSQ1(J1)).EQ.0) GO TO 120
00602            R1 = R1+G(LG)
00603            QSUM = QSUM+G(LG)*T(LTA)**4
00604            GO TO 125
00605        120 GSUM = GSUM+G(LG)
00606            QSUM = QSUM+G(LG)*T(LTA)
00606     C      CHECK FOR LAST CONDUCTOR
00607        125 IF(NSQ1(J1).GT.0) GO TO 115
00607     C      DAMPEN RADIATION ON THIS NODE IF PRESENT
00611            IF(R1.LE.0.) GO TO 145
00613            R2 = R1*T(I)**4
00614            T2 = (QSUM-R2)/GSUM
00615            R1 = R1*T2**4
00616            S = (R1+R2)/2.0
00616     C      OBTAIN THE NEW TEMPERATURE
00617        145 T2 = DN*((QSUM-S)/GSUM)+DD*T(I)
00617     C      OBTAIN THE CALCULATED TEMPERATURE DIFFERENCE
00620            T1 = ABS(T(I)-T2)
00620     C      STORE THE NEW AND OLD TEMPERATURES
00621            GO TO (160,155,150), JJ
00622        150 LE2 = IE2+I
00623            LE3 = IE3+I
00624            R1 = T2-T(I)
00625            X(LE2) = T(I)
00626            X(LE3) = R1/(R1-X(LE3))
00627            GO TO 160
00630        155 LE3 = IE3+I
00631            X(LE3) = T2-T(I)
00632        160 T(I) = T2
00633            IF(RLXD.GE.T1) GO TO 165
00635            RLXD = T1
00636            KK1 = I
00637        165 CONTINUE
00641            GO TO (180,180,170), JJ
00641     C      PERFORM LINEAR EXTRAPOLATION ON THE ERROR FUNCTION CURVE
00642        170 JJ = 0
```

```
00643  194*          DO 175 I = 1,NND
00646  195*          LE3 = IE3+I
00646  196*    C     SEE IF THE EXTRAPOLATION IS ALLOWABLE
00647  197*          IF(X(LE3).GE.0.) GO TO 175
00647  198*    C     LIMIT THE EXTRAPOLATION
00651  199*          IF(X(LE3).LT.-10.) X(LE3) = -10.
00653  200*          LE2 = IE2+I
00654  201*          T(I) = X(LE3)*X(LE2)+(1.0-X(LE3))*T(I)
00655  202*      175 CONTINUE
00657  203*      180 IF(NNA.LE.0) GO TO 220
00661  204*          JJ1 = J1
00662  205*          JJ2 = J2
00663  206*          DO 230 I = 1,NNT
00666  207*      230 T(I) = T(I)-460.0
00670  208*          DO 215 I = NN,NNC
00673  209*          L = I
00674  210*          IF(K1.GT.2) GO TO 6000
00676  211*          INCLUDE VRO2,LIST
00677  211*          IF(FLD(4,1,NSQ1(JJ1+1)).EQ.0) GO TO 6000
00701  211*          NTYPE = FLD(0,5,NSQ2(JJ2))                                              *NEW
00702  211*          LA = FLD(5,17,NSQ2(JJ2))                                                *NEW
00703  211*          LK = FLD(22,14,NSQ2(JJ2))                                               *NEW
00704  211*          GO TO (5005,5010,5015,5020,5025,5030,5035,5040,5030), NTYPE             **-3
00705  211*     5005 Q(L) = XK(LK)+Q(L)
00706  211*          GO TO 5999
00707  211*     5010 Q1 = 0.0
00710  211*     5012 CALL D1D1WM(T(L),A(LA),XK(LK),Q2)
00711  211*          GO TO 5998
00712  211*     5015 Q1 = 0.0
00713  211*     5017 CALL D1D1WM(CON(14),A(LA),XK(LK),Q2)
00714  211*          GO TO 5998
00715  211*     5020 CALL D1D1WM(CON(14),A(LA),XK(LK),Q1)
00716  211*     5022 JJ2 = JJ2+1
00717  211*          LA = FLD(5,17,NSQ2(JJ2))                                                *NEW
00720  211*          LK = FLD(22,14,NSQ2(JJ2))                                               *NEW
00721  211*          GO TO 5017                                                              **-2
00722  211*     5025 Q1 = XK(LK)*XK(LA)
00723  211*          GO TO 5022
00724  211*     5030 CALL D1D1WM(CON(14),A(LA),XK(LK),Q1)
00725  211*          JJ2 = JJ2+1
00726  211*          LA = FLD(5,17,NSQ2(JJ2))                                                *NEW
00727  211*          LK = FLD(22,14,NSQ2(JJ2))                                               *NEW
00730  211*          Q2 = XK(LK)*XK(LA)                                                      **-2
00731  211*          GO TO 5998
00732  211*     5035 CALL D1D1WM(CON(14),A(LA),XK(LK),Q1)
00733  211*     5037 JJ2 = JJ2+1
00734  211*          LA = FLD(5,17,NSQ2(JJ2))                                                *NEW
00735  211*          LK = FLD(22,14,NSQ2(JJ2))                                               *NEW
00736  211*          GO TO 5012                                                              **-2
00737  211*     5040 Q1 = XK(LK)*XK(LA)
00740  211*          GO TO 5037
00741  211*     5998 Q(L) = Q1+Q2+Q(L)
00742  211*     5999 JJ2 = JJ2+1
00743  211*     6000 CONTINUE
00744  211*          END
00745  212*          GSUM = 0.0
00746  213*          QSUM = Q(I)
00747  214*      185 JJ1 = JJ1+1
00750  215*          LG = FLD(5,16,NSQ1(JJ1))
00751  216*          LTA = FLD(22,14,NSQ1(JJ1))
```

CNFWBK.CNFWBK

```
      IF(K1.GT.2) GO TO 4000
      INCLUDE VRG2,LIST
      IF(FLD(2,1,NSQ1(JJ1)).EQ.0) GO TO 4000
      NTYPE = FLD(0,5,NSQ2(JJ2))                                        *NEW
      LA = FLD(5,17,NSQ2(JJ2))                                          *NEW
      LK = FLD(22,14,NSQ2(JJ2))                                         *NEW
      GOTO(3005,3010,3015,3020,3025,3030,3035,3040,3045,3050,3055,      **-3
     $     3060,3065), NTYPE
 3005 TM = (T(L)+T(LTA))/2.0
 3007 CALL D1D1WM(TM,A(LA),XK(LK),G(LG))
      GO TO 3999
 3010 TM = T(L)
      GO TO 3007
 3015 CALL D1D1WM(T(L),A(LA),XK(LK),G1)
 3017 JJ2 = JJ2+1
      LA = FLD(5,17,NSQ2(JJ2))                                          *NEW
      LK = FLD(22,14,NSQ2(JJ2))                                         *NEW
      CALL D1D1WM(T(LTA),A(LA),XK(LK),G2)                               **-2
      GO TO 3998
 3020 G1 = XK(LK)*XK(LA)
      GO TO 3017
 3025 CALL D1D1WM(T(L),A(LA),XK(LK),G1)
      JJ2 = JJ2+1
      LA = FLD(5,17,NSQ2(JJ2))                                          *NEW
      LK = FLD(22,14,NSQ2(JJ2))                                         *NEW
      G2 = XK(LK)*XK(LA)                                                **-2
      GO TO 3998
 3030 TM = (T(L)+T(LTA))/2.0
 3032 CALL PLYAWM(A(LA),TM,A(LA+1),XK(LK),G(LG))
      GO TO 3999
 3035 TM = T(L)
      GO TO 3032
 3040 CALL PLYAWM(A(LA),T(L),A(LA+1),XK(LK),G1)
 3042 JJ2 = JJ2+1
      LA = FLD(5,17,NSQ2(JJ2))                                          *NEW
      LK = FLD(22,14,NSQ2(JJ2))                                         *NEW
      CALL PLYAWM(A(LA),T(LTA),A(LA+1),XK(LK),G2)                       **-2
      GO TO 3998
 3045 G1 = XK(LK)*XK(LA)
      GO TO 3042
 3050 CALL PLYAWM(A(LA),T(L),A(LA+1),XK(LK),G1)
      JJ2 = JJ2+1
      LA = FLD(5,17,NSQ2(JJ2))                                          *NEW
      LK = FLD(22,14,NSQ2(JJ2))                                         *NEW
      G2 = XK(LK)*XK(LA)                                                **-2
      GO TO 3998
 3055 TM = (T(L)+T(LTA))/2.0
      CALL D2D1WM(TM,CON(14),A(LA),XK(LK),G(LG))
      GO TO 3999
 3060 TM = T(LTA)
      GO TO 3007
 3065 TM = T(LTA)
      GO TO 3032
 3998 G(LG) = 1./(1./G1+1./G2)
      IF(FLD(3,1,NSQ1(JJ1)).EQ.1) G(LG) = G1*G2
 3999 JJ2 = JJ2+1
 4000 CONTINUE
      END
      T1 = T(I)+460.0
      T2 = T(LTA)+460.0
```

```
01047  C      CHECK FOR RADIATION CONDUCTOR
01050         IF(FLD(3,1,NSQ1(JJ1)).EQ.0) GO TO 190
01052         GV = G(LG)*(T1*T1+T2*T2)*(T1+T2)
01053         GO TO 195
01054     190 GV = G(LG)
01055     195 GSUM = GSUM+GV
01056         QSUM = QSUM+GV*T2
01056  C      CHECK FOR LAST CONDUCTOR
01057         IF(NSQ1(JJ1).GT.0) GO TO 185
01057  C      CALCULATE THE NEW TEMPERATURE
01061         T2 = AN*QSUM/GSUM+AA*T1
01062         T1 = ABS(T2-T1)
01063         T(I) = T2-460.0
01064         IF(RLXA.GE.T1) GO TO 215
01066         RLXA = T1
01067         KK2 = I
01070     215 CONTINUE
01072         DO 235 I = 1,NNT
01075     235 T(I) = T(I)+460.0
01075  C      SEE IF THE ARITHMETIC RELAXATION CRITERIA WAS MET
01077         IF(RLXA.GT.CON(19)) GO TO 225
01077  C      SEE IF THE DIFFUSION RELAXATION CRITERIA WAS MET
01101     220 IF(RLXD.LE.CON(26)) GO TO 245
01103     225 IF(KON(7).EQ.0) GO TO 240
01105         CALL OUTCAL
01106     240 CONTINUE
01110         IF(KON(28).GE.65) CALL TOPLIN
01112         WRITE(6,882)
01114         KON(28) = KON(28)+2
01114  C      SEE IF THE TEMPERATURE CHANGES WERE TOO LARGE
01115     245 TCGD = 0.0
01116         TCGA = 0.0
01117         DO 250 I = 1,NND
01122         LE = IE1+I
01123         C(I) = C(I)*TSTEP
01124         T1 = ABS(T(I)-X(LE))
01125         IF(TCGD.GT.T1) GO TO 250
01127         TCGD = T1
01130         KON(36) = I
01131     250 CONTINUE
01133         IF(TCGD.LE.CON(6)) GO TO 265
01135         TSTEPN = 0.95*TSTEPN*CON(6)/TCGD
01136     255 DO 260 I = 1,NNT
01141         LE = IE1+I
01142     260 T(I) = X(LE)-460.0
01144         GO TO 30
01145     265 IF(NNA.LE.0) GO TO 275
01147         DO 270 I = NN,NNC
01152         LE = IE1+I
01153         T1 = ABS(T(I)-X(LE))
01154         IF(TCGA.GT.T1) GO TO 270
01156         TCGA = T1
01157         KON(38) = I
01160     270 CONTINUE
01162         IF(TCGA.LE.CON(11)) GO TO 275
01164         TSTEPN = 0.95*TSTEPN*CON(11)/TCGA
01165         GO TO 255
01165  C       CONVERT TEMPERATURES BACK TO FARENHEIT
01166     275 DO 280 I = 1,NNT
01171     280 T(I) = T(I)-460.0
```

CNFWBK.CNFWBK

```
C     STORE THE TEMPERATURE AND RELAXATION CHANGES
      CON(15) = TCGD
      CON(16) = TCGA
      CON(27) = RLXD
      IF(RLXA.GT.RLXD) GO TO 285
      KK2 = KK1
      RLXA = RLXD
  285 KON(37) = KK2
      CON(30) = RLXA
      KON(12) = 0
      CALL VARBL2
C     CHECK THE BACKUP SWITCH
      IF(KON(12).NE.0) GO TO 255
C     ADVANCE TIME
      CON(13) = CON(1)
      TSUM = TSUM+TSTEPN
C     CHECK FOR TIME TO PRINT
      IF(TSUM.GE.CON(18)) GO TO 290
C     CHECK FOR PRINT EVERY ITERATION
      IF(KON(7).NE.0) CALL OUTCAL
      GO TO 10
C     TRY TO EVEN THE OUTPUT INTERVALS
  290 TPRINT = TPRINT+TSUM
      CALL OUTCAL
C     IS TIME GREATER THAN END COMPUTE TIME
      IF(CON(1)*1.000001.LT.CON(3)) GO TO 5
      NTH = IE1
      NDIM = NLA
      RETURN
  990 WRITE(6,880)
      GO TO 1000
  991 WRITE(6,881)
      GO TO 1000
  993 WRITE(6,883)
      GO TO 1000
  994 WRITE(6,884) NDIM
      GO TO 1000
  995 WRITE(6,885)
      GO TO 1000
  996 WRITE(6,886)
      GO TO 1000
  997 WRITE(6,887)
      GO TO 1000
  998 WRITE(6,888)
      GO TO 1000
  999 WRITE(6,889)
 1000 CALL OUTCAL
      CALL EXIT
  880 FORMAT(29H TRANSIENT TIME NOT SPECIFIED)
  881 FORMAT(45H CNFWBK REQUIRES LONG PSEUDO-COMPUTE SEQUENCE)
  882 FORMAT(28H RELAXATION CRITERIA NOT MET)
  883 FORMAT(24H CSGMIN ZERO OR NEGATIVE)
  884 FORMAT(I8,20H LOCATIONS AVAILABLE)
  885 FORMAT(10H NO DRLXCA)
  886 FORMAT(10H NO DTIMEI)
  887 FORMAT(10H NO ARLXCA)
  888 FORMAT(19H NO OUTPUT INTERVAL)
  889 FORMAT(9H NO NLOOP)
      END
```

```
QIW   FOR,*        CNVARB,CNVARB                                            21 FEB 71
UNIVAC 1108 FORTRAN V ATHENA VERSION 131K-10D   CREATED ON 20 AUG 70
THIS COMPILATION WAS DONE ON 09 JUN 70 AT 14:00:51

   SUBROUTINE CNVARB    ENTRY POINT 004272

   STORAGE USED (BLOCK, NAME, LENGTH)

         0001    *CODE            004311
         0000    *CONST+TEMP      000131
         0002    *SIMPLE VAR      000072
         0004    *ARRAYS          000000
         0005    *BLANK           000000
         0006    TITLE    000001
         0007    TEMP     000001
         0010    CAP      000001
         0011    SOURCE   000001
         0012    COND     000001
         0013    PC1      000001
         0014    PC2      000001
         0015    KONST    000001
         0016    ARRAY    000001
         0017    FIXCON   000001
         0020    XSPACE   000003
         0021    DIMENS   000010


   EXTERNAL REFERENCES (BLOCK, NAME)

         0022    VARBL1
         0023    OUTCAL
         0024    D1D1WM
         0025    PLYAWM
         0026    D2D1WM
         0027    TOPLIN
         0030    VARBL2
         0031    EXIT
         0032    NERR2$
         0033    NWDU$
         0034    NIO2$
         0035    NER10$


   STORAGE ASSIGNMENT FOR VARIABLES (BLOCK, TYPE, RELATIVE LOCATION, NAME)

    0016 R 000000 A        0002 R 000017 AA      0002 R 000016 AN      0002 R 000054 BETAN   0002 R 000053 BETAO
    0010 R 000000 C        0017 R 000000 CON     0002 R 000035 C1      0002 R 000036 C2      0002 R 000015 DD
    0002 R 000014 DN       0012 R 000000 G       0002 R 000057 GSUM    0002 R 000047 GV      0002 R 000046 G1
    0002 R 000042 G2       0006 R 000000 H       0002 I 000026 I       0002 I 000004 IE1     0002 I 000005 IE2
    0002 I 000006 IE3      0002 I 000007 J       0002 I 000031 JJ      0002 I 000063 JJ1     0002 I 000064 JJ2
    0002 I 000022 J1       0002 I 000025 J2      0015 I 000000 K       0002 I 000062 KK1     0002 I 000066 KK2
    0017 I 000000 KON      0002 I 000021 K1      0002 I 000065 L       0002 I 000033 LA      0002 I 000013 LAX
    0002 I 000071 LE       0002 I 000027 LE1     0002 I 000037 LE2     0002 I 000061 LE3     0002 I 000043 LG
    0002 I 000034 LK       0021 I 000006 LSQ1    0021 I 000007 LSQ2    0002 I 000044 LTA     0021 I 000005 NAT
    0021 I 000004 NCT      0020 I 000000 NDIM    0021 I 000003 NGT     0002 I 000001 NLA     0002 I 000002 NN
    0021 I 000001 NHA      0002 I 000003 NNC     0021 I 000000 NND     0021 I 000002 NNT     0013 I 000000 NSQ1
    0014 I 000000 NSQ2     0020 I 000001 NTH     0002 I 000032 NTYPE   0020   000002 NX      0002 R 000000 PASS
```

CNVARB,CNVARB

```
0011 R 000000 Q        0002 R 000056 QSUM     0002 R 000040 Q1       0002 R 000041 Q2       0002 R 000030 RC
0002 R 000023 RLXA     0002 R 000024 RLXD     0002 R 000052 R1       0002 R 000060 R2       0002 R 000055 S
0007 R 000000 T        0002 R 000070 TCGA     0002 R 000067 TCGD     0002 R 000045 TM       0002 R 000010 TPRINT
0002 R 000020 TSTEP    0002 R 000012 TSTEPN   0002 R 000011 TSUM     0002 R 000050 T1       0002 R 000051 T2
0020 R 000002 X        0015 R 000000 XK       0001   000162 10L      0001   002075 100L     0001   004244 1000L
0001   000427 1005L    0001   000450 1010L    0001   000466 1012L    0001   000521 1015L    0001   000527 1020L
0001   000567 1025L    0001   000613 1030L    0001   000634 1032L    0001   000672 1035L    0001   000700 1040L
0001   000743 1045L    0001   002150 105L     0001   003655 1073G    0001   002201 110L     0001   003726 1120G
0001   003773 1137G    0001   002205 115L     0001   004016 1150G    0001   004061 1167G    0001   002246 120L
0001   002256 125L     0001   002325 145L     0001   000171 15L      0001   002351 150L     0001   002374 155L
0001   002403 160L     0001   002416 165L     0001   002431 170L     0001   002500 175L     0001   002502 180L
0001   003021 185L     0001   003600 190L     0001   003603 195L     0001   000765 1998L    0001   000770 1999L
0001   000174 20L      0001   000773 2000L    0001   001346 2005L    0001   001353 2007L    0001   001373 2010L
0001   001376 2015L    0001   001414 2017L    0001   001450 2020L    0001   001456 2025L    0001   001516 2030L
0001   001523 2032L    0001   001546 2035L    0001   001551 2040L    0001   001572 2042L    0001   001631 2045L
0001   001637 2050L    0001   001702 2055L    0001   001730 2060L    0001   001734 2065L    0001   003647 215L
0001   003666 220L     0001   000274 221G     0001   003673 225L     0001   000316 233G     0001   000322 240G
0001   003677 240L     0001   003720 245L     0001   003753 250L     0001   003767 255L     0001   000361 261G
0001   004006 265L     0001   004040 270L     0001   004055 275L     0001   004104 285L     0001   004134 290L
0001   001740 2998L    0001   001766 2999L    0001   000206 30L      0001   001771 3000L    0001   003107 3005L
0001   003115 3007L    0001   003135 3010L    0001   003141 3015L    0001   003160 3017L    0001   003214 3020L
0001   003222 3025L    0001   003263 3030L    0001   003271 3032L    0001   003314 3035L    0001   003320 3040L
0001   003342 3042L    0001   003401 3045L    0001   003407 3050L    0001   003453 3055L    0001   003502 3060L
0001   003506 3065L    0001   000221 35L      0001   003512 3998L    0001   003540 3999L    0001   000233 40L
0001   003543 4000L    0001   001046 4005L    0001   001053 4010L    0001   001054 4012L    0001   001073 4015L
0001   001074 4017L    0001   001111 4020L    0001   001125 4022L    0001   001142 4025L    0001   001150 4030L
0001   001206 4035L    0001   001222 4037L    0001   001237 4040L    0001   000237 45L      0001   001245 4998L
0001   001251 4999L    0001   000150 5L       0001   001254 5000L    0001   002604 5005L    0001   002612 5010L
0001   002613 5012L    0001   002633 5015L    0001   002634 5017L    0001   002651 5020L    0001   002665 5022L
0001   002702 5025L    0001   002710 5030L    0001   002746 5035L    0001   002762 5037L    0001   002777 5040L
0001   002132 544G     0001   002166 555G     0001   003005 5998L    0001   003012 5999L    0001   000351 60L
0001   003015 6000L    0001   002441 644G     0001   002525 664G     0001   002533 671G     0001   001264 70L
0001   002041 75L      0001   002052 80L      0000   000000 880F     0000   000006 881F     0000   000017 882F
0000   000025 883F     0000   000032 884F     0000   000037 885F     0000   000042 886F     0000   000045 887F
0000   000050 888F     0000   000054 889F     0001   004156 990L     0001   004164 991L     0001   004172 993L
0001   004200 994L     0001   004207 995L     0001   004215 996L     0001   004223 997L     0001   004231 998L
0001   004237 999L
```

```
00101    1*        SUBROUTINE CNVARB
00101    2*   C    IMPLICIT FORWARD-BACKWARD DIFFERENCING EXECUTION SUBROUTINE
00101    3*   C    WITH AN INTERNALLY CALCULATED BETA WEIGHTING FACTOR
00101    4*   C    THE LONG PSEUDO-COMPUTE SEQUENCE IS REQUIRED, SINDA  FORTRAN V
00101    5*   C    ALL NODES RECEIVE A SUCCESSIVE POINT ITERATION
00101    6*   C    RELAXATION CRITERIA MUST BE SPECIFIED
00101    7*   C    OVER-RELAXATION IS ALLOWED, THE DAMPENING FACTORS ARE ADDRESSABLE
00103    8*        INCLUDE COMM,LIST
00104    8*        COMMON /TITLE/H(1) /TEMP/T(1) /CAP/C(1) /SOURCE/Q(1) /COND/G(1)
00105    8*        COMMON /PC1/NSQ1(1) /PC2/NSQ2(1) /KONST/K(1) /ARRAY/A(1)
00106    8*        COMMON /FIXCON/KON(1) /XSPACE/NDIM,NTH,X(1)
00107    8*        COMMON /DIMENS/ NND,NNA,NNT,NGT,NCT,NAT,LSQ1,LSQ2
00110    8*        DIMENSION CON(1),XK(1),NX(1)
00111    8*        EQUIVALENCE (KON(1),CON(1)),(K(1),XK(1)),(X(1),NX(1))
00112    8*        END
00113    9*        INCLUDE DEFF,LIST
00113    9*   C*************  CONTROL CONSTANT DEFINITIONS AND NAMES  ****************
00113    9*   C    CONTROL CONSTANT 1 CONTAINS THE NEW PROBLEM TIME              (TIMEN)
00113    9*   C    CONTROL CONSTANT 2 CONTAINS THE TIME STEP USED               (DTIMEU)
```

```
00113    9*   C     CONTROL CONSTANT 3 CONTAINS THE PROBLEM STOP TIME          (TIMEND)
00113    9*   C     CONTROL CONSTANT 4 CONTAINS THE TIME STEP FACTOR,EXPLICIT  (CSGFAC)
00113    9*   C     CC5 IS THE INPUT NUMBER OF ITERATION DO LOOPS, INTEGER      (NLOOP)
00113    9*   C     CC6 CONTAINS THE DIFFUSION TEMPERATURE CHANGE ALLOWED      (DTMPCA)
00113    9*   C     CC7 CONTAINS THE OUTPUT EACH ITERATION SWITCH              (OPEITR)
00113    9*   C     CC8 CONTAINS THE MAXIMUM ALLOWED TIME STEP                 (DTIMEH)
00113    9*   C     CC9 CONTAINS THE NEW ARITHMETIC TEMP. DAMPING FACTOR        (DAMPA)
00113    9*   C     CC10 CONTAINS THE NEW DIFFUSION TEMP. DAMPING FACTOR        (DAMPD)
00113    9*   C     CC11 CONTAINS THE MAXIMUM ALLOWED ARITHMETIC TEMP. CHANGE  (ATMPCA)
00113    9*   C     CC12 CONTAINS THE BACKUP SWITCH CHECKED AFTER VARIABLES    (BACKUP)
00113    9*   C     CC13 CONTAINS THE PRESENT TIME OR PROBLEM START TIME        (TIMEO)
00113    9*   C     CC14 CONTAINS THE MEAN TIME BETWEEN AN ITERATION            (TIMEM)
00113    9*   C     CC15 CONTAINS THE DIFFUSION TEMPERATURE CHANGE CALCULATED  (DTMPCC)
00113    9*   C     CC16 CONTAINS ARITHMETIC TEMPERATURE CHANGE CALCULATED     (ATMPCC)
00113    9*   C     CONTROL CONSTANT 17 IS RESERVED FOR THE C/SG MINIMUM       (CSGMIN)
00113    9*   C     CONTROL CONSTANT 18 CONTAINS THE OUTPUT INTERVAL           (OUTPUT)
00113    9*   C     CC19 CONTAINS THE ARITHMETIC RELAXATION CRITERIA ALLOWED   (ARLXCA)
00113    9*   C     CC20 CONTAINS THE NUMBER OF RELAXATION LOOPS USED,INTEGER  (LOOPCT)
00113    9*   C     CC21 CONTAINS THE MINIMUM ALLOWED TIME STEP                (DTIMEL)
00113    9*   C     CC22 IS FOR THE INPUT TIME STEP IMPLICIT                   (DTIMEI)
00113    9*   C     CC23 CONTAINS THE C/SG MAXIMUM                             (CSGMAX)
00113    9*   C     CC24 CONTAINS THE C/SG RANGE ALLOWED                       (CSGRAL)
00113    9*   C     CC25 CONTAINS THE C/SG RANGE CALCULATED                    (CSGRCL)
00113    9*   C     CC26 CONTAINS THE DIFFUSION RELAXATION CRITERIA ALLOWED    (DRLXCA)
00113    9*   C     CC27 CONTAINS THE DIFFUSION RELAXATION CHANGE CALCULATED   (DRLXCC)
00113    9*   C     CC28 CONTAINS THE LINE COUNTER, INTEGER                    (LINECT)
00113    9*   C     CC29 CONTAINS THE PAGE COUNTER, INTEGER                    (PAGECT)
00113    9*   C     CC30 CONTAINS ARITHMETIC RELAXATION CHANGE CALCULATED      (ARLXCC)
00113    9*   C     CC31 IS INDICATOR, 0=THERMAL SPCS,1=THERMAL LPCS,2=GENERAL  (LSPCS)
00113    9*   C     CC32 CONTAINS THE ENERGY BALANCE OF THE SYSTEM, IN - OUT   (ENGBAL)
00113    9*   C     CC33 CONTAINS THE DESIRED ENERGY BALANCE, USER INPUT       (BALENG)
00113    9*   C     CC34 CONTAINS THE NOCOPY SWITCH FOR MATRIX USERS           (NOCOPY)
00113    9*   C     CC35 CONTAINS RELATIVE NODE NUMBER OF CSGMIN
00113    9*   C     CC36 CONTAINS RELATIVE NODE NUMBER OF DTMPCC
00113    9*   C     CC37 CONTAINS RELATIVE NODE NUMBER OF ARLXCC
00113    9*   C     CC38 CONTAINS RELATIVE NODE NUMBER OF ATMPCC
00113    9*   C     CC39-40-41-42-43 CONTAIN DUMMY INTEGER CONSTANTS   (I-J-K-L-MTEST)
00113    9*   C     CC44-45-46-47-48 CONTAIN DUMMY FLOATING CONSTANTS  (R-S-T-U-VTEST)
00113    9*   C     CC49 IS THE QUASI-LINEARIZATION INTERVAL FOR CINDSM        (LAXFAC)
00113    9*   C     CC50 IS NOT USED AT PRESENT
00114    9*         END
00115   10*         IF(KON(5).LE.0) GO TO 999
00117   11*         IF(CON(6).LE.0.) CON(6) = 1.E+8
00121   12*         IF(CON(8).LE.0.) CON(8) = 1.E+8
00123   13*         IF(CON(9).LE.0.) CON(9) = 1.0
00125   14*         IF(CON(10).LE.0.) CON(10) = 1.0
00127   15*         IF(CON(11).LE.0.) CON(11) = 1.E+8
00131   16*         IF(CON(3).LE.CON(13)) GO TO 990
00133   17*         IF(CON(18).LE.0.) GO TO 998
00135   18*         IF(NNA.GT.0.AND.CON(19).LE.0.) GO TO 997
00137   19*         IF(CON(22).LE.0.) GO TO 996
00141   20*         IF(NND.GT.0.AND.CON(26).LE.0.) GO TO 995
00143   21*         IF(KON(31).NE.1) GO TO 991
00145   22*         PASS = -1.0
00146   23*         NLA = NDIM
00147   24*         NN = NND+1
00150   25*         NNC = NND+NNA
00151   26*         IE1 = NTH
00152   27*         IE2 = NTH+NNT
00153   28*         IE3 = IE2+NND
```

```
00154              J =  2*NND+NNT
00155              NTH = NTH+J
00156              NDIM =  NDIM-J
00156      C       CHECK FOR EXTRA LOCATIONS FOR CALCULATED NODES
00157              IF(NDIM.LT.0) GO TO 994
00161              TPRINT = CON(13)
00161      C       INITALIZE TIME SUM BETWEEN OUTPUT INTERVALS
00162            5 TSUM = 0.0
00162      C       DOES OLD TIME PLUS THE OUTPUT INTERVAL EXCEED THE STOP TIME
00163              IF(CON(13)+CON(18).GT.CON(3)) CON(18) = CON(3)-CON(13)
00163      C       DONT EXCEED IT
00165           10 TSTEPN = CON(22)
00166              IF(TSTEPN.LE.CON(8)) GO TO 20
00170           15 TSTEPN = CON(8)
00171              GO TO 35
00171      C       DOES THE TIME SUM PLUS THE TIME STEP EXCEED OUTPUT INTERVAL
00172           20 IF(TSUM+TSTEPN-CON(18)) 30,35,25
00172      C       DONT EXCEED IT
00175           25 TSTEPN = CON(18)-TSUM
00176              GO TO 35
00176      C       DOES TIME SUM PLUS TWO TIME STEPS EXCEED OUTPUT INTERVAL
00177           30 IF(TSUM+2.0*TSTEPN.LE.CON(18)) GO TO 35
00177      C       APPROACH THE OUTPUT INTERVAL GRADUALLY
00201              TSTEPN = (CON(18)-TSUM)/2.0
00201      C       STORE DELTA TIME STEP IN THE CONSTANTS
00202           35 CON(2) = TSTEPN
00202      C       CALCULATE THE NEW TIME
00203              IF(PASS.GT.0.) GO TO 40
00205              CON(1) = TPRINT
00206              CON(2) = 0.0
00207              GO TO 45
00210           40 CON(1) = TPRINT+TSUM+TSTEPN
00210      C       COMPUTE THE MEAN TIME BETWEEN ITERATIONS
00211           45 CON(14) = (CON(1)+CON(13))/2.0
00212              LAX = KON(5)
00213              DN = CON(10)
00214              DD = 1.0-DN
00215              AN = CON(9)
00216              AA = 1.0-AN
00217              TSTEP = TSTEPN/2.0
00217      C       DO THE RELAXATION LOOP
00220              DO 240 K1 = 1,LAX
00223              KON(20) = K1
00224              J1 = 0
00225              RLXA = 0.0
00226              RLXD = 0.0
00227              IF(K1.GT.1) GO TO 105
00231              J2 = 1
00231      C       ZERO OUT ALL SOURCE LOCATIONS AND SHIFT TEMPERATURES
00232              DO 50 I = 1,NNC
00235           50 Q(I) = 0.0
00237              DO 55 I = 1,NNT
00242              LE1 = IE1+I
00243           55 X(LE1) = T(I)
00245              CON(12) = 0.0
00246              CALL VARBL1
00246      C       CHECK THE BACKUP SWITCH
00247              IF(KON(12).NE.0) GO TO 15
00247      C       CHECK FOR FIRST PASS
00251              IF(PASS.GE.0.) GO TO 60
```

```
      CALL OUTCAL
      PASS = 1.0
      GO TO 10
   60 RC = 1.E+8
      JJ = 0
C     CALCULATE THE OLD CONTRIBUTION AND THE CSGMIN
      DO 100 I = 1,NND
      INCLUDE VARC,LIST
C     FOLD DELTAT INTO THE CAPACITANCES
      IF(FLD(1,1,NSQ1(J1+1)).EQ.0) GO TO 2000
      NTYPE = FLD(0,5,NSQ2(J2))                                          *NEW
      LA = FLD(5,17,NSQ2(J2))                                            *NEW
      LK = FLD(22,14,NSQ2(J2))                                           *NEW
      GO TO (1005,1010,1015,1020,1025,1030,1035,1040,1045), NTYPE        **-3
 1005 CALL D1D1WM(T(I),A(LA),XK(LK),C(I))
      GO TO 1999
 1010 CALL D1D1WM(T(I),A(LA),XK(LK),C1)
 1012 J2 = J2+1
      LA = FLD(5,17,NSQ2(J2))                                            *NEW
      LK = FLD(22,14,NSQ2(J2))                                           *NEW
      CALL D1D1WM(T(I),A(LA),XK(LK),C2)                                  **-2
      GO TO 1998
 1015 C1 = XK(LK)*XK(LA)
      GO TO 1012
 1020 CALL D1D1WM(T(I),A(LA),XK(LK),C1)
      J2 = J2+1
      LA = FLD(5,17,NSQ2(J2))                                            *NEW
      LK = FLD(22,14,NSQ2(J2))                                           *NEW
      C2 = XK(LK)*XK(LA)                                                 **-2
      GO TO 1998
 1025 CALL PLYAWM(A(LA),T(I),A(LA+1),XK(LK),C(I))
      GO TO 1999
 1030 CALL PLYAWM(A(LA),T(I),A(LA+1),XK(LK),C1)
 1032 J2 = J2+1
      LA = FLD(5,17,NSQ2(J2))                                            *NEW
      LK = FLD(22,14,NSQ2(J2))                                           *NEW
      CALL PLYAWM(A(LA),T(I),A(LA+1),XK(LK),C2)                          **-2
      GO TO 1998
 1035 C1 = XK(LK)*XK(LA)
      GO TO 1032
 1040 CALL PLYAWM(A(LA),T(I),A(LA+1),XK(LK),C1)
      J2 = J2+1
      LA = FLD(5,17,NSQ2(J2))                                            *NEW
      LK = FLD(22,14,NSQ2(J2))                                           *NEW
      C2 = XK(LK)*XK(LA)                                                 **-2
      GO TO 1998
 1045 CALL D2D1WM(T(I),CON(14),A(LA),XK(LK),C(I))
      GO TO 1999
 1998 C(I) = C1+C2
 1999 J2 = J2+1
 2000 CONTINUE
      END
      C(I) = C(I)/TSTEP
      LE2 = IE2+I
      X(LE2) = 0.0
      INCLUDE VARQ,LIST
      IF(FLD(4,1,NSQ1(J1+1)).EQ.0) GO TO 5000
      NTYPE = FLD(0,5,NSQ2(J2))                                          *NEW
      LA = FLD(5,17,NSQ2(J2))                                            *NEW
      LK = FLD(22,14,NSQ2(J2))                                           *NEW
```

CNVARB,CNVARB

```
00351  101*       GO TO (4005,4010,4015,4020,4025,4030,4035,4040,4030), NTYPE        **-3
00352  101*  4005 Q(I) = XK(LK)+Q(I)
00353  101*       GO TO 4999
00354  101*  4010 Q1 = 0.0
00355  101*  4012 CALL D1D1WM(T(I),A(LA),XK(LK),Q2)
00356  101*       GO TO 4998
00357  101*  4015 Q1 = 0.0
00360  101*  4017 CALL D1D1WM(CON(14),A(LA),XK(LK),Q2)
00361  101*       GO TO 4998
00362  101*  4020 CALL D1D1WM(CON(14),A(LA),XK(LK),Q1)
00363  101*  4022 J2 = J2+1
00364  101*       LA = FLD(5,17,NSQ2(J2))                                            *NEW
00365  101*       LK = FLD(22,14,NSQ2(J2))                                           *NEW
00366  101*       GO TO 4017                                                         **-2
00367  101*  4025 Q1 = XK(LK)*XK(LA)
00370  101*       GO TO 4022
00371  101*  4030 CALL D1D1WM(CON(14),A(LA),XK(LK),Q1)
00372  101*       J2 = J2+1
00373  101*       LA = FLD(5,17,NSQ2(J2))                                            *NEW
00374  101*       LK = FLD(22,14,NSQ2(J2))                                           *NEW
00375  101*       Q2 = XK(LK)*XK(LA)                                                 **-2
00376  101*       GO TO 4998
00377  101*  4035 CALL D1D1WM(CON(14),A(LA),XK(LK),Q1)
00400  101*  4037 J2 = J2+1
00401  101*       LA = FLD(5,17,NSQ2(J2))                                            *NEW
00402  101*       LK = FLD(22,14,NSQ2(J2))                                           *NEW
00403  101*       GO TO 4012                                                         **-2
00404  101*  4040 Q1 = XK(LK)*XK(LA)
00405  101*       GO TO 4037
00406  101*  4998 Q(I) = Q1+Q2+Q(I)
00407  101*  4999 J2 = J2+1
00410  101*  5000 CONTINUE
00411  101*       END
00412  102*       Q(I) = 2.0*Q(I)+C(I)*(T(I)+460.0)
00413  103*       G2 = 0.0
00414  104*    70 J1 = J1+1
00415  105*       LG = FLD(5,16,NSQ1(J1))
00416  106*       LTA = FLD(22,14,NSQ1(J1))
00417  107*       INCLUDE VARG,LIST
00420  107*       IF(FLD(2,1,NSQ1(J1)).EQ.0) GO TO 3000
00422  107*       NTYPE = FLD(0,5,NSQ2(J2))                                          *NEW
00423  107*       LA = FLD(5,17,NSQ2(J2))                                            *NEW
00424  107*       LK = FLD(22,14,NSQ2(J2))                                           *NEW
00425  107*       GOTO(2005,2010,2015,2020,2025,2030,2035,2040,2045,2050,2055,       **-3
00425  107*      $     2060,2065), NTYPE
00426  107*  2005 TM = (T(I)+T(LTA))/2.0
00427  107*  2007 CALL D1D1WM(TM,A(LA),XK(LK),G(LG))
00430  107*       GO TO 2999
00431  107*  2010 TM = T(I)
00432  107*       GO TO 2007
00433  107*  2015 CALL D1D1WM(T(I),A(LA),XK(LK),G1)
00434  107*  2017 J2 = J2+1
00435  107*       LA = FLD(5,17,NSQ2(J2))                                            *NEW
00436  107*       LK = FLD(22,14,NSQ2(J2))                                           *NEW
00437  107*       CALL D1D1WM(T(LTA),A(LA),XK(LK),G2)                                **-2
00440  107*       GO TO 2998
00441  107*  2020 G1 = XK(LK)*XK(LA)
00442  107*       GO TO 2017
00443  107*  2025 CALL D1D1WM(T(I),A(LA),XK(LK),G1)
00444  107*       J2 = J2+1
```

```
      LA = FLD(5,17,NSQ2(J2))                                     *NEW
      LK = FLD(22,14,NSQ2(J2))                                    *NEW
      G2 = XK(LK)*XK(LA)                                          **-2
      GO TO 2998
2030  TM = (T(I)+T(LTA))/2.0
2032  CALL PLYAWM(A(LA),TM,A(LA+1),XK(LK),G(LG))
      GO TO 2999
2035  TM = T(I)
      GO TO 2032
2040  CALL PLYAWM(A(LA),T(I),A(LA+1),XK(LK),G1)
2042  J2 = J2+1
      LA = FLD(5,17,NSQ2(J2))                                     *NEW
      LK = FLD(22,14,NSQ2(J2))                                    *NEW
      CALL PLYAWM(A(LA),T(LTA),A(LA+1),XK(LK),G2)                 **-2
      GO TO 2998
2045  G1 = XK(LK)*XK(LA)
      GO TO 2042
2050  CALL PLYAWM(A(LA),T(I),A(LA+1),XK(LK),G1)
      J2 = J2+1
      LA = FLD(5,17,NSQ2(J2))                                     *NEW
      LK = FLD(22,14,NSQ2(J2))                                    *NEW
      G2 = XK(LK)*XK(LA)                                          **-2
      GO TO 2998
2055  TM = (T(I)+T(LTA))/2.0
      CALL D2D1WM(TM,CON(14),A(LA),XK(LK),G(LG))
      GO TO 2999
2060  TM = T(LTA)
      GO TO 2007
2065  TM = T(LTA)
      GO TO 2032
2998  G(LG) = 1./(1./G1+1./G2)
      IF(FLD(3,1,NSQ1(J1)).EQ.1) G(LG) = G1*G2
2999  J2 = J2+1
3000  CONTINUE
      END
      GV = G(LG)
      T1 = X(IE1+I)+460.0
      T2 = X(IE1+LTA)+460.0
C     CHECK FOR RADIATION CONDUCTOR
      IF(FLD(3,1,NSQ1(J1)).EQ.0) GO TO 75
      G2 = G2+GV*(T1*T1+T2*T2)*(T1+T2)
      X(LE2) = X(LE2)+GV*(T2**4-T1**4)
      GO TO 80
   75 X(LE2) = X(LE2)+GV*(T2-T1)
      G2 = G2+GV
   80 IF(NSQ1(J1).GT.0) GO TO 70
C     OBTAIN THE MINIMUM STABILITY CRITERIA
      R1 = C(I)/G2
      IF(R1.GE.RC) GO TO 100
      RC = R1
      KON(35) = I
  100 CONTINUE
      CON(17) = RC*TSTEP
      IF(RC.LE.0.) GO TO 993
      BETAO = 2.0*CON(17)/CON(2)
      IF(BETAO.GT.1.0) BETAO = 1.0
      BETAN = 2.0-BETAO
C     CONVERT TEMPERATURES TO RANKINE
      DO 65 I = 1,NNT
      LE1 = IE1+I
```

CNVARB,CNVARB

```
00547          T(I) = T(I)+460.
00550       65 X(LE1) = X(LE1)+460.
00550    C     NOW RELAX THE NETWORK BY SUCCESSIVE POINT AND EXTRAPOLATION
00552          J1 = 0
00553      105 JJ = JJ+1
00554          DO 165 I = 1,NND
00557          IF(K1.GT.1) GO TO 110
00561          LE2 = IE2+I
00562          Q(I) = G(I)+BETAO*X(LE2)
00563      110 R1 = 0.0
00564          S = 0.0
00565          QSUM = 0.0
00566          GSUM = 0.0
00567      115 J1 = J1+1
00570          LG = FLD(5,16,NSQ1(J1))
00571          LTA = FLD(22,14,NSQ1(J1))
00572          GV = G(LG)
00572    C     CHECK FOR RADIATION CONDUCTOR
00573          IF(FLD(3,1,NSQ1(J1)).EQ.0) GO TO 120
00575          R1 = R1+GV
00576          QSUM = QSUM+GV*T(LTA)**4
00577          GO TO 125
00600      120 GSUM = GSUM+GV
00601          QSUM = QSUM+GV*T(LTA)
00602      125 IF(NSQ1(J1).GT.0) GO TO 115
00604          QSUM = Q(I)+BETAN*QSUM
00605          GSUM = C(I)+BETAN*GSUM
00605    C     DAMPEN RADIATION ON THIS NODE IF PRESENT
00606          IF(R1.LE.0.) GO TO 145
00610          R1 = R1*BETAN
00611          R2 = R1*T(I)**4
00612          T2 = (QSUM-R2)/GSUM
00613          R1 = R1*T2**4
00614          S = (R1+R2)/2.0
00614    C     OBTAIN THE NEW TEMPERATURE
00615      145 T2 = DN*((QSUM-S)/GSUM)+DD*T(I)
00615    C     OBTAIN THE CALCULATED TEMPERATURE DIFFERENCE
00616          T1 = ABS(T(I)-T2)
00616    C     STORE THE NEW AND OLD TEMPERATURES
00617          GO TO (160,155,150), JJ
00620      150 LE2 = IE2+I
00621          LE3 = IE3+I
00622          R1 = T2-T(I)
00623          X(LE2) = T(I)
00624          X(LE3) = R1/(R1-X(LE3))
00625          GO TO 160
00626      155 LE3 = IE3+I
00627          X(LE3) = T2-T(I)
00630      160 T(I) = T2
00631          IF(RLXD.GE.T1) GO TO 165
00633          RLXD = T1
00634          KK1 = I
00635      165 CONTINUE
00637          GO TO (180,180,170), JJ
00637    C     PERFORM LINEAR EXTRAPOLATION ON THE ERROR FUNCTION CURVE
00640      170 JJ = 0
00641          IF(KON(34).NE.0) GO TO 180
00643          DO 175 I = 1,NND
00646          LE3 = IE3+I
00646    C     SEE IF THE EXTRAPOLATION IS ALLOWABLE
```

```
00647  193*         IF(X(LE3).GE.0.) GO TO 175
00647  194*    C    LIMIT THE EXTRAPOLATION
00651  195*         IF(X(LE3).LT.-10.) X(LE3) = -10.
00653  196*         LE2 = IE2+I
00654  197*         T(I) = X(LE3)*X(LE2)+(1.0-X(LE3))*T(I)
00655  198*     175 CONTINUE
00657  199*     180 IF(NNA.LE.0) GO TO 220
00661  200*         JJ1 = J1
00662  201*         JJ2 = J2
00663  202*         DO 230 I = 1,NNT
00666  203*     230 T(I) = T(I)-460.
00670  204*         DO 215 I = NN,NNC
00673  205*         GSUM = 0.0
00674  206*         IF(K1.GT.1) GO TO 6000
00676  207*         INCLUDE VRO2,LIST
00677  207*         IF(FLD(4,1,NSQ1(JJ1+1)).EQ.0) GO TO 6000
00701  207*         NTYPE = FLD(0,5,NSQ2(JJ2))                                        *NEW
00702  207*         LA = FLD(5,17,NSQ2(JJ2))                                          *NEW
00703  207*         LK = FLD(22,14,NSQ2(JJ2))                                         *NEW
00704  207*         GO TO (5005,5010,5015,5020,5025,5030,5035,5040,5030), NTYFE       **-3
00705  207*    5005 Q(L) = XK(LK)+Q(L)
00706  207*         GO TO 5999
00707  207*    5010 Q1 = 0.0
00710  207*    5012 CALL D1D1WM(T(L),A(LA),XK(LK),Q2)
00711  207*         GO TO 5998
00712  207*    5015 Q1 = 0.0
00713  207*    5017 CALL D1D1WM(CON(14),A(LA),XK(LK),Q2)
00714  207*         GO TO 5998
00715  207*    5020 CALL D1D1WM(CON(14),A(LA),XK(LK),Q1)
00716  207*    5022 JJ2 = JJ2+1
00717  207*         LA = FLD(5,17,NSQ2(JJ2))                                          *NEW
00720  207*         LK = FLD(22,14,NSQ2(JJ2))                                         *NEW
00721  207*         GO TO 5017                                                        **-2
00722  207*    5025 Q1 = XK(LK)*XK(LA)
00723  207*         GO TO 5022
00724  207*    5030 CALL D1D1WM(CON(14),A(LA),XK(LK),Q1)
00725  207*         JJ2 = JJ2+1
00726  207*         LA = FLD(5,17,NSQ2(JJ2))                                          *NEW
00727  207*         LK = FLD(22,14,NSQ2(JJ2))                                         *NEW
00730  207*         Q2 = XK(LK)*XK(LA)                                                **-2
00731  207*         GO TO 5998
00732  207*    5035 CALL D1D1WM(CON(14),A(LA),XK(LK),Q1)
00733  207*    5037 JJ2 = JJ2+1
00734  207*         LA = FLD(5,17,NSQ2(JJ2))                                          *NEW
00735  207*         LK = FLD(22,14,NSQ2(JJ2))                                         *NEW
00736  207*         GO TO 5012                                                        **-2
00737  207*    5040 Q1 = XK(LK)*XK(LA)
00740  207*         GO TO 5037
00741  207*    5998 Q(L) = Q1+Q2+Q(L)
00742  207*    5999 JJ2 = JJ2+1
00743  207*    6000 CONTINUE
00744  207*         END
00745  208*         QSUM = Q(I)
00746  209*         L = I
00747  210*     185 JJ1 = JJ1+1
00750  211*         LG = FLD(5,16,NSQ1(JJ1))
00751  212*         LTA = FLD(22,14,NSQ1(JJ1))
00752  213*         IF(K1.GT.2) GO TO 4000
00754  214*         INCLUDE VRG2,LIST
00754  215*    C    CHECK FOR RADIATION CONDUCTOR
```

CNVARB,CNVARB

```
00755  215*        IF(FLD(2,1,NSQ1(JJ1)).EQ.0) GO TO 4000
00757  215*        NTYPE = FLD(0,5,NSQ2(JJ2))                                    *NEW
00760  215*        LA = FLD(5,17,NSQ2(JJ2))                                      *NEW
00761  215*        LK = FLD(22,14,NSQ2(JJ2))                                     *NEW
00762  215*        GOTO(3005,3010,3015,3020,3025,3030,3035,3040,3045,3050,3055,  **-3
00762  215*       $     3060,3065), NTYPE
00763  215*   3005 TM = (T(L)+T(LTA))/2.0
00764  215*   3007 CALL D1D1WM(TM,A(LA),XK(LK),G(LG))
00765  215*        GO TO 3999
00766  215*   3010 TM = T(L)
00767  215*        GO TO 3007
00770  215*   3015 CALL D1D1WM(T(L),A(LA),XK(LK),G1)
00771  215*   3017 JJ2 = JJ2+1
00772  215*        LA = FLD(5,17,NSQ2(JJ2))                                      *NEW
00773  215*        LK = FLD(22,14,NSQ2(JJ2))                                     *NEW
00774  215*        CALL D1D1WM(T(LTA),A(LA),XK(LK),G2)                           **-2
00775  215*        GO TO 3998
00776  215*   3020 G1 = XK(LK)*XK(LA)
00777  215*        GO TO 3017
01000  215*   3025 CALL D1D1WM(T(L),A(LA),XK(LK),G1)
01001  215*        JJ2 = JJ2+1
01002  215*        LA = FLD(5,17,NSQ2(JJ2))                                      *NEW
01003  215*        LK = FLD(22,14,NSQ2(JJ2))                                     *NEW
01004  215*        G2 = XK(LK)*XK(LA)                                            **-2
01005  215*        GO TO 3998
01006  215*   3030 TM = (T(L)+T(LTA))/2.0
01007  215*   3032 CALL PLYAWM(A(LA),TM,A(LA+1),XK(LK),G(LG))
01010  215*        GO TO 3999
01011  215*   3035 TM = T(L)
01012  215*        GO TO 3032
01013  215*   3040 CALL PLYAWM(A(LA),T(L),A(LA+1),XK(LK),G1)
01014  215*   3042 JJ2 = JJ2+1
01015  215*        LA = FLD(5,17,NSQ2(JJ2))                                      *NEW
01016  215*        LK = FLD(22,14,NSQ2(JJ2))                                     *NEW
01017  215*        CALL PLYAWM(A(LA),T(LTA),A(LA+1),XK(LK),G2)                   **-2
01020  215*        GO TO 3998
01021  215*   3045 G1 = XK(LK)*XK(LA)
01022  215*        GO TO 3042
01023  215*   3050 CALL PLYAWM(A(LA),T(L),A(LA+1),XK(LK),G1)
01024  215*        JJ2 = JJ2+1
01025  215*        LA = FLD(5,17,NSQ2(JJ2))                                      *NEW
01026  215*        LK = FLD(22,14,NSQ2(JJ2))                                     *NEW
01027  215*        G2 = XK(LK)*XK(LA)                                            **-2
01030  215*        GO TO 3998
01031  215*   3055 TM = (T(L)+T(LTA))/2.0
01032  215*        CALL D2D1WM(TM,CON(14),A(LA),XK(LK),G(LG))
01033  215*        GO TO 3999
01034  215*   3060 TM = T(LTA)
01035  215*        GO TO 3007
01036  215*   3065 TM = T(LTA)
01037  215*        GO TO 3032
01040  215*   3998 G(LG) = 1./(1./G1+1./G2)
01041  215*        IF(FLD(3,1,NSQ1(JJ1)).EQ.1) G(LG) = G1*G2
01043  215*   3999 JJ2 = JJ2+1
01044  215*   4000 CONTINUE
01045  215*        END
01046  216*        T1 = T(I)+460.0
01047  217*        T2 = T(LTA)+460.0
01050  218*        IF(FLD(3,1,NSQ1(JJ1)).EQ.0) GO TO 190
01052  219*        GV = G(LG)*(T1*T1+T2*T2)*(T1+T2)
```

```
01053              GO TO 195
01054          190 GV = G(LG)
01055          195 GSUM = GSUM+GV
01056              QSUM = QSUM+GV*T2
01056    C         CHECK FOR NEGATIVE CONDUCTOR
01057              IF(NSQ1(JJ1).GT.0) GO TO 185
01057    C         CALCULATE THE NEW TEMPERATURE
01061              T2 = AN*QSUM/GSUM+AA*T1
01062              T1 = ABS(T2-T1)
01063              T(I) = T2-460.0
01064              IF(RLXA.GE.T1) GO TO 215
01066              RLXA = T1
01067              KK2 = I
01070          215 CONTINUE
01072              DO 235 I = 1,NNT
01075          235 T(I) = T(I)+460.
01075    C         SEE IF THE ARITHMETIC RELAXATION CRITERIA WAS MET
01077              IF(RLXA.GT.CON(19)) GO TO 225
01077    C         SEE IF THE DIFFUSION RELAXATION CRITERIA WAS MET
01101          220 IF(RLXD.LE.CON(26)) GO TO 245
01103          225 IF(KON(7).EQ.0) GO TO 240
01105              CALL OUTCAL
01106          240 CONTINUE
01110              IF(KON(28).GE.65) CALL TOPLIN
01112              WRITE(6,882)
01114              KON(28) = KON(28)+2
01114    C         SEE IF THE TEMPERATURE CHANGES WERE TOO LARGE
01115          245 TCGD = 0.0
01116              TCGA = 0.0
01117              DO 250 I = 1,NND
01122              LE = IE1+I
01123              C(I) = C(I)*TSTEP
01124              T1 = ABS(T(I)-X(LE))
01125              IF(TCGD.GT.T1) GO TO 250
01127              TCGD = T1
01130              KON(36) = I
01131          250 CONTINUE
01133              IF(TCGD.LE.CON(6)) GO TO 265
01135              TSTEPN = 0.95*TSTEPN*CON(6)/TCGD
01136          255 DO 260 I = 1,NNT
01141              LE = IE1+I
01142          260 T(I) = X(LE)-460.
01144              GO TO 30
01145          265 IF(NNA.LE.0) GO TO 275
01147              DO 270 I = NN,NNC
01152              LE = IE1+I
01153              T1 = ABS(T(I)-X(LE))
01154              IF(TCGA.GT.T1) GO TO 270
01156              TCGA = T1
01157              KON(38) = I
01160          270 CONTINUE
01162              IF(TCGA.LE.CON(11)) GO TO 275
01164              TSTEPN = 0.95*TSTEPN*CON(11)/TCGA
01165              GO TO 255
01165    C          CONVERT TEMPERATURES BACK TO FARENHEIT
01166          275 DO 280 I = 1,NNT
01171          280 T(I) = T(I)-460.
01171    C         STORE THE TEMPERATURE AND RELAXATION CHANGES
01173              CON(15) = TCGD
01174              CON(16) = TCGA
```

```
                     CNVARB,CNVARB

01175  280*          CON(27) = RLXD
01176  281*          IF(RLXA.GT.RLXD) GO TO 285
01200  282*          KK2 = KK1
01201  283*          RLXA = RLXD
01202  284*      285 KON(37) = KK2
01203  285*          CON(30) = RLXA
01204  286*          CON(12) = 0.0
01205  287*          CALL VARBL2
01205  288*   C      CHECK THE BACKUP SWITCH
01206  289*          IF(KON(12).NE.0) GO TO 255
01206  290*   C      ADVANCE TIME
01210  291*          CON(13) = CON(1)
01211  292*          TSUM = TSUM+TSTEPN
01211  293*   C      CHECK FOR TIME TO PRINT
01212  294*          IF(TSUM.GE.CON(18)) GO TO 290
01212  295*   C      CHECK FOR PRINT EVERY ITERATION
01214  296*          IF(KON(7).NE.0) CALL OUTCAL
01216  297*          GO TO 10
01216  298*   C      TRY TO EVEN THE OUTPUT INTERVALS
01217  299*      290 TPRINT = TPRINT+TSUM
01220  300*          CALL OUTCAL
01220  301*   C      IS TIME GREATER THAN END COMPUTE TIME
01221  302*          IF(CON(1)*1.000001.LT.CON(3)) GO TO 5
01223  303*          NTH = IE1
01224  304*          NDIM = NLA
01225  305*          RETURN
01226  306*      990 WRITE(6,880)
01230  307*          GO TO 1000
01231  308*      991 WRITE(6,881)
01233  309*          GO TO 1000
01234  310*      993 WRITE(6,883)
01236  311*          GO TO 1000
01237  312*      994 WRITE(6,884) NDIM
01242  313*          GO TO 1000
01243  314*      995 WRITE(6,885)
01245  315*          GO TO 1000
01246  316*      996 WRITE(6,886)
01250  317*          GO TO 1000
01251  318*      997 WRITE(6,887)
01253  319*          GO TO 1000
01254  320*      998 WRITE(6,888)
01256  321*          GO TO 1000
01257  322*      999 WRITE(6,889)
01261  323*     1000 CALL OUTCAL
01262  324*          CALL EXIT
01263  325*      880 FORMAT(29H TRANSIENT TIME NOT SPECIFIED)
01264  326*      881 FORMAT(45H CNVARB REQUIRES LONG PSEUDO-COMPUTE SEQUENCE)
01265  327*      882 FORMAT(28H RELAXATION CRITERIA NOT MET)
01266  328*      883 FORMAT(24H CSGMIN ZERO OR NEGATIVE)
01267  329*      884 FORMAT(I8,20H LOCATIONS AVAILABLE)
01270  330*      885 FORMAT(10H NO DRLXCA)
01271  331*      886 FORMAT(10H NO DTIMEI)
01272  332*      887 FORMAT(10H NO ARLXCA)
01273  333*      888 FORMAT(19H NO OUTPUT INTERVAL)
01274  334*      889 FORMAT(9H NO NLOOP)
01275  335*          END

   END OF UNIVAC 1108 FORTRAN V COMPILATION.     0 *DIAGNOSTIC* MESSAGE(S)
 CNVARB          SYMBOLIC
 CNVARB  CODE    RELOCATABLE
```

C. COMPUTER LISTINGS OF SINDA STEADY STATE SOLUTION ROUTINES

```
QIW   FOR,*       CINDSS,CINDSS                                                       11 FEB 71
UNIVAC 1108 FORTRAN V ATHENA VERSION 131K-10D   CREATED ON 20 AUG 70
THIS COMPILATION WAS DONE ON 09 JUN 70 AT 14:00:06

   SUBROUTINE CINDSS     ENTRY POINT 003062

   STORAGE USED (BLOCK, NAME, LENGTH)

         0001    *CODE          003077
         0000    *CONST+TEMP    000106
         0002    *SIMPLE VAR    000045
         0004    *ARRAYS        000000
         0005    *BLANK         000000
         0006    TITLE    000001
         0007    TEMP     000001
         0010    CAP      000001
         0011    SOURCE   000001
         0012    COND     000001
         0013    PC1      000001
         0014    PC2      000001
         0015    KONST    000001
         0016    ARRAY    000001
         0017    FIXCON   000001
         0020    XSPACE   000003
         0021    DIMENS   000010

   EXTERNAL REFERENCES (BLOCK, NAME)

         0022    VARBL1
         0023    OUTCAL
         0024    D1D1WM
         0025    PLYAWM
         0026    D2D1WM
         0027    NONLIN
         0030    VARBL2
         0031    EXIT
         0032    NERR2$
         0033    NWDU$
         0034    NI02$
         0035    NER10$

   STORAGE ASSIGNMENT FOR VARIABLES (BLOCK, TYPE, RELATIVE LOCATION, NAME)

    0016 R 000000 A        0010 R 000000 C        0017 R 000000 CON      0002 R 000015 DD       0002 R 000014 DN
    0012 R 000000 G        0002 R 000040 GSUM     0002 R 000033 GV       0002 R 000027 G1       0002 R 000030 G2
    0006 R 000000 H        0002 I 000007 I        0002 I 000003 IE       0002 I 000036 JJ1      0002 I 000037 JJ2
    0002 I 000010 J1       0002 I 000011 J2       0015 I 000000 K        0017 I 000000 KON      0002 I 000006 K1
    0002 I 000041 L        0002 I 000020 LA       0002 I 000005 LAX      0002 I 000016 LE       0002 I 000034 LE1
    0002 I 000024 LG       0002 I 000021 LK       0021 I 000006 LS01     0021 I 000007 LS02     0002 I 000025 LTA
    0021 I 000005 NAT      0021 I 000004 NCT      0020 I 000000 NDIM     0021 I 000003 NGT      0002 I 000004 NLA
    0002 I 000001 NN       0021 I 000001 NNA      0002 I 000002 NNC      0021 I 000000 NND      0021 I 000002 NNT
    0013 I 000000 NSQ1     0014 I 000000 NSQ2     0020 I 000001 NTH      0002 I 000017 NTYPE    0020   000002 NX
    0002 I 000035 N1       0002 I 000042 N2       0002 R 000000 PASS     0011 R 000000 Q        0002 R 000044 QIN
    0002 R 000043 QOUT     0002 R 000022 Q1       0002 R 000023 Q2       0002 R 000013 RLXA     0002 R 000012 RLXD
    0007 R 000000 T        0002 R 000026 TM       0002 R 000031 T1       0002 R 000032 T2       0020 R 000002 X
```

```
CINDSS,CINDSS

0015 R 000000 XK       0001  000104 10L      0001  003036 1030L    0001  002537 110L     0001  002543 115L
0001   002605 125L     0001  000125 15L      0001  000140 155G     0001  002620 155L     0001  000145 161G
0001   002634 165L     0001  002712 170L     0001  002721 175L     0001  000256 1998L    0001  000261 1999L
0001   000264 2000L    0001  000624 2005L    0001  000631 2007L    0001  000651 2010L    0001  000654 2015L
0001   000672 2017L    0001  000726 2020L    0001  000734 2025L    0001  000203 203G     0001  000774 2030L
0001   001001 2032L    0001  001024 2035L    0001  001027 2040L    0001  001050 2042L    0001  001107 2045L
0001   001115 2050L    0001  001160 2055L    0001  001206 2060L    0001  001212 2065L    0001  000215 211G
0001   000161 25L      0001  001216 2998L    0001  001244 2999L    0001  001247 3000L    0001  001777 3005L
0001   002005 3007L    0001  002025 3010L    0001  002031 3015L    0001  002050 3017L    0001  002104 3020L
0001   002112 3025L    0001  002153 3030L    0001  002161 3032L    0001  002204 3035L    0001  002210 3040L
0001   002232 3042L    0001  002271 3045L    0001  002277 3050L    0001  002343 3055L    0001  002372 3060L
0001   002376 3065L    0001  000535 35L      0001  002402 3998L    0001  002430 3999L    0001  001300 40L
0001   002433 4000L    0001  000327 4005L    0001  000334 4010L    0001  000335 4012L    0001  000354 4015L
0001   000355 4017L    0001  000372 4020L    0001  000406 4022L    0001  000423 4025L    0001  000431 4030L
0001   000467 4035L    0001  000503 4037L    0001  000520 4040L    0001  001431 434G     0001  001303 45L
0001   000526 4998L    0001  000532 4999L    0001  001351 50L      0001  000535 5000L    0001  001502 5005L
0001   001510 5010L    0001  001511 5012L    0001  001531 5015L    0001  001532 5017L    0001  001547 5020L
0001   001563 5022L    0001  001600 5025L    0001  001606 5030L    0001  001644 5035L    0001  001660 5037L
0001   001675 5040L    0001  001703 5998L    0001  001710 5999L    0001  001713 6000L    0001  002631 657G
0001   001406 70L      0001  001416 75L      0001  001713 80L      0001  002470 85L      0000  000000 882F
0000   000001 883F     0000  000007 884F     0000  000020 885F     0000  000030 886F     0000  000035 887F
0000   000040 888F     0000  000043 889F     0001  002473 90L      0001  003000 994L     0001  003006 996L
0001   003015 997L     0001  003023 998L     0001  003031 999L
```

```
00101    1*          SUBROUTINE CINDSS
00101    2*    C     STEADY STATE EXECUTION SUBROUTINE FOR SINDA      FORTRAN V
00101    3*    C     THE SHORT PSEUDO-COMPUTE SEQUENCE IS REQUIRED
00101    4*    C     DIFFUSION NODES RECEIVE A BLOCK ITERATION
00101    5*    C     ARITHMETIC NODES RECEIVE A SUCESSIVE POINT ITERATION
00101    6*    C     OVER-RELAXATION IS ALLOWED, THE DAMPING FACTORS ARE ADDRESSABLE
00103    7*          INCLUDE COMM,LIST
00104    7*          COMMON /TITLE/H(1) /TEMP/T(1) /CAP/C(1) /SOURCE/Q(1) /COND/G(1)
00105    7*          COMMON /PC1/NSQ1(1) /PC2/NSQ2(1) /KONST/K(1) /ARRAY/A(1)
00106    7*          COMMON /FIXCON/KON(1) /XSPACE/NDIM,NTH,X(1)
00107    7*          COMMON /DIMENS/ NND,NNA,NNT,NGT,NCT,NAT,LSQ1,LSQ2
00110    7*          DIMENSION CON(1),XK(1),NX(1)
00111    7*          EQUIVALENCE (KON(1),CON(1)),(K(1),XK(1)),(X(1),NX(1))
00112    7*          END
00113    8*          INCLUDE DEFF,LIST
00113    8*    C************* CONTROL CONSTANT DEFINITIONS AND NAMES  ****************
00113    8*    C     CONTROL CONSTANT 1 CONTAINS THE NEW PROBLEM TIME            (TIMEN)
00113    8*    C     CONTROL CONSTANT 2 CONTAINS THE TIME STEP USED             (DTIMEU)
00113    8*    C     CONTROL CONSTANT 3 CONTAINS THE PROBLEM STOP TIME          (TIMEND)
00113    8*    C     CONTROL CONSTANT 4 CONTAINS THE TIME STEP FACTOR,EXPLICIT (CSGFAC)
00113    8*    C     CC5 IS THE INPUT NUMBER OF ITERATION DO LOOPS, INTEGER      (NLOOP)
00113    8*    C     CC6 CONTAINS THE DIFFUSION TEMPERATURE CHANGE ALLOWED     (DTMPCA)
00113    8*    C     CC7 CONTAINS THE OUTPUT EACH ITERATION SWITCH             (OPEITR)
00113    8*    C     CC8 CONTAINS THE MAXIMUM ALLOWED TIME STEP                (DTIMEH)
00113    8*    C     CC9 CONTAINS THE NEW ARITHMETIC TEMP. DAMPING FACTOR       (DAMPA)
00113    8*    C     CC10 CONTAINS THE NEW DIFFUSION TEMP. DAMPING FACTOR       (DAMPD)
00113    8*    C     CC11 CONTAINS THE MAXIMUM ALLOWED ARITHMETIC TEMP. CHANGE (ATMPCA)
00113    8*    C     CC12 CONTAINS THE BACKUP SWITCH CHECKED AFTER VARIABLES   (BACKUP)
00113    8*    C     CC13 CONTAINS THE PRESENT TIME OR PROBLEM START TIME       (TIMEO)
00113    8*    C     CC14 CONTAINS THE MEAN TIME BETWEEN AN ITERATION           (TIMEM)
00113    8*    C     CC15 CONTAINS THE DIFFUSION TEMPERATURE CHANGE CALCULATED (DTMPCC)
00113    8*    C     CC16 CONTAINS ARITHMETIC TEMPERATURE CHANGE CALCULATED    (ATMPCC)
00113    8*    C     CONTROL CONSTANT 17 IS RESERVED FOR THE C/SG MINIMUM      (CSGMIN)
```

```
C     CONTROL CONSTANT 18 CONTAINS THE OUTPUT INTERVAL             (OUTPUT)
C     CC19 CONTAINS THE ARITHMETIC RELAXATION CRITERIA ALLOWED     (ARLXCA)
C     CC20 CONTAINS THE NUMBER OF RELAXATION LOOPS USED,INTEGER    (LOOPCT)
C     CC21 CONTAINS THE MINIMUM ALLOWED TIME STEP                  (DTIMEL)
C     CC22 IS FOR THE INPUT TIME STEP IMPLICIT                     (DTIMEI)
C     CC23 CONTAINS THE C/SG MAXIMUM                               (CSGMAX)
C     CC24 CONTAINS THE C/SG RANGE ALLOWED                         (CSGRAL)
C     CC25 CONTAINS THE C/SG RANGE CALCULATED                      (CSGRCL)
C     CC26 CONTAINS THE DIFFUSION RELAXATION CRITERIA ALLOWED      (DRLXCA)
C     CC27 CONTAINS THE DIFFUSION RELAXATION CHANGE CALCULATED     (DRLXCC)
C     CC28 CONTAINS THE LINE COUNTER, INTEGER                      (LINECT)
C     CC29 CONTAINS THE PAGE COUNTER, INTEGER                      (PAGECT)
C     CC30 CONTAINS ARITHMETIC RELAXATION CHANGE CALCULATED        (ARLXCC)
C     CC31 IS INDICATOR, 0=THERMAL SPCS,1=THERMAL LPCS,2=GENERAL  (LSPCS)
C     CC32 CONTAINS THE ENERGY BALANCE OF THE SYSTEM, IN - OUT    (ENGBAL)
C     CC33 CONTAINS THE DESIRED ENERGY BALANCE, USER INPUT         (BALENG)
C     CC34 CONTAINS THE NOCOPY SWITCH FOR MATRIX USERS             (NOCOPY)
C     CC35 CONTAINS RELATIVE NODE NUMBER OF CSGMIN
C     CC36 CONTAINS RELATIVE NODE NUMBER OF DTMPCC
C     CC37 CONTAINS RELATIVE NODE NUMBER OF ARLXCC
C     CC38 CONTAINS RELATIVE NODE NUMBER OF ATMPCC
C     CC39-40-41-42-43 CONTAIN DUMMY INTEGER CONSTANTS   (I-J-K-L-MTEST)
C     CC44-45-46-47-48 CONTAIN DUMMY FLOATING CONSTANTS  (R-S-T-U-VTEST)
C     CC49 IS THE QUASI-LINEARIZATION INTERVAL FOR CINDSM         (LAXFAC)
C     CC50 IS NOT USED AT PRESENT
      END
      IF(KON(5).LE.0) GO TO 999
      IF(CON(9).LE.0.) CON(9) = 1.0
      IF(CON(10).LE.0.) CON(10) = 1.0
      IF(NNA.GT.0.AND.CON(19).LE.0.) GO TO 998
      IF(NND.GT.0.AND.CON(26).LE.0.) GO TO 997
      IF(KON(31).NE.0) GO TO 994
      PASS = -1.0
      NN = NND+1
      NNC=NNA+NND
      IE=NTH
      NLA = NDIM
      NTH=NTH+NND
      NDIM=NDIM-NND
      IF(NDIM.LT.0) GO TO 996
      CON(1) = CON(13)
      CON(2) = 0.0
      CON(14) = CON(13)
      GO TO 15
   10 CON(1) = CON(13)+CON(18)
      IF(CON(1)-CON(3).GT.0.) CON(1) = CON(3)
      CON(14) = (CON(1)+CON(13))/2.0
      CON(2) = CON(1)-CON(13)
C     COMPUTE STEADY STATE TEMPERATURES
   15 LAX = KON(5)
      DO 120 K1 = 1,LAX
      KON(20) = K1
C     ZERO OUT ALL SOURCE LOCATIONS
      DO 20 I = 1,NNC
   20 Q(I)=0.
      CALL VARBL1
      IF(PASS.GE.0.)GO TO 25
      CALL OUTCAL
      PASS = 1.0
   25 J1 = 0
```

CINDSS,CINDSS

```
      J2 = 1
      RLXD = 0.0
      RLXA = 0.0
      IF(NND.LE.0) GO TO 75
      DN = CON(10)
      DD = 1.0-DN
C     ZERO OUT EXTRA LOCATIONS
      DO 30 I = 1,NND
      LE=IE+I
   30 X(LE)=0.0
C     DO A BLOCK ITERATION ON THE DIFFUSION NODES
      DO 70 I = 1,NND
      LE=IE+I
      INCLUDE DUMC,LIST
      IF(FLD(1,1,NSQ1(J1+1)).EQ.0) GO TO 2000
      NTYPE = FLD(0,5,NSQ2(J2))                                                    *NEW
      GO TO (1999,1998,1998,1998,1999,1998,1998,1998,1999), NTYPE                  **-1
 1998 J2 = J2+1
 1999 J2 = J2+1
 2000 CONTINUE
      END
      INCLUDE VARQ,LIST
      IF(FLD(4,1,NSQ1(J1+1)).EQ.0) GO TO 5000
      NTYPE = FLD(0,5,NSQ2(J2))                                                    *NEW
      LA = FLD(5,17,NSQ2(J2))                                                      *NEW
      LK = FLD(22,14,NSQ2(J2))                                                     *NEW
      GO TO (4005,4010,4015,4020,4025,4030,4035,4040,4030), NTYPE                  **-3
 4005 Q(I) = XK(LK)+Q(I)
      GO TO 4999
 4010 Q1 = 0.0
 4012 CALL D1D1WM(T(I),A(LA),XK(LK),Q2)
      GO TO 4998
 4015 Q1 = 0.0
 4017 CALL D1D1WM(CON(14),A(LA),XK(LK),Q2)
      GO TO 4998
 4020 CALL D1D1WM(CON(14),A(LA),XK(LK),Q1)
 4022 J2 = J2+1
      LA = FLD(5,17,NSQ2(J2))                                                      *NEW
      LK = FLD(22,14,NSQ2(J2))                                                     *NEW
      GO TO 4017                                                                   **-2
 4025 Q1 = XK(LK)*XK(LA)
      GO TO 4022
 4030 CALL D1D1WM(CON(14),A(LA),XK(LK),Q1)
      J2 = J2+1
      LA = FLD(5,17,NSQ2(J2))                                                      *NEW
      LK = FLD(22,14,NSQ2(J2))                                                     *NEW
      Q2 = XK(LK)*XK(LA)                                                           **-2
      GO TO 4998
 4035 CALL D1D1WM(CON(14),A(LA),XK(LK),Q1)
 4037 J2 = J2+1
      LA = FLD(5,17,NSQ2(J2))                                                      *NEW
      LK = FLD(22,14,NSQ2(J2))                                                     *NEW
      GO TO 4012                                                                   **-2
 4040 Q1 = XK(LK)*XK(LA)
      GO TO 4037
 4998 Q(I) = Q1+Q2+Q(I)
 4999 J2 = J2+1
 5000 CONTINUE
      END
   35 J1 = J1+1
```

CINDSS.CINDSS

```
      LG = FLD(5,16,NSQ1(J1))
      IF(LG.EQ.0) GO TO 50
      LTA = FLD(22,14,NSQ1(J1))
      INCLUDE VARG.LIST
C     CHECK FOR RADIATION CONDUCTOR
      IF(FLD(2,1,NSQ1(J1)).EQ.0) GO TO 3000
      NTYPE = FLD(0,5,NSQ2(J2))                                          *NEW
      LA = FLD(5,17,NSQ2(J2))                                            *NEW
      LK = FLD(22,14,NSQ2(J2))                                           *NEW
      GOTO(2005,2010,2015,2020,2025,2030,2035,2040,2045,2050,2055,       **-3
     $     2060,2065), NTYPE
 2005 TM = (T(I)+T(LTA))/2.0
 2007 CALL D1D1WM(TM,A(LA),XK(LK),G(LG))
      GO TO 2999
 2010 TM = T(I)
      GO TO 2007
 2015 CALL D1D1WM(T(I),A(LA),XK(LK),G1)
 2017 J2 = J2+1
      LA = FLD(5,17,NSQ2(J2))                                            *NEW
      LK = FLD(22,14,NSQ2(J2))                                           *NEW
      CALL D1D1WM(T(LTA),A(LA),XK(LK),G2)                                **-2
      GO TO 2998
 2020 G1 = XK(LK)*XK(LA)
      GO TO 2017
 2025 CALL D1D1WM(T(I),A(LA),XK(LK),G1)
      J2 = J2+1
      LA = FLD(5,17,NSQ2(J2))                                            *NEW
      LK = FLD(22,14,NSQ2(J2))                                           *NEW
      G2 = XK(LK)*XK(LA)                                                 **-2
      GO TO 2998
 2030 TM = (T(I)+T(LTA))/2.0
 2032 CALL PLYAWM(A(LA),TM,A(LA+1),XK(LK),G(LG))
      GO TO 2999
 2035 TM = T(I)
      GO TO 2032
 2040 CALL PLYAWM(A(LA),T(I),A(LA+1),XK(LK),G1)
 2042 J2 = J2+1
      LA = FLD(5,17,NSQ2(J2))                                            *NEW
      LK = FLD(22,14,NSQ2(J2))                                           *NEW
      CALL PLYAWM(A(LA),T(LTA),A(LA+1),XK(LK),G2)                        **-2
      GO TO 2998
 2045 G1 = XK(LK)*XK(LA)
      GO TO 2042
 2050 CALL PLYAWM(A(LA),T(I),A(LA+1),XK(LK),G1)
      J2 = J2+1
      LA = FLD(5,17,NSQ2(J2))                                            *NEW
      LK = FLD(22,14,NSQ2(J2))                                           *NEW
      G2 = XK(LK)*XK(LA)                                                 **-2
      GO TO 2998
 2055 TM = (T(I)+T(LTA))/2.0
      CALL D2D1WM(TM,CON(14),A(LA),XK(LK),G(LG))
      GO TO 2999
 2060 TM = T(LTA)
      GO TO 2007
 2065 TM = T(LTA)
      GO TO 2032
 2998 G(LG) = 1./(1./G1+1./G2)
      IF(FLD(3,1,NSQ1(J1)).EQ.1) G(LG) = G1*G2
 2999 J2 = J2+1
 3000 CONTINUE
```

```
00372   63*           END
00373   64*           IF(FLD(3,1,NSQ1(J1)).EQ.0) GO TO 40
00375   65*           T1 = T(I)+460.0
00376   66*           T2 = T(LTA)+460.0
00377   67*           GV = G(LG)*(T1*T1+T2*T2)*(T1+T2)
00400   68*           GO TO 45
00401   69*        40 GV = G(LG)
00402   70*        45 X(LE) = X(LE)+GV
00403   71*           Q(I) = Q(I)+GV*T(LTA)
00403   72*     C     CHECK FOR ADJOINING DIFFUSION NODE, WATCH FOR ONE WAY CONDUCTOR
00404   73*           IF(LTA.GT.NND.OR.FLD(21,1,NSQ1(J1)).EQ.1) GO TO 50
00406   74*           LE1 = IE+LTA
00407   75*           X(LE1) = X(LE1)+GV
00410   76*           Q(LTA) = Q(LTA)+GV*T(I)
00410   77*     C     CHECK FOR LAST CONDUCTOR TO THIS NODE
00411   78*        50 IF(NSQ1(J1).GT.0) GO TO 35
00413   79*           T2 = DD*T(I)+DN*Q(I)/X(LE)
00413   80*     C     OBTAIN THE CALCULATED TEMPERATURE DIFFERENCE
00414   81*           T1=ABS(T(I)-T2)
00414   82*     C     STORE THE NEW TEMPERATURE
00415   83*           T(I)=T2
00415   84*     C     SAVE THE MAXIMUM DIFFUSION RELAXATION  CHANGE
00416   85*           IF(RLXD.GT.T1) GO TO 70
00420   86*           RLXD = T1
00421   87*           N1 = I
00422   88*        70 CONTINUE
00424   89*           CON(27) = RLXD
00425   90*           IF(NNA.LE.0) GO TO 115
00427   91*        75 DN = CON(9)
00430   92*           DD = 1.0-DN
00431   93*           JJ1 = J1
00432   94*           JJ2 = J2
00432   95*     C     DO A SUCCESSIVE POINT ITERATION ON THE ARITHMETIC NODES
00433   96*           DO 110 I = NN,NNC
00436   97*           GSUM = 0.0
00437   98*           L = I
00440   99*           INCLUDE VRQ2,LIST
00441   99*           IF(FLD(4,1,NSQ1(JJ1+1)).EQ.0) GO TO 6000
00443   99*           NTYPE = FLD(0,5,NSQ2(JJ2))                                        *NEW
00444   99*           LA = FLD(5,17,NSQ2(JJ2))                                          *NEW
00445   99*           LK = FLD(22,14,NSQ2(JJ2))                                         *NEW
00446   99*           GO TO (5005,5010,5015,5020,5025,5030,5035,5040,5030), NTYPE       **-3
00447   99*      5005 Q(L) = XK(LK)+Q(L)
00450   99*           GO TO 5999
00451   99*      5010 Q1 = 0.0
00452   99*      5012 CALL D1D1WM(T(L),A(LA),XK(LK),Q2)
00453   99*           GO TO 5998
00454   99*      5015 Q1 = 0.0
00455   99*      5017 CALL D1D1WM(CON(14),A(LA),XK(LK),Q2)
00456   99*           GO TO 5998
00457   99*      5020 CALL D1D1WM(CON(14),A(LA),XK(LK),Q1)
00460   99*      5022 JJ2 = JJ2+1
00461   99*           LA = FLD(5,17,NSQ2(JJ2))                                          *NEW
00462   99*           LK = FLD(22,14,NSQ2(JJ2))                                         *NEW
00463   99*           GO TO 5017                                                        **-2
00464   99*      5025 Q1 = XK(LK)*XK(LA)
00465   99*           GO TO 5022
00466   99*      5030 CALL D1D1WM(CON(14),A(LA),XK(LK),Q1)
00467   99*           JJ2 = JJ2+1
00470   99*           LA = FLD(5,17,NSQ2(JJ2))                                          *NEW
```

```
 99*            LK = FLD(22,14,NSQ2(JJ2))                                        *NEW
 99*            Q2 = XK(LK)*XK(LA)                                               **-2
 99*            GO TO 5998
 99*       5035 CALL D1D1WM(CON(14),A(LA),XK(LK),Q1)
 99*       5037 JJ2 = JJ2+1
 99*            LA = FLD(5,17,NSQ2(JJ2))                                         *NEW
 99*            LK = FLD(22,14,NSQ2(JJ2))                                        *NEW
 99*            GO TO 5012                                                       **-2
 99*       5040 Q1 = XK(LK)*XK(LA)
 99*            GO TO 5037
 99*       5998 Q(L) = Q1+Q2+Q(L)
 99*       5999 JJ2 = JJ2+1
 99*       6000 CONTINUE
 99*            END
100*         80 JJ1 = JJ1+1
101*            LG = FLD(5,16,NSQ1(JJ1))
102*            LTA = FLD(22,14,NSQ1(JJ1))
103*            INCLUDE VRG2,LIST
104*       C    CHECK FOR RADIATION CONDUCTOR
104*            IF(FLD(2,1,NSQ1(JJ1)).EQ.0) GO TO 4000
104*            NTYPE = FLD(0,5,NSQ2(JJ2))                                       *NEW
104*            LA = FLD(5,17,NSQ2(JJ2))                                         *NEW
104*            LK = FLD(22,14,NSQ2(JJ2))                                        *NEW
104*            GOTO(3005,3010,3015,3020,3025,3030,3035,3040,3045,3050,3055,     **-3
104*         $       3060,3065), NTYPE
104*       3005 TM = (T(L)+T(LTA))/2.0
104*       3007 CALL D1D1WM(TM,A(LA),XK(LK),G(LG))
104*            GO TO 3999
104*       3010 TM = T(L)
104*            GO TO 3007
104*       3015 CALL D1D1WM(T(L),A(LA),XK(LK),G1)
104*       3017 JJ2 = JJ2+1
104*            LA = FLD(5,17,NSQ2(JJ2))                                         *NEW
104*            LK = FLD(22,14,NSQ2(JJ2))                                        *NEW
104*            CALL D1D1WM(T(LTA),A(LA),XK(LK),G2)                              **-2
104*            GO TO 3998
104*       3020 G1 = XK(LK)*XK(LA)
104*            GO TO 3017
104*       3025 CALL D1D1WM(T(L),A(LA),XK(LK),G1)
104*            JJ2 = JJ2+1
104*            LA = FLD(5,17,NSQ2(JJ2))                                         *NEW
104*            LK = FLD(22,14,NSQ2(JJ2))                                        *NEW
104*            G2 = XK(LK)*XK(LA)                                               **-2
104*            GO TO 3998
104*       3030 TM = (T(L)+T(LTA))/2.0
104*       3032 CALL PLYAWM(A(LA),TM,A(LA+1),XK(LK),G(LG))
104*            GO TO 3999
104*       3035 TM = T(L)
104*            GO TO 3032
104*       3040 CALL PLYAWM(A(LA),T(L),A(LA+1),XK(LK),G1)
104*       3042 JJ2 = JJ2+1
104*            LA = FLD(5,17,NSQ2(JJ2))                                         *NEW
104*            LK = FLD(22,14,NSQ2(JJ2))                                        *NEW
104*            CALL PLYAWM(A(LA),T(LTA),A(LA+1),XK(LK),G2)                      **-2
104*            GO TO 3998
104*       3045 G1 = XK(LK)*XK(LA)
104*            GO TO 3042
104*       3050 CALL PLYAWM(A(LA),T(L),A(LA+1),XK(LK),G1)
104*            JJ2 = JJ2+1
104*            LA = FLD(5,17,NSQ2(JJ2))                                         *NEW
```

CINDSS,CINDSS

```
      LK = FLD(22,14,NSQ2(JJ2))                                     *NEW
      G2 = XK(LK)*XK(LA)                                            **-2
      GO TO 3998
 3055 TM = (T(L)+T(LTA))/2.0
      CALL D2D1WM(TM,CON(14),A(LA),XK(LK),G(LG))
      GO TO 3999
 3060 TM = T(LTA)
      GO TO 3007
 3065 TM = T(LTA)
      GO TO 3032
 3998 G(LG) = 1./(1./G1+1./G2)
      IF(FLD(3,1,NSQ1(JJ1)).EQ.1) G(LG) = G1*G2
 3999 JJ2 = JJ2+1
 4000 CONTINUE
      END
      IF(FLD(3,1,NSQ1(JJ1)).EQ.0) GO TO 85
      T1 = T(I)+460.0
      T2 = T(LTA)+460.0
      GV = G(LG)*(T1*T1+T2*T2)*(T1+T2)
      GO TO 90
   85 GV = G(LG)
   90 Q(I) = Q(I)+GV*T(LTA)
      GSUM = GSUM+GV
C     CHECK FOR LAST CONDUCTOR TO THIS NODE
      IF(NSQ1(JJ1).GT.0) GO TO 80
      T2 = DD*T(I)+DN*Q(I)/GSUM
      T1 = ABS(T(I)-T2)
C     STORE THE NEW TEMPERATURES
      T(I) = T2
      IF(RLXA.GT.T1) GO TO 110
      RLXA = T1
      N2 = I
  110 CONTINUE
      CON(30) = RLXA
C     SEE IF THE RELAXATION CRITERIA ARE MET
  115 IF(RLXA.LE.CON(19).AND.RLXD.LE.CON(26)) GO TO 125
      IF(KON(7).NE.0) CALL OUTCAL
  120 CONTINUE
      WRITE(6,882)
      WRITE(6,885) LAX
      KON(28) = KON(28)+2
  125 KON(37) = N2
      IF(RLXA.GT.RLXD) GO TO 155
      CON(30) = RLXD
      KON(37) = N1
C     CHECK THE ENERGY BALANCE OF THE SYSTEM
  155 CALL NONLIN
      QOUT = 0.0
      QIN = 0.0
      J1 = 0
      DO 195 I = 1,NNC
      QIN = QIN+Q(I)
  165 J1 = J1+1
      LTA = FLD(22,14,NSQ1(J1))
      IF(LTA.LE.NNC) GO TO 175
      LG = FLD(5,16,NSQ1(J1))
      IF(FLD(3,1,NSQ1(J1)).EQ.0) GO TO 170
      T1 = T(I)+460.0
      T2 = T(LTA)+460.0
      QOUT = QOUT+G(LG)*(T1**4-T2**4)
```

```
00674              GO TO 175
00675          170 QOUT = QOUT+G(LG)*(T(I)-T(LTA))
00675        C     CHECK FOR LAST CONDUCTOR TO THIS NODE
00676          175 IF(NSQ1(J1).GT.0) GO TO 165
00700          195 CONTINUE
00702              CON(32) = ABS(QIN-QOUT)
00703              CALL VARBL2
00704              CON(13) = CON(1)
00705              CALL OUTCAL
00706              WRITE(6,882)
00710              WRITE(6,883) KON(20),CON(32)
00714              KON(28) = KON(28)+2
00715              IF(CON(3).GT.CON(1)*1.000001) GO TO 10
00717              NTH=IE
00720              NDIM = NLA
00721              RETURN
00722          994 WRITE(6,884)
00724              GO TO 1000
00725          996 WRITE(6,886) NDIM
00730              GO TO 1000
00731          997 WRITE(6,887)
00733              GO TO 1000
00734          998 WRITE(6,888)
00736              GO TO 1000
00737          999 WRITE(6,889)
00741         1000 CALL OUTCAL
00742              CALL EXIT
00743          882 FORMAT(1H )
00744          883 FORMAT(10H LOOPCT = I6,10H ENGBAL = E12.5)
00745          884 FORMAT(46H CINDSS REQUIRES SHORT PSEUDO-COMPUTE SEGUENCE)
00746          885 FORMAT(35H ITERATION COUNT EXCEEDED, NLOOP = ,  I10)
00747          886 FORMAT(I8,20H LOCATIONS AVAILABLE) .
00750          887 FORMAT(10H NO DRLXCA)
00751          888 FORMAT(10H NO ARLXCA)
00752          889 FORMAT(14H NO LOOP COUNT)
00753              END

    END OF UNIVAC 1108 FORTRAN V COMPILATION.     0 *DIAGNOSTIC* MESSAGE(S)
   CINDSS          SYMBOLIC
   CINDSS  CODE    RELOCATABLE
```

```
QIW   FOR,*        CINDSL,CINDSL                                                        21 FEB 71        13:59:58, 71
UNIVAC 1108 FORTRAN V ATHENA VERSION 131K-10D   CREATED ON 20 AUG 70
THIS COMPILATION WAS DONE ON 09 JUN 70 AT 13:59:58

   SUBROUTINE CINDSL    ENTRY POINT 003163

   STORAGE USED (BLOCK, NAME, LENGTH)

         0001    *CODE          003200
         0000    *CONST+TEMP    000110
         0002    *SIMPLE VAR    000050
         0004    *ARRAYS        000000
         0005    *BLANK         000000
         0006    TITLE    000001
         0007    TEMP     000001
         0010    CAP      000001
         0011    SOURCE   000001
         0012    COND     000001
         0013    PC1      000001
         0014    PC2      000001
         0015    KONST    000001
         0016    ARRAY    000001
         0017    FIXCON   000001
         0020    XSPACE   000003
         0021    DIMENS   000010

   EXTERNAL REFERENCES (BLOCK, NAME)

         0022    VARBL1
         0023    OUTCAL
         0024    D1D1WM
         0025    PLYAWM
         0026    D2D1WM
         0027    NONLIN
         0030    VARBL2
         0031    EXIT
         0032    NERR2$
         0033    NWDU$
         0034    NI02$
         0035    NER10$

   STORAGE ASSIGNMENT FOR VARIABLES (BLOCK, TYPE, RELATIVE LOCATION, NAME)

    0016 R 000000 A         0010 R 000000 C         0017 R 000000 CON       0002 R 000017 DD        0002 R 000016 DN
    0012 R 000000 G         0002 R 000020 GSUM      0002 R 000035 GV        0002 R 000031 G1        0002 R 000032 G2
    0006 R 000000 H         0002 I 000011 I         0002 I 000003 IE1       0002 I 000004 IE2       0002 I 000006 JJ
    0002 I 000042 JJ1       0002 I 000043 JJ2       0002 I 000012 J1        0002 I 000013 J2        0015 I 000000 K
    0017 I 000000 KON       0002 I 000010 K1        0002 I 000044 L         0002 I 000022 LA        0002 I 000007 LAX
    0002 I 000036 LE1       0002 I 000037 LE2       0002 I 000026 LG        0002 I 000023 LK        0021 I 000006 LSQ1
    0021 I 000007 LSQ2      0002 I 000027 LTA       0021 I 000005 NAT       0021 I 000004 NCT       0020 I 000000 NDIM
    0021 I 000003 NGT       0002 I 000005 NLA       0002 I 000001 NN        0021 I 000001 NNA       0002 I 000002 NNC
    0021 I 000000 NND       0021 I 000002 NNT       0013 I 000000 NSQ1      0014 I 000000 NSQ2      0020 I 000001 NTH
    0002 I 000021 NTYPE     0020   000002 NX        0002 I 000041 N1        0002 I 000045 N2        0002 R 000000 PASS
    0011 R 000000 Q         0002 R 000047 QIN       0002 R 000046 QOUT      0002 R 000024 Q1        0002 R 000025 Q2
    0002 R 000015 RLXA      0002 R 000014 RLXD      0002 R 000040 R1        0007 R 000000 T         0002 R 000030 TM
```

CINDSL,CIN SL

```
0002 R 000033 T1       0002 R 000034 T2       0020 R 000002 X        0015 R 000000 XK       0001   000133 10L
0001   003137 1000L    0001   002522 110L     0001   002545 115L     0001   002554 120L     0001   002567 125L
0001   002573 130L     0001   002660 135L     0001   002662 140L     0001   002706 150L     0001   002721 155L
0001   000147 160G     0001   000157 165G     0001   002735 165L     0001   003013 170L     0001   003022 175L
0001   000253 1998L    0001   000256 1999L    0001   000173 20L      0001   000261 2000L    0001   000614 2005L
0001   000621 2007L    0001   000641 2010L    0001   000644 2015L    0001   000662 2017L    0001   000716 2020L
0001   000724 2025L    0001   000764 2030L    0001   000771 2032L    0001   001014 2035L    0001   001017 2040L
0001   001040 2042L    0001   001077 2045L    0001   001105 2050L    0001   001150 2055L    0001   001176 2060L
0001   001202 2065L    0001   000214 207G     0001   000532 25L      0001   001206 2998L    0001   001234 2999L
0001   001270 30L      0001   001237 3000L    0001   001770 3005L    0001   001776 3007L    0001   002016 3010L
0001   002022 3015L    0001   002041 3017L    0001   002075 3020L    0001   002103 3025L    0001   002144 3030L
0001   002152 3032L    0001   002175 3035L    0001   002201 3040L    0001   002223 3042L    0001   002262 3045L
0001   002270 3050L    0001   002334 3055L    0001   002363 3060L    0001   002367 3065L    0001   001273 35L
0001   002373 3998L    0001   002421 3999L    0001   002424 4000L    0001   000324 4005L    0001   000331 4010L
0001   000332 4012L    0001   000351 4015L    0001   000352 4017L    0001   000367 4020L    0001   000403 4022L
0001   000420 4025L    0001   000426 4030L    0001   000464 4035L    0001   000500 4037L    0001   000515 4040L
0001   001424 434G     0001   000523 4998L    0001   000527 4999L    0001   000112 5L       0001   000532 5000L
0001   001475 5005L    0001   001503 5010L    0001   001504 5012L    0001   001524 5015L    0001   001525 5017L
0001   001542 5020L    0001   001556 5022L    0001   001573 5025L    0001   001601 5030L    0001   001637 5035L
0001   001653 5037L    0001   001670 5040L    0001   001335 55L      0001   001676 5998L    0001   001703 5999L
0001   001360 60L      0001   001706 6000L    0001   001367 65L      0001   002621 650G     0001   001402 70L
0001   002732 707G     0001   001412 75L      0001   001706 80L      0001   002455 85L      0000   000000 882F
0000   000001 883F     0000   000007 884F     0000   000020 885F     0000   000030 886F     0000   000035 887F
0000   000040 888F     0000   000043 889F     0001   002460 90L      0001   003101 994L     0001   003107 996L
0001   003116 997L     0001   003124 998L     0001   003132 999L
```

```
00101     1*       SUBROUTINE CIN SL
00101     2*   C   STEADY STATE EXECUTION SUBROUTINE FOR SINDA      FORTRAN V
00101     3*   C   THE LONG PSEUDO-COMPUTE SEQUENCE IS REQUIRED
00101     4*   C   DIFFUSION NODES RECEIVE A SUCCESSIVE POINT ITERATION
00101     5*   C   ARITHMETIC NODES RECEIVE A SUCESSIVE POINT ITERATION
00101     6*   C   OVER-RELAXATION IS ALLOWED, THE DAMPING FACTORS ARE ADDRESSABLE
00103     7*       INCLUDE COMM,LIST
00104     7*       COMMON /TITLE/H(1) /TEMP/T(1) /CAP/C(1) /SOURCE/Q(1) /COND/G(1)
00105     7*       COMMON /PC1/NSQ1(1) /PC2/NSQ2(1) /KONST/K(1) /ARRAY/A(1)
00106     7*       COMMON /FIXCON/KON(1) /XSPACE/NDIM,NTH,X(1)
00107     7*       COMMON /DIMENS/ NND,NNA,NNT,NGT,NCT,NAT,LSQ1,LSQ2
00110     7*       DIMENSION CON(1),XK(1),NX(1)
00111     7*       EQUIVALENCE (KON(1),CON(1)),(K(1),XK(1)),(X(1),NX(1))
00112     7*       END
00113     8*       INCLUDE DEFF,LIST
00113     8*   C*************  CONTROL CONSTANT DEFINITIONS AND NAMES  *****************
00113     8*   C   CONTROL CONSTANT 1 CONTAINS THE NEW PROBLEM TIME               (TIMEN)
00113     8*   C   CONTROL CONSTANT 2 CONTAINS THE TIME STEP USED               (DTIMEU)
00113     8*   C   CONTROL CONSTANT 3 CONTAINS THE PROBLEM STOP TIME            (TIMEND)
00113     8*   C   CONTROL CONSTANT 4 CONTAINS THE TIME STEP FACTOR,EXPLICIT (CSGFAC)
00113     8*   C   CC5 IS THE INPUT NUMBER OF ITERATION DO LOOPS, INTEGER        (NLOOP)
00113     8*   C   CC6 CONTAINS THE DIFFUSION TEMPERATURE CHANGE ALLOWED        (DTMPCA)
00113     8*   C   CC7 CONTAINS THE OUTPUT EACH ITERATION SWITCH                (OPEITR)
00113     8*   C   CC8 CONTAINS THE MAXIMUM ALLOWED TIME STEP                   (DTIMEH)
00113     8*   C   CC9 CONTAINS THE NEW ARITHMETIC TEMP. DAMPING FACTOR          (DAMPA)
00113     8*   C   CC10 CONTAINS THE NEW DIFFUSION TEMP. DAMPING FACTOR          (DAMPD)
00113     8*   C   CC11 CONTAINS THE MAXIMUM ALLOWED ARITHMETIC TEMP. CHANGE (ATMPCA)
00113     8*   C   CC12 CONTAINS THE BACKUP SWITCH CHECKED AFTER VARIABLES    (BACKUP)
00113     8*   C   CC13 CONTAINS THE PRESENT TIME OR PROBLEM START TIME          (TIMEO)
00113     8*   C   CC14 CONTAINS THE MEAN TIME BETWEEN AN ITERATION              (TIMEM)
00113     8*   C   CC15 CONTAINS THE DIFFUSION TEMPERATURE CHANGE CALCULATED (DTMPCC)
```

CINDSL,CINDSL

```
00113     8*    C      CC16 CONTAINS ARITHMETIC TEMPERATURE CHANGE CALCULATED     (ATMPCC)
00113     8*    C      CONTROL CONSTANT 17 IS RESERVED FOR THE C/SG MINIMUM       (CSGMIN)
00113     8*    C      CONTROL CONSTANT 18 CONTAINS THE OUTPUT INTERVAL           (OUTPUT)
00113     8*    C      CC19 CONTAINS THE ARITHMETIC RELAXATION CRITERIA ALLOWED   (ARLXCA)
00113     8*    C      CC20 CONTAINS THE NUMBER OF RELAXATION LOOPS USED,INTEGER  (LOOPCT)
00113     8*    C      CC21 CONTAINS THE MINIMUM ALLOWED TIME STEP                (DTIMEL)
00113     8*    C      CC22 IS FOR THE INPUT TIME STEP IMPLICIT                   (DTIMEI)
00113     8*    C      CC23 CONTAINS THE C/SG MAXIMUM                             (CSGMAX)
00113     8*    C      CC24 CONTAINS THE C/SG RANGE ALLOWED                       (CSGRAL)
00113     8*    C      CC25 CONTAINS THE C/SG RANGE CALCULATED                    (CSGRCL)
00113     8*    C      CC26 CONTAINS THE DIFFUSION RELAXATION CRITERIA ALLOWED    (DRLXCA)
00113     8*    C      CC27 CONTAINS THE DIFFUSION RELAXATION CHANGE CALCULATED   (DRLXCC)
00113     8*    C      CC28 CONTAINS THE LINE COUNTER, INTEGER                    (LINECT)
00113     8*    C      CC29 CONTAINS THE PAGE COUNTER, INTEGER                    (PAGECT)
00113     8*    C      CC30 CONTAINS ARITHMETIC RELAXATION CHANGE CALCULATED      (APLXCC)
00113     8*    C      CC31 IS INDICATOR, 0=THERMAL SPCS,1=THERMAL LPCS,2=GENERAL (LSPCS)
00113     8*    C      CC32 CONTAINS THE ENERGY BALANCE OF THE SYSTEM, IN - OUT   (ENGBAL)
00113     8*    C      CC33 CONTAINS THE DESIRED ENERGY BALANCE, USER INPUT       (BALENG)
00113     8*    C      CC34 CONTAINS THE NOCOPY SWITCH FOR MATRIX USERS           (NOCOPY)
00113     8*    C      CC35 CONTAINS RELATIVE NODE NUMBER OF CSGMIN
00113     8*    C      CC36 CONTAINS RELATIVE NODE NUMBER OF DTMPCC
00113     8*    C      CC37 CONTAINS RELATIVE NODE NUMBER OF ARLXCC
00113     8*    C      CC38 CONTAINS RELATIVE NODE NUMBER OF ATMPCC
00113     8*    C      CC39-40-41-42-43 CONTAIN DUMMY INTEGER CONSTANTS   (I-J-K-L-MTEST)
00113     8*    C      CC44-45-46-47-48 CONTAIN DUMMY FLOATING CONSTANTS  (R-S-T-U-VTEST)
00113     8*    C      CC49 IS THE QUASI-LINEARIZATION INTERVAL FOR CINDSM        (LAXFAC)
00113     8*    C      CC50 IS NOT USED AT PRESENT
00114     8*           END
00115     9*           IF(KON(5).LE.0) GO TO 999
00117    10*           IF(CON(9).LE.0.) CON(9) = 1.0
00121    11*           IF(CON(10).LE.0.) CON(10) = 1.0
00123    12*           IF(NNA.GT.0.AND.CON(19).LE.0.) GO TO 998
00125    13*           IF(NND.GT.0.AND.CON(26).LE.0.) GO TO 997
00127    14*           IF(KON(31).NE.1) GO TO 994
00131    15*           PASS = -1.0
00132    16*           NN = NND+1
00133    17*           NNC = NNA+NND
00134    18*           IE1 = NTH
00135    19*           IE2 = IE1+NNC
00136    20*           NLA = NDIM
00137    21*           JJ = 2*NNC
00140    22*           NTH = NTH+JJ
00141    23*           NDIM = NDIM-JJ
00142    24*           IF(NDIM.LT.0) GO TO 996
00144    25*           CON(1) = CON(13)
00145    26*           CON(2) = 0.0
00146    27*           CON(14) = CON(13)
00147    28*           GO TO 10
00150    29*         5 CON(1) = CON(13)+CON(18)
00151    30*           IF(CON(1)-CON(3).GT.0.) CON(1) = CON(3)
00153    31*           CON(14) = (CON(1)+CON(13))/2.0
00154    32*           CON(2) = CON(1)-CON(13)
00154    33*    C      COMPUTE STEADY STATE TEMPERATURES
00155    34*        10 LAX = KON(5)
00156    35*           JJ = 0
00157    36*           DO 145 K1 = 1,LAX
00162    37*           JJ = JJ+1
00163    38*           KON(20) = K1
00163    39*    C      ZERO OUT ALL SOURCE LOCATIONS
00164    40*           DO 15 I = 1,NNC
```

```
00167   41*      15 Q(I) = 0.0
00171   42*         CALL VARBL1
00172   43*         IF(PASS.GE.0.) GO TO 20
00174   44*         CALL OUTCAL
00175   45*         PASS = 1.0
00176   46*      20 J1 = 0
00177   47*         J2 = 1
00200   48*         RLXD = 0.0
00201   49*         RLXA = 0.0
00202   50*         IF(NND.LE.0) GO TO 75
00204   51*         DN = CON(10)
00205   52*         DD = 1.0-DN
00205   53*   C     DO A SUCCESSIVE POINT ITERATION ON THE DIFFUSION NODES
00206   54*         DO 70 I = 1,NND
00211   55*         GSUM = 0.0
00212   56*         INCLUDE DUMC,LIST
00213   56*         IF(FLD(1,1,NSQ1(J1+1)).EQ.0) GO TO 2000
00215   56*         NTYPE = FLD(0,5,NSQ2(J2))                                          *NEW
00216   56*         GO TO (1999,1998,1998,1998,1999,1998,1998,1998,1999), NTYPE        **-1
00217   56*    1998 J2 = J2+1
00220   56*    1999 J2 = J2+1
00221   56*    2000 CONTINUE
00222   56*         END
00223   57*         INCLUDE VARQ,LIST
00224   57*         IF(FLD(4,1,NSQ1(J1+1)).EQ.0) GO TO 5000
00226   57*         NTYPE = FLD(0,5,NSQ2(J2))                                          *NEW
00227   57*         LA = FLD(5,17,NSQ2(J2))                                            *NEW
00230   57*         LK = FLD(22,14,NSQ2(J2))                                           *NEW
00231   57*         GO TO (4005,4010,4015,4020,4025,4030,4035,4040,4030), NTYPE        **-3
00232   57*    4005 Q(I) = XK(LK)+Q(I)
00233   57*         GO TO 4999
00234   57*    4010 Q1 = 0.0
00235   57*    4012 CALL D1D1WM(T(I),A(LA),XK(LK),Q2)
00236   57*         GO TO 4998
00237   57*    4015 Q1 = 0.0
00240   57*    4017 CALL D1D1WM(CON(14),A(LA),XK(LK),Q2)
00241   57*         GO TO 4998
00242   57*    4020 CALL D1D1WM(CON(14),A(LA),XK(LK),Q1)
00243   57*    4022 J2 = J2+1
00244   57*         LA = FLD(5,17,NSQ2(J2))                                            *NEW
00245   57*         LK = FLD(22,14,NSQ2(J2))                                           *NEW
00246   57*         GO TO 4017                                                         **-2
00247   57*    4025 Q1 = XK(LK)*XK(LA)
00250   57*         GO TO 4022
00251   57*    4030 CALL D1D1WM(CON(14),A(LA),XK(LK),Q1)
00252   57*         J2 = J2+1
00253   57*         LA = FLD(5,17,NSQ2(J2))                                            *NEW
00254   57*         LK = FLD(22,14,NSQ2(J2))                                           *NEW
00255   57*         Q2 = XK(LK)*XK(LA)                                                 **-2
00256   57*         GO TO 4998
00257   57*    4035 CALL D1D1WM(CON(14),A(LA),XK(LK),Q1)
00260   57*    4037 J2 = J2+1
00261   57*         LA = FLD(5,17,NSQ2(J2))                                            *NEW
00262   57*         LK = FLD(22,14,NSQ2(J2))                                           *NEW
00263   57*         GO TO 4012                                                         **-2
00264   57*    4040 Q1 = XK(LK)*XK(LA)
00265   57*         GO TO 4037
00266   57*    4998 Q(I) = Q1+Q2+Q(I)
00267   57*    4999 J2 = J2+1
00270   57*    5000 CONTINUE
```

```
00271    57*          END
00272    58*       25 J1 = J1+1
00273    59*          LG = FLD(5,16,NSQ1(J1))
00274    60*          LTA = FLD(22,14,NSQ1(J1))
00275    61*          INCLUDE VARG,LIST
00275    62*    C     CHECK FOR RADIATION CONDUCTOR
00276    62*          IF(FLD(2,1,NSQ1(J1)).EQ.0) GO TO 3000
00300    62*          NTYPE = FLD(0,5,NSQ2(J2))                                         *NEW
00301    62*          LA = FLD(5,17,NSQ2(J2))                                           *NEW
00302    62*          LK = FLD(22,14,NSQ2(J2))                                          *NEW
00303    62*          GOTO(2005,2010,2015,2020,2025,2030,2035,2040,2045,2050,2055,      **-3
00303    62*         $     2060,2065), NTYPE
00304    62*     2005 TM = (T(I)+T(LTA))/2.0
00305    62*     2007 CALL D1D1WM(TM,A(LA),XK(LK),G(LG))
00306    62*          GO TO 2999
00307    62*     2010 TM = T(I)
00310    62*          GO TO 2007
00311    62*     2015 CALL D1D1WM(T(I),A(LA),XK(LK),G1)
00312    62*     2017 J2 = J2+1
00313    62*          LA = FLD(5,17,NSQ2(J2))                                           *NEW
00314    62*          LK = FLD(22,14,NSG2(J2))                                          *NEW
00315    62*          CALL D1D1WM(T(LTA),A(LA),XK(LK),G2)                               **-2
00316    62*          GO TO 2998
00317    62*     2020 G1 = XK(LK)*XK(LA)
00320    62*          GO TO 2017
00321    62*     2025 CALL D1D1WM(T(I),A(LA),XK(LK),G1)
00322    62*          J2 = J2+1
00323    62*          LA = FLD(5,17,NSQ2(J2))                                           *NEW
00324    62*          LK = FLD(22,14,NSQ2(J2))                                          *NEW
00325    62*          G2 = XK(LK)*XK(LA)                                                **-2
00326    62*          GO TO 2998
00327    62*     2030 TM = (T(I)+T(LTA))/2.0
00330    62*     2032 CALL PLYAWM(A(LA),TM,A(LA+1),XK(LK),G(LG))
00331    62*          GO TO 2999
00332    62*     2035 TM = T(I)
00333    62*          GO TO 2032
00334    62*     2040 CALL PLYAWM(A(LA),T(I),A(LA+1),XK(LK),G1)
00335    62*     2042 J2 = J2+1
00336    62*          LA = FLD(5,17,NSQ2(J2))                                           *NEW
00337    62*          LK = FLD(22,14,NSQ2(J2))                                          *NEW
00340    62*          CALL PLYAWM(A(LA),T(LTA),A(LA+1),XK(LK),G2)                       **-2
00341    62*          GO TO 2998
00342    62*     2045 G1 = XK(LK)*XK(LA)
00343    62*          GO TO 2042
00344    62*     2050 CALL PLYAWM(A(LA),T(I),A(LA+1),XK(LK),G1)
00345    62*          J2 = J2+1
00346    62*          LA = FLD(5,17,NSQ2(J2))                                           *NEW
00347    62*          LK = FLD(22,14,NSQ2(J2))                                          *NEW
00350    62*          G2 = XK(LK)*XK(LA)                                                **-2
00351    62*          GO TO 2998
00352    62*     2055 TM = (T(I)+T(LTA))/2.0
00353    62*          CALL D2D1WM(TM,CON(J4),A(LA),XK(LK),G(LG))
00354    62*          GO TO 2999
00355    62*     2060 TM = T(LTA)
00356    62*          GO TO 2007
00357    62*     2065 TM = T(LTA)
00360    62*          GO TO 2032
00361    62*     2998 G(LG) = 1./(1./G1+1./G2)
00362    62*          IF(FLD(3,1,NSQ1(J1)).EQ.1) G(LG) = G1*G2
00364    62*     2999 J2 = J2+1
```

```
00365    62*    3000 CONTINUE
00366    62*         END
00367    63*         IF(FLD(3,1,NSQ1(J1)).EQ.0) GO TO 30
00371    64*         T1 = T(I)+460.0
00372    65*         T2 = T(LTA)+460.0
00373    66*         GV = G(LG)*(T1*T1+T2*T2)*(T1+T2)
00374    67*         GO TO 35
00375    68*      30 GV = G(LG)
00376    69*      35 GSUM = GSUM+GV
00377    70*         Q(I) = Q(I)+GV*T(LTA)
00377    71*   C     CHECK FOR LAST CONDUCTOR TO THIS NODE
00400    72*         IF(NSQ1(J1).GT.0) GO TO 25
00402    73*         T2 = DD*T(I)+DN*Q(I)/GSUM
00402    74*   C     OBTAIN THE CALCULATED TEMPERATURE DIFFERENCE
00403    75*         T1 = ABS(T(I)-T2)
00403    76*   C     STORE THE NEW TEMPERATURES AND EXTRAPOLATION FACTORS
00404    77*         GO TO(65,60,55),JJ
00405    78*      55 LE1 = IE1+I
00406    79*         LE2 = IE2+I
00407    80*         R1 = T2-T(I)
00410    81*         X(LE1) = T(I)
00411    82*         X(LE2) = R1/(R1-X(LE2))
00412    83*         GO TO 65
00413    84*      60 LE2 = IE2+I
00414    85*         X(LE2) = T2-T(1)
00415    86*      65 T(I) = T2
00415    87*   C     SAVE THE MAXIMUM DIFFUSION RELAXATION CHANGE
00416    88*         IF(RLXD.GT.T1) GO TO 70
00420    89*         RLXD = T1
00421    90*         N1 = I
00422    91*      70 CONTINUE
00424    92*         CON(27) = RLXD
00425    93*         IF(NNA.LE.0) GO TO 130
00427    94*      75 DN = CON(9)
00430    95*         DD = 1.0-DN
00431    96*         JJ1 = J1
00432    97*         JJ2 = J2
00432    98*   C     DO SUCCESSIVE POINT ITERATION ON ARITHMETIC NODES
00433    99*         DO 125 I = NN,NNC
00436   100*         GSUM = 0.0
00437   101*         L = I
00440   102*         INCLUDE VRQ2,LIST
00441   102*         IF(FLD(4,1,NSQ1(JJ1+1)).EQ.0) GO TO 6000
00443   102*         NTYPE = FLD(0,5,NSQ2(JJ2))                                        *NEW
00444   102*         LA = FLD(5,17,NSQ2(JJ2))                                          *NEW
00445   102*         LK = FLD(22,14,NSQ2(JJ2))                                         *NEW
00446   102*         GO TO (5005,5010,5015,5020,5025,5030,5035,5040,5030), NTYPE       **-3
00447   102*    5005 Q(L) = XK(LK)+Q(L)
00450   102*         GO TO 5999
00451   102*    5010 Q1 = 0.0
00452   102*    5012 CALL D1D1WM(T(L),A(LA),XK(LK),Q2)
00453   102*         GO TO 5998
00454   102*    5015 Q1 = 0.0
00455   102*    5017 CALL D1D1WM(CON(14),A(LA),XK(LK),Q2)
00456   102*         GO TO 5998
00457   102*    5020 CALL D1D1WM(CON(14),A(LA),XK(LK),Q1)
00460   102*    5022 JJ2 = JJ2+1
00461   102*         LA = FLD(5,17,NSQ2(JJ2))                                          *NEW
00462   102*         LK = FLD(22,14,NSQ2(JJ2))                                         *NEW
00463   102*         GO TO 5017                                                        **-2
```

CINDSL,CINDSL

```
102*      5025 Q1 = XK(LK)*XK(LA)
102*           GO TO 5022
102*      5030 CALL D1D1WM(CON(14),A(LA),XK(LK),Q1)
102*           JJ2 = JJ2+1
102*           LA = FLD(5,17,NSQ2(JJ2))                                  *NEW
102*           LK = FLD(22,14,NSQ2(JJ2))                                 *NEW
102*           Q2 = XK(LK)*XK(LA)                                        **-2
102*           GO TO 5998
102*      5035 CALL D1D1WM(CON(14),A(LA),XK(LK),Q1)
102*      5037 JJ2 = JJ2+1
102*           LA = FLD(5,17,NSQ2(JJ2))                                  *NEW
102*           LK = FLD(22,14,NSQ2(JJ2))                                 *NEW
102*           GO TO 5012                                                **-2
102*      5040 Q1 = XK(LK)*XK(LA)
102*           GO TO 5037
102*      5998 Q(L) = Q1+Q2+Q(L)
102*      5999 JJ2 = JJ2+1
102*      6000 CONTINUE
102*           END
103*        80 JJ1 = JJ1+1
104*           LG = FLD(5,16,NSQ1(JJ1))
105*           LTA = FLD(22,14,NSQ1(JJ1))
106*           INCLUDE VRG2,LIST
107*   C       CHECK FOR RADIATION CONDUCTOR
107*           IF(FLD(2,1,NSQ1(JJ1)).EQ.0) GO TO 4000
107*           NTYPE = FLD(0,5,NSQ2(JJ2))                                *NEW
107*           LA = FLD(5,17,NSQ2(JJ2))                                  *NEW
107*           LK = FLD(22,14,NSQ2(JJ2))                                 *NEW
107*           GOTO(3005,3010,3015,3020,3025,3030,3035,3040,3045,3050,3055,  **-3
107*          $     3060,3065), NTYPE
107*      3005 TM = (T(L)+T(LTA))/2.0
107*      3007 CALL D1D1WM(TM,A(LA),XK(LK),G(LG))
107*           GO TO 3999
107*      3010 TM = T(L)
107*           GO TO 3007
107*      3015 CALL D1D1WM(T(L),A(LA),XK(LK),G1)
107*      3017 JJ2 = JJ2+1
107*           LA = FLD(5,17,NSQ2(JJ2))                                  *NEW
107*           LK = FLD(22,14,NSQ2(JJ2))                                 *NEW
107*           CALL D1D1WM(T(LTA),A(LA),XK(LK),G2)                       **-2
107*           GO TO 3998
107*      3020 G1 = XK(LK)*XK(LA)
107*           GO TO 3017
107*      3025 CALL D1D1WM(T(L),A(LA),XK(LK),G1)
107*           JJ2 = JJ2+1
107*           LA = FLD(5,17,NSQ2(JJ2))                                  *NEW
107*           LK = FLD(22,14,NSQ2(JJ2))                                 *NEW
107*           G2 = XK(LK)*XK(LA)                                        **-2
107*           GO TO 3998
107*      3030 TM = (T(L)+T(LTA))/2.0
107*      3032 CALL PLYAWM(A(LA),TM,A(LA+1),XK(LK),G(LG))
107*           GO TO 3999
107*      3035 TM = T(L)
107*           GO TO 3032
107*      3040 CALL PLYAWM(A(LA),T(L),A(LA+1),XK(LK),G1)
107*      3042 JJ2 = JJ2+1
107*           LA = FLD(5,17,NSQ2(JJ2))                                  *NEW
107*           LK = FLD(22,14,NSQ2(JJ2))                                 *NEW
107*           CALL PLYAWM(A(LA),T(LTA),A(LA+1),XK(LK),G2)               **-2
107*           GO TO 3998
```

CINDSL,CINDSL

```
00557  107*   3045 G1 = XK(LK)*XK(LA)
00560  107*        GO TO 3042
00561  107*   3050 CALL PLYAWM(A(LA),T(L),A(LA+1),XK(LK),G1)
00562  107*        JJ2 = JJ2+1
00563  107*        LA = FLD(5,17,NSQ2(JJ2))                                    *NEW
00564  107*        LK = FLD(22,14,NSQ2(JJ2))                                   *NEW
00565  107*        G2 = XK(LK)*XK(LA)                                          **-2
00566  107*        GO TO 3998
00567  107*   3055 TM = (T(L)+T(LTA))/2.0
00570  107*        CALL D2D1WM(TM,CON(14),A(LA),XK(LK),G(LG))
00571  107*        GO TO 3999
00572  107*   3060 TM = T(LTA)
00573  107*        GO TO 3007
00574  107*   3065 TM = T(LTA)
00575  107*        GO TO 3032
00576  107*   3998 G(LG) = 1./(1./G1+1./G2)
00577  107*        IF(FLD(3,1,NSQ1(JJ1)).EQ.1) G(LG) = G1*G2
00601  107*   3999 JJ2 = JJ2+1
00602  107*   4000 CONTINUE
00603  107*        END
00604  108*        IF(FLD(3,1,NSQ1(JJ1)).EQ.0) GO TO 85
00606  109*        T1 = T(I)+460.0
00607  110*        T2 = T(LTA)+460.0
00610  111*        GV = G(LG)*(T1*T1+T2*T2)*(T1+T2)
00611  112*        GO TO 90
00612  113*     85 GV = G(LG)
00613  114*     90 Q(I) = Q(I)+GV*T(LTA)
00614  115*        GSUM = GSUM+GV
00614  116*   C    CHECK FOR LAST CONDUCTOR TO THIS NODE
00615  117*        IF(NSQ1(JJ1).GT.0) GO TO 80
00617  118*        T2 = DD*T(I)+DN*Q(I)/GSUM
00620  119*        T1 = ABS(T(I)-T2)
00620  120*   C    STORE THE NEW TEMPERATURES AND EXTRAPOLATION FACTORS
00621  121*        GO TO(120,115,110),JJ
00622  122*    110 LE1 = IE1+I
00623  123*        LE2 = IE2+I
00624  124*        R1 = T2-T(I)
00625  125*        X(LE1) = T(I)
00626  126*        X(LE2) = R1/(R1-X(LE2))
00627  127*        GO TO 120
00630  128*    115 LE2 = IE2+I
00631  129*        X(LE2) = T2-T(I)
00632  130*    120 T(I) = T2
00633  131*        IF(RLXA.GT.T1) GO TO 125
00635  132*        RLXA = T1
00636  133*        N2 = I
00637  134*    125 CONTINUE
00641  135*        CON(30) = RLXA
00641  136*   C    SEE IF THE RELAXATION CRITERIA ARE MET
00642  137*    130 IF(RLXA.LE.CON(19).AND.RLXD.LE.CON(26)) GO TO 150
00644  138*        IF(JJ.LE.2) GO TO 140
00646  139*        JJ = 0
00647  140*        DO 135 I = 1,NNC
00652  141*        LE2 = IE2+I
00652  142*   C    SEE IF THE EXTRAPOLATION CRITERIA ARE MET
00653  143*        IF(X(LE2).GE.0.) GO TO 135
00655  144*        IF(X(LE2).LT.-8.) X(LE2) = -8.
00657  145*        LE1 = IE1+I
00660  146*        T(I) = X(LE2)*X(LE1)+(1.0-X(LE2))*T(I)
00661  147*    135 CONTINUE
```

CINDSL,CINDSL

```
00663      140 IF(KON(7).NE.0) CALL OUTCAL
00665      145 CONTINUE
00667          WRITE(6,882)
00671          WRITE(6,885) LAX
00674          KON(28) = KON(28)+2
00675      150 KON(37) = N2
00676          IF(RLXA.GT.RLXD) GO TO 155
00700          CON(30) = RLXD
00701          KON(37) = N1
00701    C     CHECK THE ENERGY BALANCE OF THE SYSTEM
00702      155 CALL NONLIN
00703          QOUT = 0.0
00704          QIN = 0.0
00705          J1 = 0
00706          DO 195 I = 1,NNC
00711          QIN = QIN+Q(I)
00712      165 J1 = J1+1
00713          LTA = FLD(22,14,NSQ1(J1))
00714          IF(LTA.LE.NNC) GO TO 175
00716          LG = FLD(5,16,NSQ1(J1))
00717          IF(FLD(3,1,NSQ1(J1)).EQ.0) GO TO 170
00721          T1 = T(I)+460.0
00722          T2 = T(LTA)+460.0
00723          QOUT = QOUT+G(LG)*(T1**4-T2**4)
00724          GO TO 175
00725      170 QOUT = QOUT+G(LG)*(T(I)-T(LTA))
00725    C     CHECK FOR LAST CONDUCTOR TO THIS NODE
00726      175 IF(NSQ1(J1).GT.0) GO TO 165
00730      195 CONTINUE
00732          CON(32) = ABS(QIN-QOUT)
00733          CALL VARBL2
00734          CON(13) = CON(1)
00735          CALL OUTCAL
00736          WRITE(6,882)
00740          WRITE(6,883) KON(20),CON(32)
00744          KON(28) = KON(28)+2
00745          IF(CON(3).GT.CON(1)*1.000001) GO TO 5
00747          NTH = IE1
00750          NDIM = NLA
00751          RETURN
00752      994 WRITE(6,884)
00754          GO TO 1000
00755      996 WRITE(6,886) NDIM
00760          GO TO 1000
00761      997 WRITE(6,887)
00763          GO TO 1000
00764      998 WRITE(6,888)
00766          GO TO 1000
00767      999 WRITE(6,889)
00771     1000 CALL OUTCAL
00772          CALL EXIT
00773      882 FORMAT(1H )
00774      883 FORMAT(10H LOOPCT = I6,10H ENGBAL = E12.5)
00775      884 FORMAT(45H CINDSL REQUIRES LONG PSEUDO-COMPUTE SEQUENCE)
00776      885 FORMAT(35H ITERATION COUNT EXCEEDED, LOOPCT =,  I10)
00777      886 FORMAT(I8,20H LOCATIONS AVAILABLE)
01000      887 FORMAT(10H NO DRLXCA)
01001      888 FORMAT(10H NO ARLXCA)
01002      889 FORMAT(14H NO LOOP COUNT)
01003          END
```

```
 @I   FOR,*   CINDSM                                                                      21 DEC 70
 UNIVAC 1108 FORTRAN V ATHENA VERSION 131K-10D   CREATED ON 20 AUG 70
 THIS COMPILATION WAS DONE ON 09 JUN 70 AT 22:16:55

   SUBROUTINE CINDSM    ENTRY POINT 001175

   STORAGE USED (BLOCK, NAME, LENGTH)

         0001    *CODE          001211
         0000    *CONST+TEMP    000067
         0002    *SIMPLE VAR    000044
         0004    *ARRAYS        000000
         0005    *BLANK         000000
         0006    TITLE    000001
         0007    TEMP     000001
         0010    CAP      000001
         0011    SOURCE   000001
         0012    COND     000001
         0013    PC1      000001
         0014    PC2      000001
         0015    KONST    000001
         0016    ARRAY    000001
         0017    FIXCON   000001
         0020    XSPACE   000003
         0021    DIMENS   000010

   EXTERNAL REFERENCES (BLOCK, NAME)

         0022    NONLIN
         0023    OUTCAL
         0024    VARBL2
         0025    EXIT
         0026    NERR2$
         0027    NWDU$
         0030    NIO2$
         0031    NER10$

   STORAGE ASSIGNMENT FOR VARIABLES (BLOCK, TYPE, RELATIVE LOCATION, NAME)

    0016 R 000000 A        0010 R 000000 C        0017 R 000000 CON      0002 R 000016 DAMP     0002 R 000013 DELXXX
    0012 R 000000 G        0002 R 000031 GSUM     0006 R 000000 H        0002 I 000022 I        0002 I 000004 IE1
    0002 I 000005 IE2      0002 I 000006 IE3      0002 I 000007 IE4      0002 I 000002 JJ       0002 I 000021 J1
    0015 I 000000 K        0002 I 000033 KK       0017 I 000000 KON      0002 I 000032 K1       0002 I 000010 LAXFAC
    0002 I 000036 LE1      0002 I 000037 LE2      0002 I 000023 LE3      0002 I 000025 LE4      0002 I 000024 LG
    0002 I 000000 LPASS    0021 I 000006 LSQ1     0021 I 000007 LSQ2     0002 I 000026 LTA      0021 I 000005 NAT
    0021 I 000004 NCT      0020 I 000000 NDIM     0021 I 000003 NGT      0002 I 000003 NLA      0002 I 000011 NLAX
    0021 I 000001 NNA      0002 I 000001 NNC      0021 I 000000 NND      0021 I 000002 NNT      0013 I 000000 NSQ1
    0014 I 000000 NSQ2     0020 I 000001 NTH      0020   000002 NX       0002 I 000041 N1       0002 I 000042 N2
    0011 R 000000 Q        0002 R 000020 QIN      0002 R 000043 QOUT     0002 R 000035 QSUM     0002 R 000015 RELAX
    0002 R 000017 RLX      0002 R 000034 RLXD     0002 R 000040 R1       0007 R 000000 T        0002 R 000027 T1
    0002 R 000030 T2       0020 R 000002 X        0015   000000 XK       0002 R 000012 XXX      0002 R 000014 XXXDUM
    0001   000116 10L      0001   001152 1000L    0001   000571 105L     0001   000644 120L     0001   000712 128L
    0001   000716 130L     0001   000727 135L     0001   001005 140L     0001   001014 145L     0001   000135 15L
    0001   001041 155L     0001   001053 160L     0001   000177 204G     0001   000300 231G     0001   000320 242G
    0001   000157 25L      0001   000332 251G     0001   000162 30L      0001   000507 320G     0001   000603 341G
```

```
                   CINDSM

 0001   000211 35L     0001   000623 3516     0001   000663 3706     0001   000261 40L      0001   000727 410G
 0001   000265 45L     0001   000074 5L       0001   000336 60L      0001   000410 65L      0001   000437 70L
 0001   000446 75L     0001   000455 80L      0001   000470 85L      0000   000000 884F     0000   000006 885F
 0000   000013 886F    0000   000016 887F     0000   000021 888F     0000   000026 889F     0001   000555 90L
 0001   000557 95L     0001   001114 995L     0001   001123 996L     0001   001131 997L     0001   001137 998L
 0001   001145 999L


00101    1*        SUBROUTINE CINDSM
00101    2*   C    STEADY STATE EXECUTION SUBROUTINE FOR SINDA, FORTRAN V
00101    3*   C    THE LONG PSEUDO-COMPUTE SEQUENCE IS REQUIRED
00101    4*   C    DIFFUSION NODES RECEIVE A SUCCESSIVE POINT ITERATION
00101    5*   C    ARITHMETIC NODES RECEIVE A SUCESSIVE POINT ITERATION
00101    6*   C    A QUASI-LINEAR BLOCK ITERATION IS PERFORMED ON RADIATION
00103    7*        INCLUDE COMM,LIST
00104    7*        COMMON /TITLE/H(1) /TEMP/T(1) /CAP/C(1) /SOURCE/Q(1) /COND/G(1)
00105    7*        COMMON /PC1/NSQ1(1) /PC2/NSQ2(1) /KONST/K(1) /ARRAY/A(1)
00106    7*        COMMON /FIXCON/KON(1) /XSPACE/NDIM,NTH,X(1)
00107    7*        COMMON /DIMENS/ NND,NNA,NNT,NGT,NCT,NAT,LSQ1,LSQ2
00110    7*        DIMENSION CON(1),XK(1),NX(1)
00111    7*        EQUIVALENCE (KON(1),CON(1)),(K(1),XK(1)),(X(1),NX(1))
00112    7*        END
00113    8*        INCLUDE DEFF,LIST
00113    9*   C    CC49 CONTAINS THE NO. OF ITER. BEFORE RAD.COND. LINEAR.   (LAXFAC)
00113    9*   C*************  CONTROL CONSTANT DEFINITIONS AND NAMES  ****************
00113    9*   C    CONTROL CONSTANT 1 CONTAINS THE NEW PROBLEM TIME           (TIMEN)
00113    9*   C    CONTROL CONSTANT 2 CONTAINS THE TIME STEP USED            (DTIMEU)
00113    9*   C    CONTROL CONSTANT 3 CONTAINS THE PROBLEM STOP TIME         (TIMEND)
00113    9*   C    CONTROL CONSTANT 4 CONTAINS THE TIME STEP FACTOR,EXPLICIT (CSGFAC)
00113    9*   C    CC5 IS THE INPUT NUMBER OF ITERATION DO LOOPS, INTEGER     (NLOOP)
00113    9*   C    CC6 CONTAINS THE DIFFUSION TEMPERATURE CHANGE ALLOWED     (DTMPCA)
00113    9*   C    CC7 CONTAINS THE OUTPUT EACH ITERATION SWITCH             (OPEITR)
00113    9*   C    CC8 CONTAINS THE MAXIMUM ALLOWED TIME STEP                (DTIMEH)
00113    9*   C    CC9 CONTAINS THE NEW ARITHMETIC TEMP. DAMPING FACTOR       (DAMPA)
00113    9*   C    CC10 CONTAINS THE NEW DIFFUSION TEMP. DAMPING FACTOR       (DAMPD)
00113    9*   C    CC11 CONTAINS THE MAXIMUM ALLOWED ARITHMETIC TEMP. CHANGE (ATMPCA)
00113    9*   C    CC12 CONTAINS THE BACKUP SWITCH CHECKED AFTER VARIABLES   (BACKUP)
00113    9*   C    CC13 CONTAINS THE PRESENT TIME OR PROBLEM START TIME       (TIMEO)
00113    9*   C    CC14 CONTAINS THE MEAN TIME BETWEEN AN ITERATION           (TIMEM)
00113    9*   C    CC15 CONTAINS THE DIFFUSION TEMPERATURE CHANGE CALCULATED (DTMPCC)
00113    9*   C    CC16 CONTAINS ARITHMETIC TEMPERATURE CHANGE CALCULATED    (ATMPCC)
00113    9*   C    CONTROL CONSTANT 17 IS RESERVED FOR THE C/SG MINIMUM      (CSGMIN)
00113    9*   C    CONTROL CONSTANT 18 CONTAINS THE OUTPUT INTERVAL          (OUTPUT)
00113    9*   C    CC19 CONTAINS THE ARITHMETIC RELAXATION CRITERIA ALLOWED  (ARLXCA)
00113    9*   C    CC20 CONTAINS THE NUMBER OF RELAXATION LOOPS USED,INTEGER (LOOPCT)
00113    9*   C    CC21 CONTAINS THE MINIMUM ALLOWED TIME STEP               (DTIMEL)
00113    9*   C    CC22 IS FOR THE INPUT TIME STEP IMPLICIT                  (DTIMEI)
00113    9*   C    CC23 CONTAINS THE C/SG MAXIMUM                            (CSGMAX)
00113    9*   C    CC24 CONTAINS THE C/SG RANGE ALLOWED                      (CSGRAL)
00113    9*   C    CC25 CONTAINS THE C/SG RANGE CALCULATED                   (CSGRCL)
00113    9*   C    CC26 CONTAINS THE DIFFUSION RELAXATION CRITERIA ALLOWED   (DRLXCA)
00113    9*   C    CC27 CONTAINS THE DIFFUSION RELAXATION CHANGE CALCULATED  (DRLXCC)
00113    9*   C    CC28 CONTAINS THE LINE COUNTER, INTEGER                   (LINECT)
00113    9*   C    CC29 CONTAINS THE PAGE COUNTER, INTEGER                   (PAGECT)
00113    9*   C    CC30 CONTAINS ARITHMETIC RELAXATION CHANGE CALCULATED     (ARLXCC)
00113    9*   C    CC31 IS INDICATOR, 0=THERMAL SPCS,1=THERMAL LPCS,2=GENERAL (LSPCS)
00113    9*   C    CC32 CONTAINS THE ENERGY BALANCE OF THE SYSTEM, IN - OUT  (ENGBAL)
00113    9*   C    CC33 CONTAINS THE DESIRED ENERGY BALANCE, USER INPUT      (BALENG)
```

```
C     CC34 CONTAINS THE NOCOPY SWITCH FOR MATRIX USERS          (NOCOPY)
C     CC35 CONTAINS RELATIVE NODE NUMBER OF CSGMIN
C     CC36 CONTAINS RELATIVE NODE NUMBER OF DTMPCC
C     CC37 CONTAINS RELATIVE NODE NUMBER OF ARLXCC
C     CC38 CONTAINS RELATIVE NODE NUMBER OF ATMPCC
C     CC39-40-41-42-43 CONTAIN DUMMY INTEGER CONSTANTS    (I-J-K-L-MTEST)
C     CC44-45-46-47-48 CONTAIN DUMMY FLOATING CONSTANTS  (R-S-T-U-VTEST)
C     CC49 IS THE QUASI-LINEARIZATION INTERVAL FOR CINDSM       (LAXFAC)
C     CC50 IS NOT USED AT PRESENT
      END
      IF(KON(5).LE.0) GO TO 999
      IF(CON(9).LE.0.0) CON(9) = 1.0
      IF(CON(10).LE.0.0) CON(10) = 1.0
      IF(KON(31).NE.1) GO TO 998
      IF(CON(33).LE.0.0) GO TO 997
      IF(KON(49).LE.0) GO TO 996
      LPASS = -1
      NNC = NNA+NND
      JJ = 3*NNC+NGT
      NLA = NDIM
      NDIM = NDIM-JJ
      IF(NDIM.LT.0) GO TO 995
      IE1 = NTH
      NTH = NTH+JJ
      IE2 = IE1+NNC
      IE3 = IE2+NNC
      IE4 = IE3+NNC
      CON(1) = CON(13)
      CON(2) = 0.0
      CON(14) = CON(13)
      GO TO 10
    5 CON(1) = CON(13)+CON(18)
      IF(CON(1)-CON(3).GT.0.0) CON(1) = CON(3)
      CON(14) = (CON(1)+CON(13))/2.0
      CON(2) = CON(1)-CON(13)
      LPASS = 0
   10 CONTINUE
      KON(20) = 0
      LAXFAC = KON(49)
      NLAX = KON(5)/LAXFAC
      XXX = 0.05
      DELXXX = 0.05/FLOAT(NLAX)
      XXXDUM = 0.001
C     EVALUATE NONLINEARITIES
   15 CALL NONLIN
      RELAX = XXX
      DAMP = CON(10)
      RLX = 1.0-DAMP
      IF(LPASS.GT.-1) GO TO 25
      CALL OUTCAL
      LPASS = 1
      GO TO 30
   25 LPASS = LPASS+1
   30 CONTINUE
C     SUM ALL SOURCES, SAVE ALL TEMPERATURES AND LINEARIZE RADIATION
      QIN = 0.0
      J1 = 0
      DO 50 I = 1,NNC
      QIN = QIN+Q(I)
      LE3 = IE3+I
```

```
00210              X(LE3) = T(I)
00211           35 J1 = J1+1
00212              LG = FLD(5,16,NSQ1(J1))
00213              LE4 = IE4+LG
00213       C      CHECK FOR RADIATION CONDUCTOR
00214              IF(FLD(3,1,NSQ1(J1)).EQ.0) GO TO 40
00216              LTA = FLD(22,14,NSQ1(J1))
00217              T1 = T(I)+460.0
00220              T2 = T(LTA)+460.0
00221              X(LE4) = G(LG)*(T1*T1+T2*T2)*(T1+T2)
00222              GO TO 45
00223           40 X(LE4) = G(LG)
00223       C      CHECK FOR LAST CONDUCTOR INTO THIS NODE
00224           45 IF(NSQ1(J1).GT.0) GO TO 35
00226           50 CONTINUE
00226       C      SWITCH CONDUCTOR VALUES WITH FOURTH EXTRA ARRAY IN X
00230              DO 55 I = 1,NGT
00233              LE4 = IE4+I
00234              GSUM = X(LE4)
00235              X(LE4) = G(I)
00236           55 G(I) = GSUM
00236       C      PERFORM LAXFAC ITERATIONS USING THE LINEARIZED CONDUCTORS
00240              JJ = 0
00241              DO 100 K1 = 1,LAXFAC
00244              KK = K1
00245              JJ = JJ+1
00246              J1 = 0
00247              RLXD = 0.0
00247       C      DO SUCCESSIVE POINT ITERATION ON ALL CALCULATED NODES
00250              DO 85 I = 1,NNC
00253              GSUM = 0.0
00254              QSUM = Q(I)
00255           60 J1 = J1+1
00256              LG = FLD(5,16,NSQ1(J1))
00257              LTA = FLD(22,14,NSQ1(J1))
00260              GSUM = GSUM+G(LG)
00261              QSUM = QSUM+G(LG)*T(LTA)
00261       C      CHECK FOR LAST CONDUCTOR INTO THIS NODE
00262              IF(NSQ1(J1).GT.0) GO TO 60
00264              T2 = QSUM/GSUM
00264       C      OBTAIN THE CALCULATED TEMPERATURE DIFFERENCE
00265              T1 = ABS(T(I)-T2)
00265       C      STORE THE NEW TEMPERATURES AND EXTRAPOLATION FACTORS
00266              GO TO (80,75,65), JJ
00267           65 LE1 = IE1+I
00270              LE2 = IE2+I
00271              R1 = T2-T(I)
00272              X(LE1) = T(I)
00273              IF(R1-X(LE2).NE.0.0) GO TO 70
00275              X(LE2) = 1.0
00276              GO TO 80
00277           70 X(LE2) = R1/(R1-X(LE2))
00300              GO TO 80
00301           75 LE2 = IE2+I
00302              X(LE2) = T2-T(I)
00303           80 T(I) = T2
00303       C      SAVE THE MAXIMUM TEMPERATURE CHANGE AND LOCATION
00304              IF(RLXD.GT.T1) GO TO 85
00306              RLXD = T1
00307              N1 = I
```

```
00310       85 CONTINUE
00310    C     SEE IF THE RELAXATION CRITERIA WAS MET
00312          IF(RLXD.LE.RELAX) GO TO 105
00314          IF(JJ.LE.2) GO TO 95
00316          JJ = 0
00317          DO 90 I = 1,NNC
00322          LE1 = IE1+I
00323          LE2 = IE2+I
00323    C     SEE IF THE EXTRAPOLATION CRITERIA ARE MET
00324          IF(X(LE2).GE.0.0.OR.ABS(X(LE2)*(T(I)-X(LE1))).GE.RLXD) GO TO 90
00326          T(I) = X(LE2)*X(LE1)+(1.0-X(LE2))*T(I)
00327       90 CONTINUE
00331       95 CON(30) = RLXD
00332          KON(37) = N1
00333          IF(KON(7).NE.0) CALL OUTCAL
00335      100 CONTINUE
00337      105 KON(20) = KON(20)+KK
00337    C     STORE CONDUCTANCE VALUES BACK IN THE G ARRAY
00340          DO 110 I = 1,NGT
00343          LE4 = IE4+I
00344      110 G(I) = X(LE4)
00344    C     CHECK IF THE INITIAL NLAX ITERATIONS HAVE BEEN PERFORMED
00346          IF(LPASS.GE.NLAX) GO TO 120
00346    C     THE NLAX INITIAL ITERATIONS HAVE NOT BEEN PERFORMED,
00346    C     APPLY DAMPING FACTOR AND REDUCE RELAX BY ONLY 0.005/NLAX
00350          DO 115 I = 1,NNC
00353          LE3 = IE3+I
00354      115 T(I) = DAMP*T(I)+RLX*X(LE3)
00356          XXX = XXX-DELXXX
00357          GO TO 15
00357    C     AFTER THE NLAX INITIAL ITERATIONS, REDUCE RELAX TO 0.001
00360      120 CONTINUE
00361          N2 = 0
00361    C     CHECK TO SEE IF MAXIMUM NUMBER OF ITERATIONS HAS BEEN EXCEEDED
00362          IF(KON(20).GE.KON(5)) GO TO 130
00362    C     HAS NOT BEEN EXCEEDED, REDUCE RELAX TO 0.001
00364          N2 = 1
00365          XXX = XXXDUM
00366          GSUM = 0.0
00366    C     OBTAIN THE MAXIMUM TEMPERATURE CHANGE
00367          DO 125 I = 1,NNC
00372          LE3 = IE3+I
00373          QOUT = ABS(T(I)-X(LE3))
00374          IF(QOUT.GT.GSUM) GSUM = QOUT
00376      125 CONTINUE
00376    C     IF THE MAXIMUM TEMPERATURE CHANGE OVER THE QUASI-INTERVAL EXCEEDS
00376    C     RELAX, REDUCE CRITERIA AND PERFORM MORE ITERATIONS
00400          IF(GSUM.LE.RELAX) GO TO 130
00400    C     REDUCE LAXFAC TO THE DIFFERENCE BETWEEN NLOOP AND LOOPCT, SO
00400    C     THAT NLOOP REMAINS THE MAXIMUM NUMBER OF ITERATIONS POSSIBLE
00402      128 LAXFAC = KON(5)-KON(20)
00403          GO TO 15
00403    C     CHECK THE ENERGY BALANCE OF THE SYSTEM
00403    C     NLOOP HAS BEEN EXCEEDED
00404      130 N2 = N2+1
00405          QOUT = 0.0
00406          J1 = 0
00406    C     DETERMINE NET HEAT FLOW TO BOUNDARY NODES
00407          DO 150 I = 1,NNC
00412      135 J1 = J1+1
```

```
00413         LTA = FLD(22,14,NSO1(J1))
00413   C     CHECK FOR BOUNDARY NODE
00414         IF(LTA.LE.NNC) GO TO 145
00416         LG = FLD(5,16,NSO1(J1))
00416   C     CHECK FOR RADIATION CONDUCTOR
00417         IF(FLD(3,1,NSO1(J1)).EQ.0) GO TO 140
00421         T1 = T(I)+460.0
00422         T2 = T(LTA)+460.0
00423         QOUT = QOUT+G(LG)*(T1**4-T2**4)
00424         GO TO 145
00425     140 QOUT = QOUT+G(LG)*(T(I)-T(LTA))
00425   C     CHECK FOR LAST CONDUCTOR INTO THIS NODE
00426     145 IF(NSO1(J1).GT.0) GO TO 135
00430     150 CONTINUE
00432         CON(32) = ABS(QIN-QOUT)
00433         GO TO (160,155), N2
00434     155 IF(CON(32).LE.CON(33)) GO TO 160
00434   C     NLOOP HAS NOT BEEN EXCEEDED, MAXIMUM TEMPERATURE CHANGE IS LESS
00434   C     THAN OR EQUAL TO RELAXATION CRITERIA BUT ENGBAL IS GREATER THAN
00434   C     BALENG.  MORE ITERATIONS WILL BE PERFORMED WITH A TIGHTER CRITERIA
00436         XXXDUM = XXXDUM/5.0
00437         XXX = XXXDUM
00440         GO TO 128
00440   C     EITHER NLOOP HAS BEEN EXCEEDED OR ENGBAL IS LESS THAN OR
00440   C     EQUAL TO BALENG.  IN EITHER CASE PRINT LOOPCT AND ENGBAL,
00440   C     INCREMENT TIME AND PROCEED WITH THE PROBLEM.
00441     160 WRITE(6,884) KON(20),CON(32)
00445         KON(28) = KON(28)+2
00446         CON(30) = RLXD
00447         KON(37) = N1
00450         CALL VARBL2
00451         CON(13) = CON(1)
00452         CALL OUTCAL
00452   C     TEST TO SEE IF TIMEND HAS BEEN REACHED
00453         IF(CON(3).GT.CON(1)*1.000001) GO TO 5
00455         NTH = IE1
00456         NDIM = NLA
00457         RETURN
00460     995 WRITE(6,885) NDIM
00463         GO TO 1000
00464     996  WRITE(6,886)
00466         GO TO 1000
00467     997 WRITE(6,887)
00471         GO TO 1000
00472     998 WRITE(6,888)
00474         GO TO 1000
00475     999 WRITE(6,889)
00477    1000 CALL OUTCAL
00500         CALL EXIT
00501     884 FORMAT(10H0LOOPCT = I6,10H ENGBAL = E12.5)
00502     885 FORMAT(I8,20H LOCATIONS AVAILABLE)
00503     886 FORMAT(10H NO LAXFAC)
00504     887 FORMAT(10H NO BALENG)
00505     888 FORMAT(21H CINDSM REQUIRES LPCS)
00506     889 FORMAT(10H NO NLOOP )
00507         END

     END OF UNIVAC 1108 FORTRAN V COMPILATION.     0 *DIAGNOSTIC* MESSAGE(S)
```