JAERI-M
91-051

IMPLEMENTATION OF REACTOR SAFETY ANALYSIS CODE RELAP5/MOD3 AND ITS VECTORIZATION ON SUPERCOMPUTER FACOM VP2600

March 1991

Misako ISHIGURO, Toshiyuki NEMOTO* and Atsushi HIRATSUKA*

日 本 原 子 力 研 究 所
Japan Atomic Energy Research Institute

JAERI-M レポートは，日本原子力研究所が不定期に公刊している研究報告書です。

入手の問合わせは，日本原子力研究所技術情報部情報資料課（〒319-11茨城県那珂郡東海村）あて，お申しこしください。なお，このほかに財団法人原子力弘済会資料センター（〒319-11 茨城県那珂郡東海村日本原子力研究所内）で複写による実費頒布をおこなっております。

JAERI-M reports are issued irregularly.

Inquiries about availability of the reports should be addressed to Information Division Department of Technical Information, Japan Atomic Energy Research Institute, Tokai-mura, Naka-gun, Ibaraki-ken 319-11, Japan.

編集兼発行　日本原子力研究所
印　　　刷　いばらき印刷㈱

JAERI-M 91-051

Implementation of Reactor Safety Analysis Code RELAP5/MOD3 and Its Vectorization on Supercomputer FACOM VP2600

Misako ISHIGURO, Toshiyuki NEMOTO* and Atsushi HIRATSUKA*

Computing and Information Systems Center
Tokai Research Establishment
Japan Atomic Energy Research Institute
Tokai-mura, Naka-gun, Ibaraki-ken

(Received February 25, 1991)

RELAP5/MOD3 is an advanced reactor safety analysis code developed at Idaho National Engineering Laboratory (INEL) under the sponsorship of USNRC. The code simulates thermohydraulic phenomena involved in loss of coolant accidents in pressurized water reactors. The code has been introduced into JAERI as a part of the technical exchange between the JAERI and USNRC under the ROSA-IV Program.

First, the conversion to FACOM (= FUJITSU) M-780 version was carried out based on the IBM version extracted from the original INEL RELAP5/MOD3 source code. Next, the FACOM version has been vectorized for efficient use of new supercomputer FACOM VP2600 at JAERI. The computing speed of vectorized version is about three times faster than the scalar. The present vectorization ratio is 78%.

In this report, both the implementation and vectorization methods on the FACOM computers are described.

Keywords: RELAP5/MOD3, Thermohydraulics, Reactor Safety, Computer Codes, Transient Analysis, Supercomputer, Vectorization, FACOM VP2600

---

* On leave from FUJITSU

軽水炉安全性解析コードRELAP5／MOD3の変換とFACOM VP2600用ベクトル化

日本原子力研究所東海研究所情報システムセンター
石黒美佐子・根本　俊行*・平塚　　篤*

（1991年2月25日受理）

RELAP5/MOD3は，アイダホ国立研究所（INEL）で開発された最新の軽水炉安全性解析コードで，加圧水型原子炉の熱流動現象をシミュレートするために使用されている。コードは，原研と米国原子力規制委員会との間のROSA－IV協定における技術情報交換の一環として原研に導入された。

まず，INELから提供された元版からIBMバージョンを抽出し，それを原研のFACOM M－780用に変換し，その後，FACOM VP2600で効率的に処理するためにベクトル化した。ベクトル化版コードは，スカラー計算に比較して約3倍速くなっている。現在のベクトル化率は78％である。

本報告書では，FACOM計算機への変換方法とベクトル化方法について述べる。

東海研究所：〒319－11　茨城県那珂郡東海村白方字白根2－4
*　外来研究員，富士通

Contents

目　　次

1. Introduction

RELAP5/MOD3 code has been developed by Idaho National Engineering Laboratory (INEL) under the suponsorship of USNRC. The code simulates thermohydraulic phenomena involved in loss of coolant accidents in pressurized water reactors. The code has been introduced into Japan Atomic Energy Research Institute (JAERI) as a part of the technical exchange between the JAERI and USNRC under the ROSA-IV Program.

RELAP5/MOD3 code was originally developed on CRAY X-MP/24 under a UNICOS operating system. But the present INEL source code in update format covers the various computer hardware/software environments: CRAY, IBM, VAX, CDC under both UNIX or mainframe dependent operating systems. Then each code-user can chose a coding type which is suited for his own computer system. The developmental version (RELAP5/MOD2.5 *1) was first delivered at JAERI in early Summer in 1989. After then we implemented FACOM version in order for prelimimary use at the Thermohydraulic Safety Engineering Laboratory of JAERI. The developmental assessment of RELAP5/MOD3 code was reported *2.

The overall goals of RELAP5/MOD3*3 code from the previous RELAP5/MOD2 *4 are explained as follows:

(1) Improved physical modeling,
(2) Faster execution speed,
(3) Easy portability,
(4) Improved documentation.

The FACOM version at JAERI is based on the coding extracted with IBM option from INEL source code in update format. The IBM code has been first converted into FACOM M-780 environment. The M-780 computer is compatible with IBM 370 series.

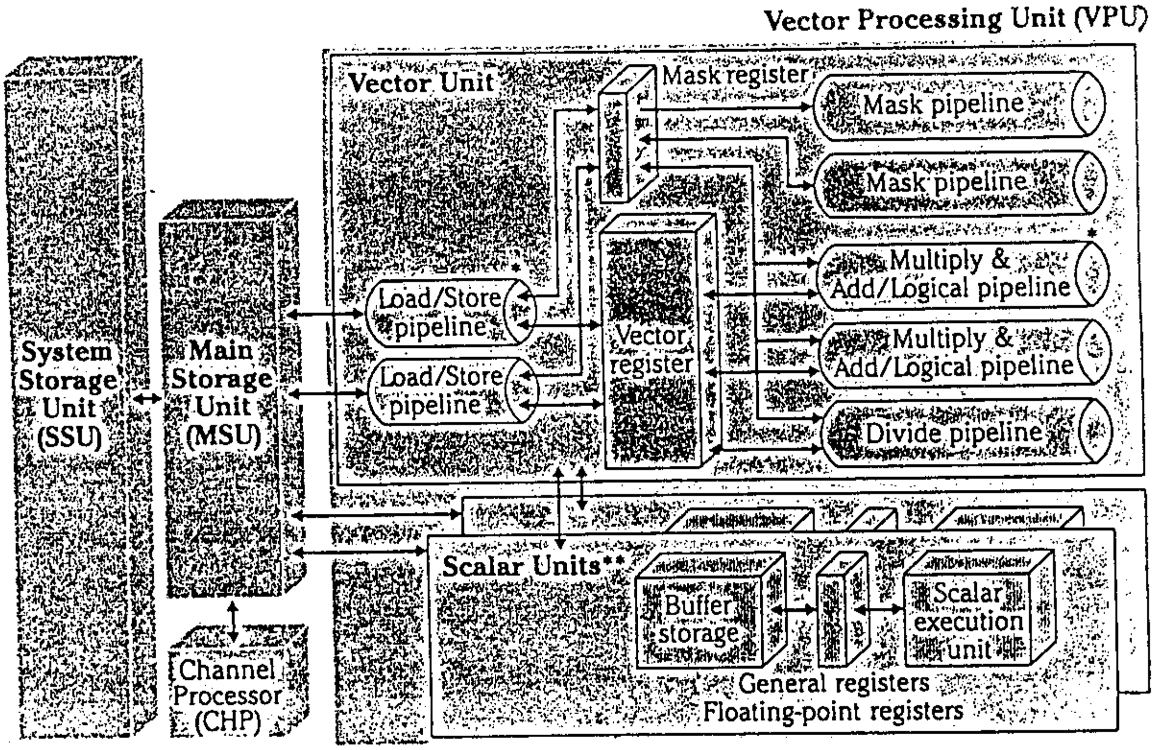
It is of great important to accelerate the computation of RELAP5 code on vector supercomputer. Therefore, the FACOM versions of RELAP5 codes implemented so far have been vectorized for efficient use of supercomputer VPs installed at JAERI. In the past, JAERI succeeded in the vectorization of RELAP5/MOD1 *5 and MOD2 *6 codes. In response to the request from USNRC, the vectorized RELAP5/MOD2 code was supplied to INEL. The computing speed for the MOD2 is four times faster than the scalar calculation on the new supercomputer FACOM VP2600 of JAERI to the sample problem TYPPWR for a small break LOCA analysis. The JAERI vectorized program structure is applied to most part of the present RELAP5/MOD3 code, except that the heat transfer calculations. The vectorization of RELAP5/MOD3 code has been carried out at JAERI to get complete vectorized version. The manpower required at JAERI was less than before. The table look-up of steam table is newly vectorized, but the obtained speedup is 2.8 times that is less than the MOD2 for the same sample problem. This is because lower vectorization ratio. At present, 53 subroutines are modified and the vectorization rario 78% is achieved. Further effort is required.

The vectorized RELAP5/MOD3 of JAERI is also supplied to USNRC.

The FACOM VP2600 hardware block diagram is shown in Fig. 1.1 The clock period is about 3.2 ns. The peak speed of the VP2600 is 5 GFLOPS, but the actual computing speed of RELAP5/MOD3 is about 100 MFLOPS for practical scale applications .

In this report, both the implementation method and vectorization method are described, in Chapters 2 and 3.

FUJITSU VP2000 Series Hardware Block diagram



**One scalar unit with uni-processor models.

Fig. 1.1 FACOM (=FUJITSU) VP-2600 hardware block diagram

2. Implementation of RELAP5/MOD3 Code on FACOM M-780 Computer

2.1 Overview

The original source code transmitted from INEL involves various coding types in order to keep a portability of the code. In fact, the INEL source code was created so as to be used by various computer users such like CRAY, IBM, VAX, CDC under both UNIX or mainframe-dependent operating systems. Users can extract their own code by specifying options appropriate for their computer.

At JAERI, implementation of RELAP5 code on FACOM M-780 has been performed automatically using software tools in order to decrease troublesome tasks and avoid careless mistakes. The M-780 computer is an IBM compatible one but the software environment is somewhat different. The operating system is FACOM F4 MSP. Therefore, some modification was required to obtain FACOM version from IBM version. The implementation procedure applied at JAERI is illustrated in Fig. 2.1.1. The procedure is summarized as follows:

(1) Extraction: IBM coding type is extracted from the original source code transmitted from INEL using a tool SELECTX

(2) Compare: Each statement of the new IBM code is compared with that of old IBM code which was extracted from the previous INEL source code, using the tool FORTCOMP. The statements different between two codes are marked and editted based on the sequence numbers assigned to the old IBM code. After then they are reserved as update cards.

(3) Conversion: Since the update cards are still aimed at 64-bit Fortran, they are converted into the coding for 32-bit Fortran. For this purpose the tool CONV32 which was transmitted from INEL is used.

(4) UPDATE: The first step is to renumber the old FACOM version, where the numbers correspond to those assigned to the old IBM code. The tool used for this purpose is SEQNUM. In the second step, the converted update Fortran statements are embedded into the old FACOM version using the tool UPDATE.

(5) Hand modification, if necessary: Finally new FACOM version is created.

Here SELECTX, FORTCOMP, SEQNUM, and UPDATE are conversion tools from Cray or CDC environment to FACOM. The tools have been developed at JAERI in cooperation with FUJITSU visited engineers.

2.2 Transmittal files

The contents of files involved in the transmittal magnetic tape sent from INEL are given by Table 2.2.1. Environmental library of the RELAP5 code was separate from RELAP5 main stream. The conversion procedure that will be described hereafter is the same between RELAP5/MOD3 main stream and environmental library.

The files used for the implementation at JAERI is shown in Table 2.2.2.

2.3 Extraction

Base version (= IBM base code) is extracted from the original INEL source code by specifying options appropriate for JAERI environment. The options specified for RELAP5/MOD3 main stream are:

IBM, IN32, MASS, COUPLD, CHNG8, CHNG9, CHNG10, BLKDTA, TIMED, PLOTS.

The options specified for environmental library are:

IBM, IN32, SCOPE1, CH8.

In the original INEL source code, a kind of directives beginning with ¥IF••• are inserted to select the Fortran statements corresponding to the specified options. The directives are illustrated in Fig. 2.3.1

At JAERI, the tool SELECTX, similar as INEL transmittal tools SELECTF and SELECTM, is used to perform the extraction according to the directives.

The input/output files for SELECTX are shown in Fig. 2.3.2 and the Job Control Languages (JCLs) on FACOM for extracting RELAP5/MOD3 and environmental library are shown in Figs. 2.3.3 and 2.3.4, respectively. Here the INEL original source file ORGNEW and the select options CONTROL are inputed. The extracted source code file IBMNEW, INCLUDE source file INC (= COMDECK file on CRAY), and check list are outputed. In the JCLs, the following file definition names are used:

INEL source file	INSOC,
Select options (CONTROL)	SYSIN,
New IBM base code (extracted)	OUTSOC, PO file,
Include file	OUTINC, PO file.

2.4 Comparison

Each Fortran statement of the extracted new IBM (base) code is compared with that of old IBM code step by step. As the result, update card set which involves modification cards to the old IBM code is generated. The update cards consist of designations of 'insert' or 'delete' of sequence numbers, as shown in Fig. 2.4.1. Note that the sequence numbers here correspond to the old IBM code.

We used a tool FORTCOMP to find the diffrences of Fortran statements between two source codes and edit them as the update cards. The input/output files for FORTCOMP and JCL of it are shown in Figs. 2.4.2 and 2.4.3, respectively.

File definition names are as follows:

New IBM code	NEWSOC,
Old IBM code	OLDSOC,
Update cards set	UPDATED.

2.5 Conversion

In the conversion step, update cards are converted from CRAY Fortran (64-bit) to the IBM Fortran (similar to FACOM Fortran), since the update cards extracted so far are aimed at CRAY except for bit-processing. Therefore, a conversion tool CONV32 was transmitted from INEL. We used modified CONV32 which was turned up for JAERI environment. The function of CONV32 is as follows:

(1) Double precision real number constants: 1.0 ==> 1.0D0

(2) Double precision REAL declarations: REAL FA(1) ==> REAL*8 FA(1)

(3) 32-bit integer array treatment: INTEGER IA(1) ==> INTEGER IA(2,1)
  (array names are specified as input data)

(4) Treatment of 32-bit integers in EQUIVALENCE statement:
  (array names are specified as input data) EQUIVALENCE (FA(1),IA(1)) ==>
  EQUIVALENCE (FA(1),IA(1,1))

(5) Use of integer arrays IA(N)=M ==> IA(2,N)=M
  (array names are specified as input data) M=IA(N) ==> M=IA(2,M)
  CALL ABC (IA(N),..)
  ==> CALL ABC (IA(1,N),...)

(6) Packing of blank characters when a statement overs 72 columns.

(7) Global declaration of specified array names: 'Dimension-up' such as the conversion (3), (4), and (5) is carried out for all the statements which involves the specified array names, over all subroutines.

The input/output files of CONV32 is shown in Fig. 2.5.1. Here the MLIST5M and MLISTE6 are files transmitted from INEL. The MLIST5M provides us with the array names for global declaration to RELAP5/MOD3 main stream and the file MLISTE6 provides with the array names to environmenttal library. After the execution of CONV32, a converted update file FCMUP is generated.

The CONV32 is used in conversational mode. The TSS command procedure for the CONV32 and example of how-to-use of commands are given by Figs. 2.5.2 and 2.5.3, respectively.

It should be noted that a kind of hand rewritting is necessary before the execution of CONV32. That is, when an argument at subroutine call is array variable which is specified for dimension-up and the corresponding virtual argument is non-array variable, the following rewritting is required, where in this example, the value IA(N) is referred and value IA(N+1) is updated in the subroutine ABC:

```
            .                                  .
            .                             IARG1 = IA (N)
            .                             IARG2 = IA (N+1)
 CALL ABC (IA(N), IA(N+1))      ===>      CALL ABC (IARG1,IARG2)
            .                             IA(N+1) = IARG2
            .                                  .
            .                                  .
```

2.6 Update

In update phase, the old FACOM version is updated using the converted update cards and new FACOM version is created. Update is carried out with two steps. In the first step, renumbering of old FACOM version is performed. For each Fortran step, a number corresponding to the old IBM code is assigned. The tool used for this purpose is SEQNUM (see Fig. 2.6.1).

In the second step, the converted update statements are embedded into the old FACOM version using the tool UPDATE (Fig. 2.6.2).

The input/output files of the tool SEQNUM and its JCL are shown in Figs. 2.6.3 and 2.6.4, respectively. The input/output files of the tool UPDATE and its JCL are shown in Figs. 2.6.5 and 2.6.6, respectively. The file definition names are as follows.

Old IBM base code	CHKSOK,
Old FACOM version	INSOC,
Old FACOM version with sequence number	OUTSOC for tool SEQNUM, OLDSOC for tool UPDATE,
Update cards set	UPDATECD,
New FACOM version	NEWSOC.

2.7 Hand modification

Special care has to be given to avoid the inconsistent coding: inconsistent use of argument type and dimension between real and virtual variables at subroutine calls, incorrect Boolian variable definition, and incorrect memory sharing between local integer arrays and real arrays by EQUIVALENCE statements. In the third case dimension-up may be required. For these situations, hand rewrittings were carried out. The inconsistencies are usually found as conversion errors after automatic code conversion. These error corrections should be added to the update cards for the future code conversion.

Finally new FACOM version is created (see Fig. 2.6.2).

Table 2.2.1 RELAP5/MOD3 transmittal tape contents version 5M5

RELAP5/MOD3
TRANSMITTAL TAPE CONTENTS
VERSION 5M5

	CDC Name	UNIX Name	Contents
File 1:	CONTENT	contents	Contents of this transmittal.
File 2:	CONTALL	contentall	Master list of transmittal tape files.
File 3:	CODETAB	codetable	Code versions and corresponding auxiliary file versions.
File 4:	RTESTS	rtests	Fortran coding for dummy installation without compiling.
File 5:	RE5M5S	re5m5.s	RELAP5/MOD3 Version 5M5 source in Update source format.
File 6:	TAPCHK1	tapchk1	Dummy file to check tape position.
File 7:	REEN611	reen611.s	Environmental library version 611 source in Update source format.
File 8:	UPEML	upeml	Fortran coded version of an Update very similar to the Update Program on CDC and Cray computers.
File 9:	USPLITS	usplit.s	Source for Utility program for systems without Update.
File 10:	CNV32S	cnv32.s	Source for convert program for 32 bit machines.
File 11:	SEGDIRC	segdirc	Segloader input used by the install_ scripts.
File 12:	MLISTE6	mliste	File used with application of cnv32 to environmental source on Vax computer.
File 13:	MLIST5M	mlistm	File used with application of cnv32 to material property source on Vax computer.
File 14:	EXBLANK	exendblank	Program to remove trailing blanks from scripts (must be run on the install script). Fixed format tapes have added blanks to fill out to column 80. Since UNIX continuation lines are a backslash immediately followed by a carriage return, the extra blanks would destroy all of the continuation characters at the end of the lines.
File 15:	INSTALG	instal'cray	Master Cray installation script.
File 16:	AWKSPLI	awksplitw	Program to split routines from an envi source.

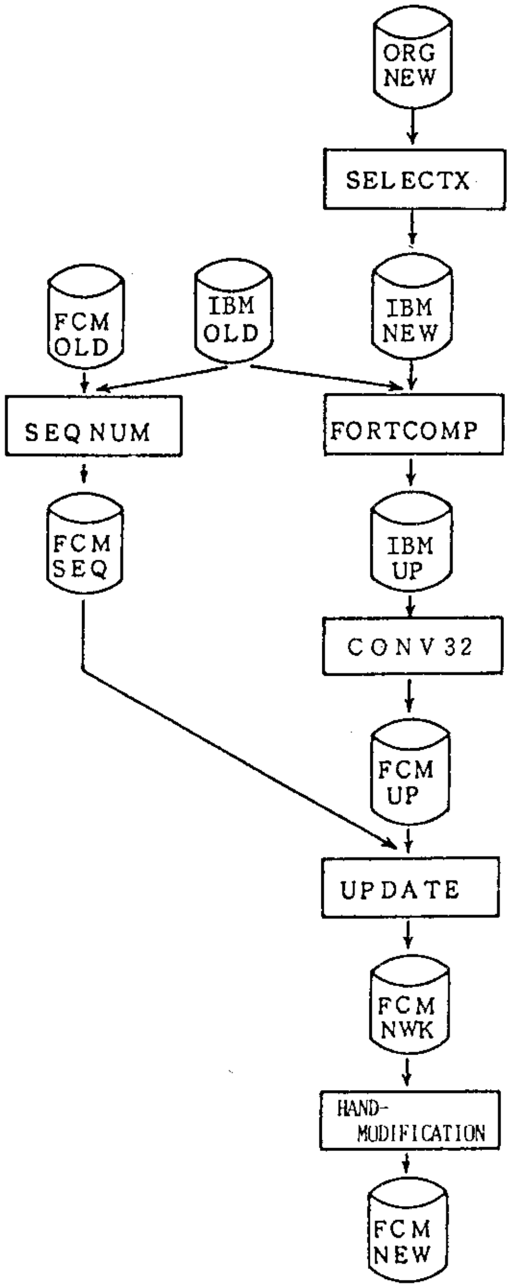
Table 2.2.1 (Continued)

RELAP5/MOD3
TRANSMITTAL TAPE CONTENTS
VERSION 5M5

	CDC Name	UNIX Name	Contents
File 17:	MASTERM	mastermake	Makefile for UNIX installations.
File 18:	MAKEUTI	makeutil	Create utility scripts for the master make script.
File 19:	DUK5K5	dukler5k5	Dukler problem sample input.
File 20:	EDH5M5	edhtrk5m5	Edwards pipe problem sample input.
File 21:	L315K3	l315k3	Loft L3-1 problem sample input.
File 22:	PR25K3	prob25k3	Workshop problem 2 sample input.
File 23:	TYA5K3	typpwr5k3	Typical PWR problem sample input.
File 24:	TYR5K3	typrst5k3	Typical PWR restart problem sample input.
File 25:	INSTRUC	instructions	Transmittal installation instructions.
File 26:	SELECTM	select.m	Compileable Select program for Masscomp.
File 27:	SELECTF	select.f	Compileable Select program (except for Masscomp).
File 28:	INSTALU	installunix	Master UNIX installation script.
File 29:	TITL5M5	title5m5	Short synopsis of all updates to create version 5m5.
File 30:	ODU5K3	odukler5k3	Output (version 5k3) from Dukler sample problem.
File 31:	OED5K3	oedhtrk5k3	Output (version 5k3) from Edwards pipe sample problem.
File 32:	OED5M5	oedhtrk5m5	Output (version 5m5) from Edwards pipe sample problem.
File 33:	OL35K3	ol315k3	Output (version 5k3) from Loft L3-1 sample problem.
File 34:	OP25K3	oprob25k3	Output (version 5k3) from Workshop problem 2 sample problem.
File 35:	OTA5K3	otyppwr5k3	Output (version 5k3) from Typical PWR sample problem.
File 36:	OTR5K3	otypr5k3	Output (version 5k3) from Typical PWR restart sample problem.
File 37:	TAPCHK2	tapchk2	Dummy file to check tape position.

Table 2.2.2 Files used at JAERI

File. 5	RE5M5S	RELAP5/MOD3 source
File. 7	REEN611	Environmental Libraly
File. 10	CNV32S	Source for convert program for 32 bit machines
File. 12	MLISTE6	Input data for CONV32 (for ENV lib)
File. 13	MLIST5M	Input data for CONV32 (for RELAP5)
File. 19	DUK5K5	DUKIER sample input data
File. 20	EDH5M5	EDHTRK sample input data
File. 21	L315K3	LOFT L3-1 sample input data
File. 22	PR25K3	PROB2 sample input data
File. 23	TYA5K3	TYPPWR sample input data
File. 24	TYR5K3	TYPPWR(Restart) sample input data
File. 30	ODU5K3	DUKIER output results
File. 31	OED5K3	EDHTRK output results
File. 32	OED5M5	EDHTRK output results
File. 33	OL35K3	LOFT L3-1 output results
File. 34	OP25K3	PROB2 output results
File. 35	OTA5K3	TYPPWR output results
File. 36	OTR5K3	TYPPWR(Restart) output results



ORGNEW : New original program.
IBMNEW : New source program of IBM version.
IBMOLD : Old source program of IBM version.
FCMOLD : Old source program of FACOM version.
CDCUP : Update-card of IBM version.
FCMSEQ : Old source program of FACOM version with the sequence number of Old source program of IBM version.
FCMUP : Update-card of FACOM version.
FCMNWK : Source program of FACOM version updated by UPDATE.
FCMNEW : New source program of FACOM version.

Fig. 2.1.1 General flow for conversion of RELAP5 by using tools

```
C  TEST FOR STEADY STATE.
C  IF STEADY STATE IS ACHIEVED, DONE = -5 AND IECF = MASK(4).
      IF(IROUTE .EQ. 1) CALL SSTCHK(IECF, SSDTIM)
       IPRNT = PRINT
   39  CONTINUE
¥IF DEF,TIMED,2
       CALL TIMEL (SAFE1)
       TIMEI(1) = TIMEI(1) + SAFE1
       IF (STSCPU(FILNDX(20)) .GE. STSOLD) THEN
         STSOLD = STSCPU(FILNDX(20)) + 10.0
         WRITE (MESSG,2099) STSCPU(FILNDX(20)),TIMEHY,DT,NCOUNT
 2099  FORMAT (' CPU=',F6.0,' SEC, PROB TIME=',F11.4,' SEC, DT=',F10.6,
     * ' SEC, ADV CNT=',I8)
¥IF DEF,NPA,1
         IF (IAND(PRINT,32) .EQ. 0) THEN
¥IF DEF,CTSS,1
           CALL MSGTTY (MESSG,132)
¥IF -DEF,CTSS,1
           WRITE (TTY,'(A)') MESSG(1:80)
¥IF DEF,NPA
         ELSE
¥IF -DEF,IN32,1
           CALL FMESSG (M,1,TIMEHY,MESSG)
¥IF DEF,IN32,2
           TIME4 = TIMEHY
           CALL FMESSG (M,1,TIME4,MESSG)
           IF (M .NE. 0) WRITE (TTY,2008)
 2008  FORMAT (' ERROR NUMBER',I5,' RETURNED FROM NPA MESSAGE ROUTINE.')
         ENDIF
¥ENDIF
¥IF -DEF,CTSS
¥IF DEF,SELAP
         IF (IAND(PRINT,32).EQ.0 .AND. NCVOL.GT.0) THEN
```

Fig. 2.3.1 Example of directives

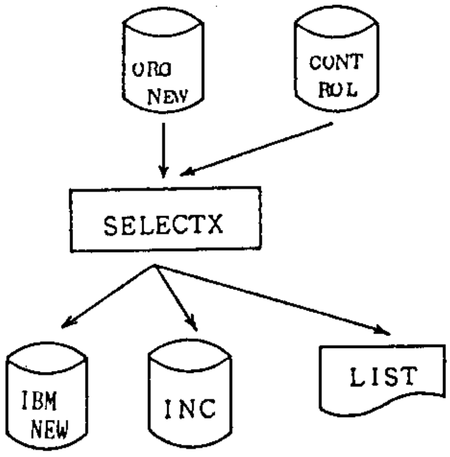


Fig. 2.3.2 Input/output files for SELECTX

```
//JCLG JOB
// EXEC JCLG
//SYSIN DD DATA,DLM='++'
// JUSER 12345678,W.ASANO,4126.99
  T.3 C.2 W.3 I.4 SRP
 OPTP PASSWORD=FUJITV
//*    T.3 W.3 I.5 C.2 SRP
//*
//*  **********************************************************
//*  ********    RELAP5/MOD2.5 SELECT-X FOR RELAP5  *********
//*  **********************************************************
//*
//           EXEC PGM=SELECTX,PARM='ELM(*),MSGLEVEL(9)'
//STEPLIB     DD DSN=J0001.R5TOOL.LOAD,DISP=SHR
//INSOC       DD DSN=J9127.LT2.RELAP5,DISP=SHR,LABEL=(,,,IN)
//OUTSOC DD DSN=J9127.@@.RELAP5.SELECT,
//          DISP=(NEW,CATLG,DELETE),
//          DCB=(LRECL=80,BLKSIZE=22000,RECFM=FB,DSORG=PO),
//          SPACE=(TRK,(100,10,60),RLSE),UNIT=TSSWK
//OUTINC DD DSN=J9127.@@.RELAP5.INCLUDE,
//          DISP=(NEW,CATLG,DELETE),
//          DCB=(LRECL=80,BLKSIZE=22000,RECFM=FB,DSORG=PO),
//          SPACE=(TRK,(10,10,60),RLSE),UNIT=TSSWK
//SYSPRINT  DD SYSOUT=*,DCB=(RECFM=FBA,LRECL=137,BLKSIZE=6850)
//SYSIN       DD *
*DEFINE IBM,IN32,MASS,COUPLD,CHNG8,CHNG9,CHNG10,BLKDTA,TIMED,PLOTS
¥DEFINE IBM,IN32,MASS,COUPLD,CHNG8,CHNG9,CHNG10,BLKDTA,TIMED,PLOTS
/*
++
//
```

Fig. 2.3.3 Example of JCL for SELECTX (RELAP5)

```
//JCLG JOB
// EXEC JCLG
//SYSIN DD DATA,DLM='++'
// JUSER 12345678,W.ASANO,4126.99
  T.3 C.2 W.3 I.4 SRP
 OPTP PASSWORD=FUJITV
//*    T.3 W.3 I.5 C.2 SRP
//*
//*  **********************************************************
//*  ********    RELAP5/MOD2.5 SELECT-X FOR ENV.LIB  *********
//*  **********************************************************
//*
//           EXEC PGM=SELECTX,PARM='ELM(*),MSGLEVEL(9)'
//STEPLIB     DD DSN=J0001.R5TOOL.LOAD,DISP=SHR
//INSOC       DD DSN=J9127.LT2.ENVRS,DISP=SHR,LABEL=(,,,IN)
//OUTSOC DD DSN=J9127.@@.ENVRS.SELECT,
//          DISP=(NEW,CATLG,DELETE),
//          DCB=(LRECL=80,BLKSIZE=22000,RECFM=FB,DSORG=PO),
//          SPACE=(TRK,(10,10,30),RLSE),UNIT=TSSWK
//OUTINC DD DSN=J9127.@@.ENVRS.INCLUDE,
//          DISP=(NEW,CATLG,DELETE),
//          DCB=(LRECL=80,BLKSIZE=22000,RECFM=FB,DSORG=PO),
//          SPACE=(TRK,(3,3,10),RLSE),UNIT=TSSWK
//SYSPRINT  DD SYSOUT=*,DCB=(RECFM=FBA,LRECL=137,BLKSIZE=6850)
//SYSIN       DD *
*DEFINE IBM,IN32,SCOPE1,CH8
¥DEFINE IBM,IN32,SCOPE1,CH8
/*
++
//
```

Fig. 2.3.4 Example of JCL for SELECTX (Env.lib)

```
*DELETE     8 -     9
*INSERT     9
C
*INSERT    35
C
C  LOCAL VARIABLES.
       INTEGER I,ICARD,ICORAN,IECF,IPLT,IPRNT,IV,IVSKP2,J,K,M,NWQA
       SAVE NWQA
       REAL DTADJ,DTREM,DTX,FACTOR,SSDTIM,STSOLD
       INTEGER IWRD8
*DELETE    38 -    38
*INSERT    38
       EXTERNAL INTERI,MAJOUT,MIREC,MOVER,PLTWRT,RSTREC,SSTCHK,TIMER
       EXTERNAL PLTREC
       EXTERNAL TIMEL
C
C  DATA STATEMENTS.
*INSERT    41
       DATA IWRD8/8/
*DELETE    43 -    48
*INSERT    48
C
      ICORAN = 1
*DELETE    59 -    60
*DELETE   160 -   160
*INSERT   160
C  IF ERRMAX IS SMALL, DOUBLE HALVED TIME-STEP.
*DELETE   166 -   169
*INSERT   169
  150  IF (IECF .NE. 0) GO TO 122
*DELETE   198 -   200
*INSERT   200
C  GET NEW TIME-STEP AFTER SUCCESSFUL ADVANCEMENT TO ORIGINAL NEWTIME.
C  DO NECESSARY EDITS AND PLOT RECORDS.
*DELETE   244 -   245
*INSERT   245
C  TEST FOR STEADY STATE.
C  IF STEADY STATE IS ACHIEVED, DONE = -5 AND IECF = MASK(4).
*DELETE   256 -   256
*INSERT   256
           WRITE (TTY,'(A)') MESSG(1:80)
*DELETE   265 -   277
*INSERT   277
       IF (IAND(IPLT,8) .NE. 0) THEN
         IF (IAND(IVSKP2,8).NE.0 .OR. IAND(IPRNT,32).NE.0) THEN
           CALL PLTWRT
           IF (IAND(IVSKP2,8) .NE. 0) THEN
             WRITE (RSTPLT) NWQA,IWRD8
             WRITE (RSTPLT) (FA(K),K=IXIPX,NWQ)
           ENDIF
         ENDIF
```

Fig. 2.4.1 Example of update cards

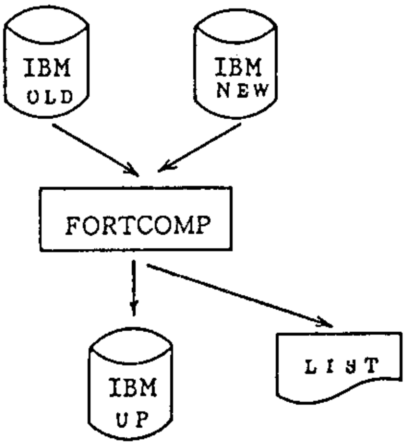


Fig. 2.4.2 Input/output files for FORTCOMP

```
//JCLG JOB
// EXEC JCLG
//SYSIN DD DATA,DLM='++'
// JUSER 12345678,W.ASANO,4126.99
  T.3 C.2 W.O I.4 SRP
 OPTP PASSWORD=FUJITV
//*   T.2 W.4 I.5 C.2 SRP
//*   ***************************************************************
//*   ********    RELAP5/MOD2.5   FORTRAN COMPARE PROGRAM   ********
//*   ***************************************************************
//*
//COMPARE   EXEC PGM=FORTCOMP
//STEPLIB   DD   DSN=J0001.R5TOOL.LOAD,DISP=SHR
//OLDSOC    DD   DSN=J9127.RE.RELAPS.IBMMOD2.SELECT,
//         DISP=SHR,LABEL=(,,,IN)
//NEWSOC    DD   DSN=J9127.@@.RELAPS.SELECT,DISP=SHR,LABEL=(,,,IN)
//UPDATECD  DD   DSN=J9127.@UPDTCD.DATA,DISP=(NEW,CATLG),
//         UNIT=TSSWK,SPACE=(TRK,(100,10,50))
//SYSPRINT  DD   DSN=J9127.@@.RELAPS.OUTLIST,DISP=(NEW,CATLG,DELETE),
//          DCB=(LRECL=137,BLKSIZE=27400,RECFM=FBA,DSORG=PS),
//          SPACE=(TRK,(50,20),RLSE),UNIT=TSSWK
++
//
```

Fig. 2.4.3 Example of JCL for FORTCOMP

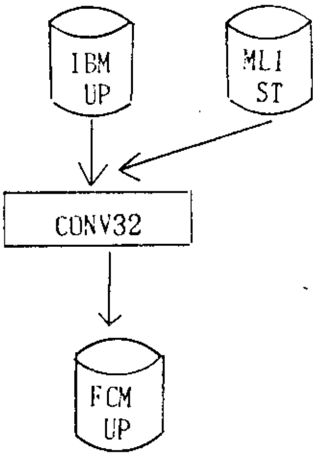


Fig. 2.5.1 Input/output files for CONV32

```
EX 'J9127.RELAP5.CLIST(CONV32)'
     'IN(@UPDTCD.DATA)' 'OUT(@UPDTCD.OUT.DATA)'

 INPUT UPDATE-CARD  :  @UPDTCD.DATA
 OUTPUT UPDATE-CARD :  @UPDTCD.OUT.DATA
```

Fig. 2.5.3 Example of TSS command for CONV32

```
PROC 0 IN('J9127.@@.RELAP5.SELECT') OUT('J9127.KKK.DATA')
CONTROL LIST MSG
FREEALL
DEL (WWW W.DATA '&OUT')
ALLOC DA(WWW) NEW T CAT SP(1 1) F(W) UNIT(TSSWK) REU
OPENFILE W OUTPUT
SET W = &OUT
WRITE W
PUTFILE W
CLOSFILE W
LIB 'J9909.SEAK.LOAD'
ALLOC DA(W.DATA) NEW T CAT SP(1 1) UNIT(TSSWK)
ALLOC DA(W.DATA) F(SYSPRINT) SHR REU
DI '&IN'
ATTR ABC LRECL(80) BLKSIZE(22000) RECFM(F B) DSORG(PO)
ALLOC DA('&OUT') NEW T CAT SP(10 10) DIR(50) RELEASE +
US(ABC) UNIT(TSSWK)
FREE F(W)
ALLOC F(FT02F001) DA(WWW) REU
ALLOC F(FT07F001) DUMMY
ALLOC F(FT08F001) DA('J9127.R5M3.MLIST5M') SHR
ALLOC F(FT66F001) DUMMY
ALLOC F(FT77F001) DUMMY
FORT77 RE.CONV32.FORT77 ELM(*) GO
EXIT
```

Fig. 2.5.2 Command procedure for CONV32

```
    MEMBER NAME  DTSTEP
      SUBROUTINE DTSTEP                                                 00000001
*IN32 JECF                                                              00000002
*IN32END                                                                00000003
C                                                                       00000004
C  CONTROLS TIME STEP SELECTION AND FREQUENCY OF OUTPUT AND PLOTTING    00000006
C  EDITS DURING TRANSIENT ADVANCEMENT.                                  00000007
C                                                                       00000008
       IMPLICIT REAL*8(A-H,O-Z)                                         00000009
*INCLUDE CMPDAT                                                         00000010
*INCLUDE COMCTL                                                         00000011
*INCLUDE CONTRL                                                         00000012
*INCLUDE FAST                                                           00000013
*INCLUDE INTRAC                                                         00000014
*INCLUDE JUNDAT                                                         00000015
*INCLUDE LCNTRL                                                         00000016
*INCLUDE MACHDS                                                         00000017
*INCLUDE NPACOM                                                         00000018
C25CLUDE SCDDAT                                                         00000019
*INCLUDE STATC                                                          00000020
*INCLUDE TIMEC                                                          00000021
*INCLUDE TRNHLP                                                         00000022
*INCLUDE TRPBLK                                                         00000023
*INCLUDE TSCTLC                                                         00000024
*INCLUDE TSTPCT                                                         00000025
*INCLUDE UFILES                                                         00000026
*INCLUDE VOLDAT                                                         00000027
*INCLUDE STATEC                                                         00000028
C25    INTEGER JECF(5)                                                  00000029
       INTEGER JECF(2,5)                                                00000029
C25    REAL DTNM(5)                                                     00000030
       REAL*8 DTNM(5)                                                   00000030
C25    EQUIVALENCE (PROP(1),JECF(1)),(S(1),DTNM(1))                     00000031
       EQUIVALENCE (PROP(1),JECF(1,1)),(S(1),DTNM(1))                   00000032
C25CLUDE SCDOUT                                                         00000033
C25CLUDE SLUMPV                                                         00000034
C25CLUDE NDXARA                                                         00000035
C25CLUDE TBLSP                                                          00000035
       LOGICAL LAST                                                     00000036
       CHARACTER MESSG*132                                              00000037
C  DATA STATEMENTS                                                      00000038
C25    DATA DTREM/100.0/                                                00000039
       DATA DTREM/100.0D0/                                              00000039
C25    DATA STSOLD/0.0/                                                 00000040
       DATA STSOLD/0.0D0/                                               00000040
C      DATA ICORAN/2/                                                   00000041
*INCLUDE MACHDF                                                         00000042
C                                                                       00000043
       IF (CHNGNO(8)) THEN                                              00000044
         ICORAN = 1                                                     00000045
       ELSE                                                             00000046
         ICORAN = 2                                                     00000047
       ENDIF                                                            00000048
       AFLAG = .FALSE.                                                  00000049
       LAST = .FALSE.                                                   00000050
       IECF = 0                                                         00000051
C25    ICARD = CURCTL(FILNDX(2))                                        00000052
       ICARD = CURCTL(2,FILNDX(2))                                      00000052
       I = FILNDX(2) + ICARD                                            00000053
C                                                                       00000054
       IF (SKIPT) GO TO 10                                              00000055
       NWQA = FILSIZ(17)/2                                              00000056
       NWQ = NWQA + IXIPX                                               00000057
       NWQA = NWQA + 1                                                  00000058
       NPANWX = FILSIZ(17) - 1                                          00000059
       NPANW = NPANWX                                                   00000060
C25    PRINT = IOR(IAND(PRINT,NOT(192)),IAND(ISHFT(TSPPAC(I),4),192))   00000061
       PRINT = IOR(IAND(PRINT,NOT(192)),IAND(ISHFT(TSPPAC(2,I),4),192)) 00000061
       IF (NCOUNT .NE. 0) SKIPT = .TRUE.                                00000062
       IECF = 15                                                        00000063
       DTHY = DTMAX(I)                                                  00000064
       DTHT = DTMAX(I)                                                  00000065
       IF (NREPET .NE. 0) GO TO 4                                       00000066
       NREPET   =  1                                                    00000067
       NSTSP = NSTSP - 1                                                00000068
C25    CURCMI(FILNDX(2)) = CURCMI(FILNDX(2)) + 1                        00000069
       CURCMI(2,FILNDX(2)) = CURCMI(2,FILNDX(2)) + 1                    00000069
C25    CURCMJ(FILNDX(2)) = CURCMJ(FILNDX(2)) + 1                        00000070
       CURCMJ(2,FILNDX(2)) = CURCMJ(2,FILNDX(2)) + 1                    00000070
C25    CURCRS(FILNDX(2)) = CURCRS(FILNDX(2)) + 1                        00000071
       CURCRS(2,FILNDX(2)) = CURCRS(2,FILNDX(2)) + 1                    00000071
```

Fig. 2.6.1 Example of old source code with sequence number

```
C25      FACTOR = 5.0                                                  00000341
         FACTOR = 5.0DO                                                00000341
C25      IF (IROUTE .EQ. 1) FACTOR = 10.0                              00000342
         IF (IROUTE .EQ. 1) FACTOR = 10.0DO                            00000342
         IF (IAND(PRINT,128) .NE. 0) DTX = FACTOR*DTX                  00000343
         IF (DT .LE. DTX) GO TO 450                                    00000344
C2521    DT = 0.5*DT                                                   00000345
   21    DT = 0.5DO*DT                                                 00000345
         NREPET = NREPET + NREPET                                      00000346
         IF (DT .GT. DTX) GO TO 21                                     00000347
C25      IECF=JECF(ICORAN)                                             00000348
         IECF=JECF(2,ICORAN)                                           00000348
C25      STRCL2(IECF) = STRCL2(IECF) + 1                               00000349
         STRCL2(2,IECF) = STRCL2(2,IECF) + 1                           00000349
  450  NCOUNT  =  NCOUNT + 1                                           00000350
       TIMEHY    =   TIMEHY + DT                                       00000351
C25      IF (IAND(TSPPAC(I),2) .EQ. 0) RETURN                          00000352
         IF (IAND(TSPPAC(2,I),2) .EQ. 0) RETURN                        00000352
```

Old FACOM version

```
*DELETE    344 -    344
*INSERT    344
CM3    IF( DT .LE. 2.0*DTX ) GO TO 450
       IF( DT .LE. 2.0DO*DTX ) GO TO 450
*DELETE    347 -    347
*INSERT    347
CM3    IF( DT.GT.2.0*DTX ) GO TO 21
       IF( DT.GT.2.0DO*DTX ) GO TO 21
*INSERT    351
C   STORE NEW-TIME VALUES FOR DT ADVANCEMENT.
          IF (SUCCES .EQ. 0) CALL MOVER
```

(converted) UPDATE-CARD

UPDATE

```
   C25      FACTOR = 5.0                                               00000341
            FACTOR = 5.0DO                                             00000341
   C25      IF (IROUTE .EQ. 1) FACTOR = 10.0                           00000342
            IF (IROUTE .EQ. 1) FACTOR = 10.0DO                         00000342
            IF (IAND(PRINT,128) .NE. 0) DTX = FACTOR*DTX               00000343
→  CM3     IF( DT .LE. 2.0*DTX ) GO TO 450                             *
           IF( DT .LE. 2.0DO*DTX ) GO TO 450                           *ONV#REP
   C2521    DT = 0.5*DT                                                00000345
      21    DT = 0.5DO*DT                                              00000345
            NREPET = NREPET + NREPET                                   00000346
→  CM3     IF( DT.GT.2.0*DTX ) GO TO 21                                *
           IF( DT.GT.2.0DO*DTX ) GO TO 21                              *ONV#REP
   C25      IECF=JECF(ICORAN)                                          00000348
            IECF=JECF(2,ICORAN)                                        00000348
   C25      STRCL2(IECF) = STRCL2(IECF) + 1                            00000349
            STRCL2(2,IECF) = STRCL2(2,IECF) + 1                        00000349
      450  NCOUNT  =  NCOUNT + 1                                       00000350
→          TIMEHY    =   TIMEHY + DT                                   00000351
   C   STORE NEW-TIME VALUES FOR DT ADVANCEMENT.                       *
            IF (SUCCES .EQ. 0) CALL MOVER                              *
   C25      IF (IAND(TSPPAC(I),2) .EQ. 0) RETURN                       00000352
            IF (IAND(TSPPAC(2,I),2) .EQ. 0) RETURN                     00000352
```

New FACOM version

Fig. 2.6.2 Example of update processes

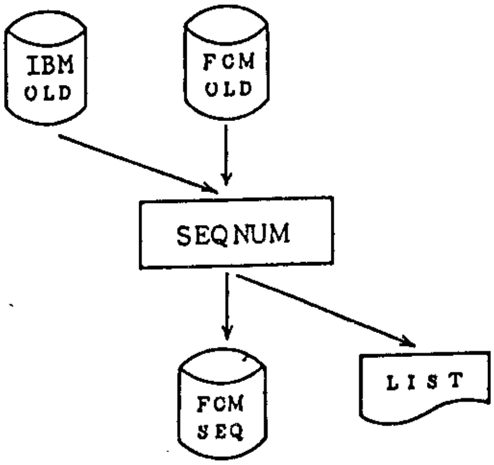


Fig. 2.6.3 Input/output files for SEQNUM

```
//JCLG JOB
// EXEC JCLG
//SYSIN DD DATA,DLM='++'
// JUSER 12345678,W.ASANO,4126.99
  T.2 C.2 W.1 I.4 SRP
 OPTP PASSWORD=FUJITV
//*   T.2 W.1 I.5 C.2 SRP
//*
//*         ********************************************************
//*         *********    RELAP5/MOD2.5 SEQNUMBER PROGRAM    *********
//*         ********************************************************
//*
//         EXEC PGM=SEQNUM
//STEPLIB  DD DSN=J0001.R5TOOL.LOAD,DISP=SHR
//CHKSOC   DD DSN=J9127.RE.RELAPS.IBMMOD2.SELECT,DISP=SHR,LABEL=(,,,IN)
//INSOC    DD DSN=J9127.RE.RELAPS.MOD25.FORT77,DISP=SHR,LABEL=(,,,IN)
//OUTSOC   DD DSN=J9127.QQ.RELAPSW.FORT77,
//        DISP=(NEW,CATLG,DELETE),
//        DCB=(LRECL=80,BLKSIZE=22000,RECFM=FB,DSORG=PO),
//        SPACE=(TRK,(200,50,60),RLSE),UNIT=TSSWK
//SYSPRINT  DD SYSOUT=*,DCB=(RECFM=FBA,LRECL=137,BLKSIZE=6850)
++
//
```

Fig. 2.6.4 Example of JCL SEQNUM

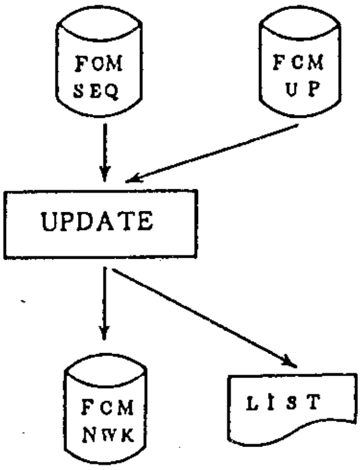


Fig. 2.6.5 Input/output files for tool UPDATE

```
//JCLG JOB
// EXEC JCLG
//SYSIN DD DATA,DLM='++'
// JUSER 12345678,W.ASANO,4126.99
  T.2 C.2 W.0 I.4 SRP
 OPTP PASSWORD=FUJITV
//*   T.2 W.4 I.5 C.2 SRP
//*
//*       ********************************************************
//*       ********    RELAP5/MOD2.5    UPDATE PROGRAM     ********
//*       ********************************************************
//*
//UPDATE   EXEC PGM=UPDATE
//STEPLIB  DD   DSN=J0001.R5TOOL.LOAD,DISP=SHR
//UPDATECD DD   DSN=J9127.@UPDTCD.DATA,DISP=SHR,LABEL=(,,,IN)
//OLDSOC   DD  DSN=J9127.@@.RELAPSW.FORT77,DISP=SHR,LABEL=(,,,IN)
//NEWSOC   DD   DSN=J9127.@@.RELAPSN.FORT77,DISP=(NEW,CATLG,DELETE),
//      DCB=(LRECL=80,BLKSIZE=22000,RECFM=FB,DSORG=PO),
//      SPACE=(TRK,(200,30,60),RLSE),UNIT=TSSWK
//SYSPRINT DD   DSN=J9127.@@.RELAPS.UPD.OUTLIST,
//      DISP=(NEW,CATLG,DELETE),
//      SPACE=(TRK,(100,50),RLSE),UNIT=TSSWK,
//      DCB=(LRECL=137,BLKSIZE=27400,RECFM=FBA,DSORG=PS)
++
//
```

Fig. 2.6.6 Example of JCL for tool UPDATE

3. Vectorization Method for RELAP5/MOD3 Code

3.1 Overview

The high speed simulation of nuclear reactor plant transient is very important in quickly predicting the anticipated danger at the nuclear accidents. A few of the present advanced thermohydraulic nuclear reactor codes for the transient analysis, however, have a computing speed necessary for real time simulation, even on the world highest speed computer. A lot of efforts have been made for reducing the computing time of RELAP5 *[8].

The FACOM version, converted from the IBM-coding type extracted from INEL transmittal source, has been vectorized. In the code, since a staggered mesh method is applied for discretization, then the code can be potentially vectorizable on volumes and/or junctions for hydrodynamic calculations. And the heat transfer calculations can be vectorized on the heat mesh points or intervals, and/or heat structures.

Although very much effort has been given by INEL group to improve the code for faster calculation, but the present RELAP5/MOD3 code still has the following vectorization difficulties:

(1) Program structure unsuitable for vectorization,
(2) Data structure unsuitable for vectorization,
(3) Many IF tests depending on the state of fluid (==> short vector),
(4) Huge program size (==> a lot of manpower for vectorization).

The problems (1)〜(3) are affected by the memory size of the past computer on which the early RELAP5 series codes were designed. Development of the first RELAP5 series code, RELAP5/MOD1, began about fifteen years ago. The RELAP5/MOD1 was designed to reduce memory requirements because the code was aimed at CDC computers with small memory. In the present code, the program structure, except for the heat transfer calculation, has been considerably improved. CDC or CRAY dependent bit-processings are removed. But the data structure has not been changed.

From above reasons, very much effort has been still necessary for the vectorization of MOD3 code: the manpower required was 3 months by two persons at JAERI, even started from the recent vectorized version of RELAP5/MOD2.

3.2 Vectorization difficulties and solution to them

Program structure unsuitable for vectorization

The subroutines at the top level had DO-loop and many subroutines were called within the DO-loop in order to keep the data on the small memory. An example of a typical program structure of RELAP5 is shown in Fig. 3.2.1. Fig. 3.2.1(a) illustrates the program structure of subroutine HTADV and its called subroutines in the original code.

The HTADV is the top-level subroutine which controls the heat advancements of heat structures and computes the heat added to hydrodynamic volumes. The DO-loop for heat structures, which can model fuel pins or plates with nuclear or electrical heating, etc., exists in the HTADV subroutine. However, this loop can not be vectorized because a subroutine HT1TDP is called in the DO-loop. The subroutine HT1TDP, in turn, calls the subroutines MADAT, HTCOND, ······. These bottom-level subroutines calculate the thermal conductivities and boundary conditions, and so on. The HT1TDP subroutine, called from HTADV, can not be vectorized too, since this subroutine do not have DO-loops.

This type of program structure appears in RELAP5 frequently. The vectorization of these subroutines can be achieved by the transfer of DO-loop from HTADV to HT1TDV and to its slave subroutines as shown in Fig. 3.2.1(b).

The original and vectorized codings are presented in Figs. 3.2.2(a) and 3.2.2 (b), respectively.

Data structure unsuitable for vectorization

An example of data structure in RELAP5/MOD3 is shown in Fig. 3.2.3. The data, which belong to the same volume or junction, are grouped by using EQUIVALENCE satements. In each group of volumes or junctions, physical quantities are ordered as indicated in EQUIVALENT statements. As shown in DO-loops of Fig. 3.2.3, when the physical quantities such as HV are referred over the total volumes, the memory is accessed by every IVSKP*8 byte.

This memory structure is suited for small memory computer, because the data belonging to one volume or junction would be stored on the memory. For the vector processor, however, the distanced memory access causes a memory conflict. However, we could not change this data structure for vectorization, since the whole program would be rewritten.

Many IF tests depending on the state of fluid

The code contained many IF statements depending on the fluid state. The IF tests are replaced into a list vector in order to process efficiently in vector processing. Fig. 3.2.4(a) shows the program of DO-loop transferred from the top-level subroutine. The same IF statements on the fluid state appeared several times in different positions of different subroutines. These IF statements are modified in vector version as shown in Fig. 3.2.4(b). At the top-level subroutine, the list vectors LIST1 and LIST2 are generated once. The list vector keeps the volume or junction numbers which should be calculated for a certain condition. At the bottom-level subroutnes, the list vectors are used in stead of IF statements. By using the list vectors, the redundant IF statements in the program are removed.

Another problem of IF statements is short vector length. Typical volume or junction number is 150∼300 and the number of volumes or junctions in the vectorized DO-loop is decreased by the IF branches.

```
 SUBROUTINE HTADV
┌ Do-loop for heat structures
│      :
│      :
│ CALL  HT1TDP
│      :
└ CONTINUE
  RETURN
  END
```

```
SUBROUTINE HT1TDP
  :
CALL  MADATA
  :
CALL  HTCOND
  :
RETURN
END
```

(a) Original program structure

```
SUBROUTINE HTADV
  :
CALL  HT1TDV
  :
RETURN
END
```

```
SUBROUTINE HT1TDV
  :
CALL  MADATV
  :
CALL  HTCOND
  :
RETURN
END
```

```
 SUBROUTINE MADATV
┌ Do loop-for heat structures
│
│     (vectorized)
│
└ CONTINUE
  RETURN
  END
```

(b) Vectorized program structure

Fig. 3.2.1 Example of typical program structure

```
SUBROUTINE  HTADV
 ⋮
IH = FILNDX(8)
DO  M = 1,NHTSTR(IH)
   ⋮
   ┌─────────────────────────┐
   │ Lower level subroutines │      ⇨
   └─────────────────────────┘
   ⋮
  IH = IH + 1
CONTINUE
RETURN
END
```

```
Lower level subroutine
 ⋮
HINDEX = IHTPTR(2,IH) + FILNDX(8)
K      = HTBVO(2,HINDEX) + FILNDX(11)
I      = GTBPTR(2,K)  + FILNDX(11)
 ⋮
RETURN
END
```

(a) Original program coding

```
                SUBROUTINE  HTADV
                 ⋮
                IH = FILNDX(8)
                NH¥= NHTSTR(IH)
               ┌DO  M = 1,NH¥
               │  HINX¥(M) = IHTPTR(2,IH) + FILNDX(8)
Defined        │  K = HTBVO(2,HINX¥(M)) + FILNDX(11)
  only once    ┤  GINX¥(M) = GTBPTR(2,K)  + FILNDX(11)
               │   ⋮
               │  IH = IH + 1
               └CONTINUE

                ┌─────────────────────────┐
                │ Lower level subroutines │      ⇨
                └─────────────────────────┘
                 ⋮
                RETURN
                END
```

```
Lower level subroutine
DO  M = 1,NH¥
  HINDEX = HINX¥(M)
  I      = GINX¥(M)
   ⋮
CONTINUE
RETURN
END
```

(b) Vectorized program coding

Fig. 3.2.2 DO-loop transfer in HTADV subroutine

```
      INTEGER LFSIZ
      PARAMETER (LFSIZ=270000)
      PARAMETER (NCOMS=80, NFILES=40)
      COMMON /FAST/ FA(LFSIZ)
      COMMON /COMCTL/ COMDAT(NCOMS) , COMDLN(NCOMS) , FILID(NFILES),
     *    FILSIZ(NFILES) , FILNDX(0:NFILES) , FILFLG(NCOMS+1) , SAFE1
      REAL*8 FA
      INTEGER IA(2,LFSIZ)
      EQUIVALENCE (FA(1),IA(1,1))
      INTEGER NVOLS(2,1) , VCTRL(2,1) , VOLNO(2,1) , IMAP(2,1),
     *       VOLMAT(2,1)
      REAL*8  V(1) , RHOGO(1) , RHOG(1)
      EQUIVALENCE (NVOLS(1,1) , IA(1,1)) , (VCTRL(1,1) , IA(1,2)),
     *            (VOLMAT(1,1), IA(1,3)) , (VOLNO(1,1) , IA(1,4)),
     *            (IMAP(1,1)  , IA(1,5)) , (V(1)       , FA(6)  ),
     *            (RHOG(1)    , FA(33) ) , (RHOGO(1)   , FA(101)),
      INTEGER NJUNS(2,1) , IJ1(2,1) , IJ2(2,1) , JC(2,1),
     *       IJ1VN(2,1) , IJ2VN(2,1),
      REAL*8 ARAT(1)
      EQUIVALENCE (NJUNS(1,1) , IA(1,1)) , (IJ1(1,1)   , IA(1,2)),
     *            (IJ2(1,1)   , IA(1,3)) , (JC(1,1)    , IA(1,4)),
     *            (IJ1VN(1,1) , IA(1,5)) , (IJ2VN(1,1) , IA(1,6)),
     *            (ARAT(1)    , FA(11) )
      INTEGER IJSK1
      PARAMETER (IJSK1=80)
      INTEGER IJSK2
      PARAMETER (IJSK2=0)
      INTEGER IJSKP
      PARAMETER (IJSKP=IJSK1+IJSK2)
      INTEGER IVSK1
      PARAMETER (IVSK1=128)
      INTEGER IVSK2
      PARAMETER (IVSK2=0)
      INTEGER IVSK3
      PARAMETER (IVSK3=IVSK1+IVSK2)
      INTEGER IVSK4
      PARAMETER (IVSK4=0)
      INTEGER IVSKP
      PARAMETER (IVSKP=IVSK3+IVSK4)
C
      IV    = FILNDX(4)
      IVE   = IV + IVSKP * (NVOLS(2,IV) - 1)
      IJ    = FILNDX(5)
      IJE   = IJ + IJSKP * (NJUNS(2,IJ) - 1)
C
      DO 2 I = IJ, IJE, IJSKP
        ARAT(I+1) = 1.0D0
    2 CONTINUE
C
      DO 3 I = IV, IVE, IVSKP
        RHOGO(I) = RHOG(I)
    3 CONTINUE
```

Fig. 3.2.3 Data structure of RELAP5/MOD3 code

(a) ORIGINAL CODING

```
DO ** I=IB,IE,ISKP
IF(......) THEN
   Process A
ELSE
   Process B
ENDIF
CONTINUE
```

(b) VECTORIZED CODING
TOP-LEVEL SUBROUTINE

```
L1=0
L2=0
DO ** I=IB,IE,ISKP
IF(......) THEN
 L1=L1+1
 LIST1(L1)=I
ELSE
 L2=L2+1
 LIST2(L2)=I
ENDIF
CONTINUE
```

(c) BOTTOM LEVEL SUBROUTINE

```
DO ** II=1,L1
I=LIST1(II)
  Process A
      :
```

```
DO ** II=1,L2
I=LIST2(II)
  Process B
      :
```

LIST VECTOR METHOD

Fig. 3.2.4 List vector method

3.3 Static and dynamic profils of RELAP5/MOD3 code

Huge program size

Finally, the thermohydraulic reactor transient analysis code such as RELAP5 is very large, since the real large scale complicated physical models are tested. For example, the number of subroutines is 355 including environmental library routines, and the number of Fortran statements is 110,000. This program size is the problem on vectorization.

The static profil of RELAP5/MOD2 code is presented by Fig. 3.3.1. Program tree of subroutines for the time consumming route is given by Fig. 3.3.2. In order to know the computing time distribution among subroutines, a dynamic profil of RELAP5/MOD3 code has been investigated for a typical PWR sample problem TYPPWR. The results are shown in Fig. 3.3.3. Here the subroutine name, number of Fortran stetements, number of subroutine calls, relative computing time estimate in scalar calculation, time percent, and bar graph are listed for the time-consumming 100 subroutines. From this figure we can see, no matter how we would make effort, we could not achieve the vectorization ratio 90%. At present, we have vectorized 53 subroutines and the vectorization ratio 78% is resulted.

```
          S T R E A M   7 7   ( C )      V01/L04 (19860901)        DATE (90.12.25)  TIME (20:06:11)              PAGE 0093

     THE CLASSIFICATION OF FORTRAN STATEMENTS (WITHOUT INCLUDED)
+-------------------------------------+-----------------------------------------+-----------------------------------------+
I COMMENT......................  27708 I    REAL*16                          0    I ASSIGNED GO TO...................      2 I
I GLOBAL CONTROL DIRECTIVE.....      0 I DOUBLE...........................      0 I ARITHMETIC IF....................    343 I
I LOCAL CONTROL DIRECTIVE......      0 I    NO PREFIX                        0    I 2-BRANCH ARITHMETIC IF...........      0 I
I CRAY COMPILER DIRECTIVE......    174 I    PREFIX TYPE                      0    I INDIRECT LOGICAL IF..............      0 I
I OPTIMIZATION CONTROL LINE....      0 I DOUBLE PRECISION.................      0 I LOGICAL IF.......................   6884 I
I PROCESS......................      0 I    NO PREFIX                        0    I    READ                             0    I
I INCLUDE......................   1966 I    PREFIX TYPE                      0    I    WRITE                          169    I
I EJECT........................      0 I COMPLEX..........................      0 I    PRINT                            0    I
I LIST ON/OFF..................      0 I    NO LENGTH SPECIFICATION          0    I    PUNCH                            0    I
I OVERLAY......................      0 I    PREFIX TYPE                      0    I    FIND                             0    I
I OVCAP........................      0 I    COMPLEX*8                        0    I    OPEN                             0    I
I PROGRAM......................      2 I    COMPLEX*16                       0    I    CLOSE                            0    I
I SUBROUTINE...................    308 I    COMPLEX*32                       0    I    INQUIRE                          0    I
I FUNCTION.....................     43 I CHARACTER........................    167 I    BACK SPACE                       0    I
I    NO TYPE SPECIFICATION       43    I BOOLEAN..........................      0 I    ENDFILE                          0    I
I    INTEGER                      0    I NCHARACTER.......................      0 I    REWIND                           0    I
I    INTEGER*2                    0    I DIMENSION........................     70 I    WAIT                             0    I
I    INTEGER*4                    0    I COMMON...........................     74 I    BUFFER IN                        0    I
I    INTEGER*8                    0    I EQUIVALENCE......................    242 I    BUFFER OUT                       0    I
I    LOGICAL                      0    I EXTERNAL.........................    100 I    DECODE                           0    I
I    LOGICAL*1                    0    I INTRINSIC........................      0 I    ENCODE                           0    I
I    LOGICAL*4                    0    I SAVE.............................     18 I    DELETE                           0    I
I    REAL                         0    I DATA.............................   1103 I    REWRITE                          0    I
I    REAL*4                       0    I POINTER..........................      0 I    CALL                           239    I
I    REAL*8                       0    I LEVEL............................      0 I    RETURN                          67    I
I    REAL*16                      0    I STATEMENT FUNCTION...............     28 I    STOP                             2    I
I    DOUBLE                       0    I NAMELIST.........................      0 I    PAUSE                            0    I
I    DOUBLE PRECISION             0    I DEFINEFILE.......................      0 I    CONTINUE                         0    I
I    COMPLEX                      0    I FORMAT...........................   1976 I    ASSIGNMENT                    1142    I
I    COMPLEX*8                    0    I READ.............................    141 I    ASSIGN                           0    I
I    COMPLEX*16                   0    I WRITE............................   2322 I    GO TO                         5265    I
I    COMPLEX*32                   0    I PRINT............................      9 I    COMPUTED GO TO                   0    I
I    CHARACTER                    0    I PUNCH............................      0 I    ASSIGNED GO TO                   0    I
I    BOOLEAN                      0    I FIND.............................      0 I    ARITHMETIC IF                    0    I
I BLOCK DATA...................      2 I OPEN.............................      3 I    2-BRANCH ARITHMETIC IF           0    I
I ENTRY........................     12 I CLOSE............................      0 I    INDIRECT LOGICAL IF              0    I
I IMPLICIT.....................    352 I INQUIRE..........................      1 I BLOCK IF.........................   1047 I
I PARAMETER....................     51 I BACK SPACE.......................      0 I ELSE IF..........................     66 I
I INTEGER......................    364 I ENDFILE..........................      1 I ELSE.............................    273 I
I    NO LENGTH SPECIFICATION    364    I REWIND...........................     14 I END IF...........................   1047 I
I    PREFIX TYPE                  0    I WAIT.............................      0 I DO...............................   1763 I
I    INTEGER*2                    0    I BUFFER IN........................      0 I DO WHILE.........................      0 I
I    INTEGER*4                    0    I BUFFER OUT.......................      0 I DO UNTIL.........................      0 I
I    INTEGER*8                    0    I DECODE...........................      0 I END..............................    355 I
I LOGICAL......................    197 I ENCODE...........................      0 I DEBUG............................      0 I
I    NO LENGTH SPECIFICATION    197    I DELETE...........................      0 I AT...............................      0 I
I    PREFIX TYPE                  0    I REWRITE..........................      0 I TRACE............................      0 I
I    LOGICAL*1                    0    I CALL.............................   1838 I INIT.............................      0 I
I    LOGICAL*2                    0    I RETURN...........................    480 I DISPLAY..........................      0 I
I    LOGICAL*4                    0    I STOP.............................     11 I END DEBUG........................      0 I
I    LOGICAL*8                    0    I PAUSE............................      0 I -- SYNTAX ERROR --...............      3 I
I REAL.........................    481 I CONTINUE.........................   1816 I                                         I
I    NO LENGTH SPECIFICATION      0    I ASSIGNMENT.......................  29873 I                                         I
I    PREFIX TYPE                  0    I ASSIGN...........................      4 I NO OF FORTRAN STATEMENTS     ==  56580 I
I    REAL*4                       2    I GO TO............................   2624 I NO OF COMMENTS & DIRECTIVES  ==  29848 I
I    REAL*8                     479    I COMPUTED GO TO...................     70 I                                         I
+-------------------------------------+-----------------------------------------+-----------------------------------------+
```

Fig. 3.3.1 Static Profil of RELAP5/MOD3 code

```
          S T R E A M   7 7   ( C )     V01/L04 (19860901)        DATE (90.12.25)  TIME (20:06:11)            PAGE 0094

      THE CLASSIFICATION OF FORTRAN STATEMENTS (WITH INCLUDED)
+-------------------------------------------+-------------------------------------------+-------------------------------------------+
I  COMMENT......................... 41948 I      REAL*16                     0     I  ASSIGNED GO TO................    2 I
I  GLOBAL CONTROL DIRECTIVE......     0 I  DOUBLE.........................   0 I  ARITHMETIC IF.................  343 I
I  LOCAL CONTROL DIRECTIVE.......     0 I      NO PREFIX                   0     I  2-BRANCH ARITHMETIC IF........    0 I
I  CRAY COMPILER DIRECTIVE.......   174 I      PREFIX TYPE                 0     I  INDIRECT LOGICAL IF...........    0 I
I  OPTIMIZATION CONTROL LINE.....     0 I  DOUBLE PRECISION...............   0 I  LOGICAL IF.................... 6884 I
I  PROCESS.......................     0 I      NO PREFIX                   0     I      READ                        0    I
I  INCLUDE.......................  1966 I      PREFIX TYPE                 0     I      WRITE                     169    I
I  EJECT.........................     0 I  COMPLEX........................   0 I      PRINT                       0    I
I  LIST ON/OFF...................     0 I      NO LENGTH SPECIFICATION     0     I      PUNCH                       0    I
I  OVERLAY.......................     0 I      PREFIX TYPE                 0     I      FIND                        0    I
I  OVCAP.........................     0 I      COMPLEX*8                   0     I      OPEN                        0    I
I  PROGRAM.......................     2 I      COMPLEX*16                  0     I      CLOSE                       0    I
I  SUBROUTINE....................   308 I      COMPLEX*32                  0     I      INQUIRE                     0    I
I  FUNCTION......................    43 I  CHARACTER......................  215 I      BACK SPACE                  0    I
I      NO TYPE SPECIFICATION     43     I  BOOLEAN........................    0 I      ENDFILE                     0    I
I      INTEGER                    0     I  NCHARACTER.....................    0 I      REWIND                      0    I
I      INTEGER*2                  0     I  DIMENSION......................   70 I      WAIT                        0    I
I      INTEGER*4                  0     I  COMMON.........................  993 I      BUFFER IN                   0    I
I      INTEGER*8                  0     I  EQUIVALENCE.................... 3432 I      BUFFER OUT                  0    I
I      LOGICAL                    0     I  EXTERNAL.......................  100 I      DECODE                      0    I
I      LOGICAL*1                  0     I  INTRINSIC......................    0 I      ENCODE                      0    I
I      LOGICAL*4                  0     I  SAVE...........................  781 I      DELETE                      0    I
I      REAL                       0     I  DATA........................... 1159 I      REWRITE                     0    I
I      REAL*4                     0     I  POINTER........................    0 I      CALL                      239    I
I      REAL*8                     0     I  LEVEL..........................    0 I      RETURN                     67    I
I      REAL*16                    0     I  STATEMENT FUNCTION.............   31 I      STOP                        2    I
I      DOUBLE                     0     I  NAMELIST.......................    0 I      PAUSE                       0    I
I      DOUBLE PRECISION           0     I  DEFINEFILE.....................    0 I      CONTINUE                    0    I
I      COMPLEX                    0     I  FORMAT......................... 1976 I      ASSIGNMENT               1142    I
I      COMPLEX*8                  0     I  READ...........................  141 I      ASSIGN                      0    I
I      COMPLEX*16                 0     I  WRITE.......................... 2322 I      GO TO                    5265    I
I      COMPLEX*32                 0     I  PRINT..........................    9 I      COMPUTED GO TO              0    I
I      CHARACTER                  0     I  PUNCH..........................    0 I      ASSIGNED GO TO              0    I
I      BOOLEAN                    0     I  FIND...........................    0 I      ARITHMETIC IF               0    I
I  BLOCK DATA....................     2 I  OPEN...........................    3 I      2-BRANCH ARITHMETIC IF      0    I
I  ENTRY.........................    12 I  CLOSE..........................    0 I      INDIRECT LOGICAL IF         0    I
I  IMPLICIT......................   352 I  INQUIRE........................    1 I  BLOCK IF...................... 1047 I
I  PARAMETER.....................  1377 I  BACK SPACE.....................    0 I  ELSE IF.......................   66 I
I  INTEGER.......................  3426 I  ENDFILE........................    1 I  ELSE..........................  273 I
I      NO LENGTH SPECIFICATION  3426    I  REWIND.........................   14 I  END IF........................ 1047 I
I      PREFIX TYPE                0     I  WAIT...........................    0 I  DO............................ 1763 I
I      INTEGER*2                  0     I  BUFFER IN......................    0 I  DO WHILE......................    0 I
I      INTEGER*4                  0     I  BUFFER OUT.....................    0 I  DO UNTIL......................    0 I
I      INTEGER*8                  0     I  DECODE.........................    0 I  END...........................  355 I
I  LOGICAL.......................   599 I  ENCODE.........................    0 I  DEBUG.........................    0 I
I      NO LENGTH SPECIFICATION   599    I  DELETE.........................    0 I  AT............................    0 I
I      PREFIX TYPE                0     I  REWRITE........................    0 I  TRACE.........................    0 I
I      LOGICAL*1                  0     I  CALL........................... 1838 I  INIT..........................    0 I
I      LOGICAL*2                  0     I  RETURN.........................  480 I  DISPLAY.......................    0 I
I      LOGICAL*4                  0     I  STOP...........................   11 I  END DEBUG.....................    0 I
I      LOGICAL*8                  0     I  PAUSE..........................    0 I  -- SYNTAX ERROR --............    3 I
I  REAL..........................  2982 I  CONTINUE....................... 1816 I                                     I
I      NO LENGTH SPECIFICATION    0     I  ASSIGNMENT.....................29873 I                                     I
I      PREFIX TYPE                0     I  ASSIGN.........................    4 I  NO OF FORTRAN STATEMENTS    == 68850 I
I      REAL*4                     2     I  GO TO.......................... 2624 I  NO OF COMMENTS & DIRECTIVES == 44088 I
I      REAL*8                  2980     I  COMPUTED GO TO.................   70 I                                     I
+-------------------------------------------+-------------------------------------------+-------------------------------------------+
```

Fig. 3.3.1 (Continued)

```
              S T R E A M   7 7   ( C )     V01/L04 (19860901)       DATE (90.12.25)  TIME (20:06:11)           PAGE 0095
           SUMMARY OF MESSAGES                                                                                     (001)
       (1)  MESSAGE CODES AND NUMBER OF PRINTOUTS
       +-----------------------------------------------------------------------------------------------------+
       I
       I     CODE     PRINT                                                                                  I
       I     600.....     3                                                                                  I
       I
       I  ......TOTAL NUMBER OF MESSAGE PRINTOUTS.....     3......                                           I
       I                                                                                                     I
       +-----------------------------------------------------------------------------------------------------+
```

Fig. 3.3.1 (Continued)

```
            S T R E A M   7 7   ( C )     V01/L04 (19860901)        DATE (90.12.25)  TIME (20:06:11)                PAGE 0096
      ANALYSIS INFORMATION OUTPUT INDEX                                                                                    (001)
NO.   NAME     LINES PAGE   NO.   NAME     LINES PAGE   NO.   NAME     LINES PAGE   NO.   NAME     LINES PAGE   NO.   NAME     LINES PAGE
--- -------- ----- ----   --- -------- ----- ----   --- -------- ----- ----   --- -------- ----- ----   --- -------- ----- ----
  1 AAETIT      36           91 HTADV      119          181 PHANTJ    1436          271 SCNREQ     981
  2 AATL       120           92 HTCOND     159          182 PHANTV    1368          272 SEARCH      37
  3 ACCUM      428           93 HTCSOL      44          183 PIMPLT     509          273 SETNDF      15
  4 BINOMX     143           94 HTHETA      56          184 PLOTMD      28          274 SIGMA       22
  5 CONA        36           95 HTRC1      199          185 PLOTRS     464          275 SIMPLT     606
  6 CONU        79           96 HTRC2      163          186 PLOT2D     431          276 SIMUL      151
  7 EDER        30           97 HT1INP     885          187 PLSTRN     184          277 SMOOTH     393
  8 FUNX       103           98 HT1SST     287          188 PLTREC      47          278 SORPTR      98
  9 GIBBAB     242           99 HT1TDP     423          189 PLTWRT     181          279 SPLINT     101
 10 HELMCD     119          100 HT2TDP     603          190 PMINVD     498          280 SPLIN2      46
 11 INDEXX      17          101 HYDRO      111          191 ENVBLK       8          281 SPL1D1     216
 12 POLATT      47          102 HYPSAT      39          192 PMINVF     155          282 SPL1D2      51
 13 POLYN       70          103 HYSATT      93          193 PMINVM      77          283 SPL2D1      39
 14 PSATK       42          104 HYTHC       36          194 PMINVR      58          284 SPL2D2      82
 15 PSATL       17          105 HYVISC      27          195 PMINV1     179          285 SPL2D3      18
 16 ROOT       232          106 HZFLOW    1353          196 PMINV4      29          286 SQOZ        23
 17 UNITS       47          107 ICMPF       26          197 POLAT       65          287 SRESTF     155
 18 WRITEA      68          108 ICMPN1     438          198 POLATR      82          288 SSTCHK     841
 19 IASME      162          109 ICOMPN     735          199 PREDNB     241          289 STACC      496
 20 ASMEBD     295          110 ICOMPT     282          200 PRESEQ     779          290 STATE      289
 21 AXISDV      37          111 ICONVR     749          201 PSATPD      95          291 STATEP    1625
 22 BLKDTA     143          112 IDFIND      31          202 PSET        26          292 STD2X3     430
 23 BRYCEJ      71          113 IEDIT      392          203 PSTDNB     309          293 STD2X4     546
 24 CCFL       157          114 IELVTN     356          204 PSTPD2      95          294 STD2X5     642
 25 CDENTH      74          115 IGNTBL      77          205 PSTPHY      58          295 STHYXI      32
 26 CELMDR      32          116 IHTCMP     958          206 PUMP       179          296 STHYXJ      32
 27 CHFCAL    2457          117 IJPROP     220          207 PUMP2      134          297 STHYXO      28
 28 CHRINT     146          118 IMIEDT     127          208 QFHTRC     125          298 STHYX1     137
 29 CKPCRD     210          119 IMLP       809          209 QFMOVE     571          299 STHYX3     357
 30 CKTCDS     184          120 INP        289          210 QFSRCH     420          300 STHYX6     540
 31 CONDEN     235          121 INPLNK      74          211 QMWR       137          301 STH2XG     153
 32 CONVAR     394          122 INPMOD      86          212 RACCUM    1117          302 SAT1        94
 33 CPLEXP      37          123 INPPCK      25          213 RADHT      132          303 SAT2        98
 34 CRAMER      70          124 INPUPK      26          214 RBRNCH    1387          304 SNPH       142
 35 CTHXPR      64          125 INPUTD     127          215 RCARDS      97          305 INTGRC     123
 36 CVIC       394          126 INP10      106          216 RCDELT     104          306 DELS       108
 37 CVIRC       15          127 INP2       146          217 RCHNG       35          307 STH2XI      35
 38 DETMNT     133          128 INP4        43          218 RCKPLT     469          308 STH2XJ      35
 39 DITTUS      81          129 INP5       162          219 RCOMPN     260          309 STH2XU     202
 40 DMPFIL      92          130 INP6        19          220 RCOMPT     417          310 STH2XO      30
 41 DMPLST      50          131 INP7        10          221 RCONVR    1610          311 STH2X1     205
 42 DOLEND     120          132 INP8        38          222 RCRVSP     159          312 STH2X3     373
 43 DPSAT       33          133 INP9        99          223 RDREDT      16          313 STH2X4     484
 44 DTSTEP     439          134 INTERI     103          224 REEDIT      16          314 STH2X5     599
 45 ECCMXJ     525          135 INVJT      134          225 RELAPS     201          315 STH2X6     594
 46 ECCMXV     696          136 INXGET      15          226 REMTIM       8          316 STOLST      37
 47 EPRIJ      236          137 IPIPE       42          227 RGNTBL     480          317 STRIP      200
 48 EQFINL     881          138 IPLOAD     270          228 RHTCMP    1082          318 SURTEN      35
 49 FABEND       9          139 IPLOT      229          229 RINTRV     181          319 SURTNV      42
 50 FIDIS       82          140 IPLT2D     591          230 RKIN       553          320 SURTN2      31
 51 FIDISJ     510          141 IPUMP      331          231 RMADAT     701          321 SVH2X2     307
 52 FIDIS2      99          142 IRADHT     317          232 RMBLNK      39          322 SWAPLT      80
 53 FOURT      555          143 IRFLHT     342          233 RMFLDS     121          323 SYSITR     217
 54 FTBCLS      46          144 IRKIN      473          234 RMIEDT     109          324 SYSSOL     135
 55 FTBCPY      30          145 ISFDES      13          235 RMTPLJ     776          325 TCNVSL      68
 56 FTBDEL      51          146 ISFOPN      16          236 RNEWP      223          326 THCNFV      76
```

Fig. 3.3.1 (Continued)

```
 57 FTBDSB    47      147 ISNGJ    160      237 RNONCN   165      327 THCNGV    51
 58 FTBERR    14      148 ISSI     619      238 RNPLNK    52      328 THCON     59
 59 FTBEXP    57      149 ISTATE  1543      239 RNP2      73      329 THCON2    70
 60 FTBFTB    17      150 ITRIP    303      240 RONOFF    24      330 TIMSET    29
 61 FTBGET    32      151 ITRSCN    55      241 RPIPE   1340      331 TPAWHT    74
 62 FTBINT    64      152 IUSRVR    79      242 RPLOPS   282      332 TRAN     130
 63 FTBLCT    95      153 IVELST   140      243 RPLOT     98      333 TRIP     105
 64 FTBMEM    14      154 IVLVEL   206      244 RPLOTF   342      334 TRNCTL    17
 65 FTBMOV    15      155 JCHOKE   884      245 RPLOTN   422      335 TRNFIN    73
 66 FTBNID    42      156 JPROP    420      246 RPLT2D   437      336 TRNSET   955
 67 FTBOPN   110      157 KATOKJ   168      247 RPMPDC   480      337 TSETSL   386
 68 FTBCHK     9      158 KLOSS     77      248 RPMPMD   164      338 TSTATE   828
 69 FTBIN     18      159 LABRT     26      249 RPMVNJ   892      339 TURBST    87
 70 FTBOUT    19      160 LAVAIL     8      250 RPUMP   1188      340 UNSQOZ    24
 71 FTBPR1    75      161 LCNTGS    33      251 RPUNIT   116      341 VALVE    544
 72 FTBPR2     8      162 LCONTG    23      252 RRADHT   704      342 VEXPLT   910
 73 FTBPR3     9      163 LIFOPN    41      253 RRESTF   382      343 VFINL    388
 74 FTBPR4    10      164 LOCF      14      254 RRKIN   1583      344 VIMPLT  1238
 75 FTBRDC    46      165 LOCFI     14      255 RRKINH   241      345 VISCG2    67
 76 FTBRSV    29      166 LOCFI4    14      256 RRSTD     71      346 VISCOG    56
 77 FTBSFT    66      167 LOCF4     14      257 RSIZPL    33      347 VISCOL    77
 78 FTBSLK    64      168 MADATA   225      258 RSNGJ    546      348 VISCVG    84
 79 FTBINC    67      169 MAINS    119      259 RSNGV    665      349 VISCVL   119
 80 FWDRAG   450      170 MAJOUT  1387      260 RSSI     458      350 VLVELA   332
 81 GAPCON   186      171 MDATA2   435      261 RSTREC    80      351 VOLVEL   490
 82 GASCON   165      172 META       8      262 RSTRIP   147      352 WFRICJ   293
 83 GASTHC    68      173 MIREC     83      263 RTMDJ    500      353 WRITPL    81
 84 GNINIT   214      174 MOVER    186      264 RTMDV    925      354 WRPLID   328
 85 GRDNRJ   114      175 MXSETS    23      265 RTRIP    471      355 ZFSLGJ    99
 86 GTINF     64      176 NFSETS    14      266 RTSC     181
 87 HELPHD    21      177 NFSIZE    15      267 RTURB   1327
 88 HELPLT   230      178 NFUNIT    16      268 RUPLAS    65
 89 HIFBUB    73      179 NUMCHR    28      269 RUSRVR   103
 90 HLOSS    386      180 PACKER   142      270 RVALVE  1372

     ===== TOTAL =====    NUMBER OF LINES.....   92780    ( WITH INCLUDED .....  133716 )
                          NUMBER OF STEPS.....   56580    ( WITH INCLUDED .....   68850 )
```

Fig. 3.3.1 (Continued)

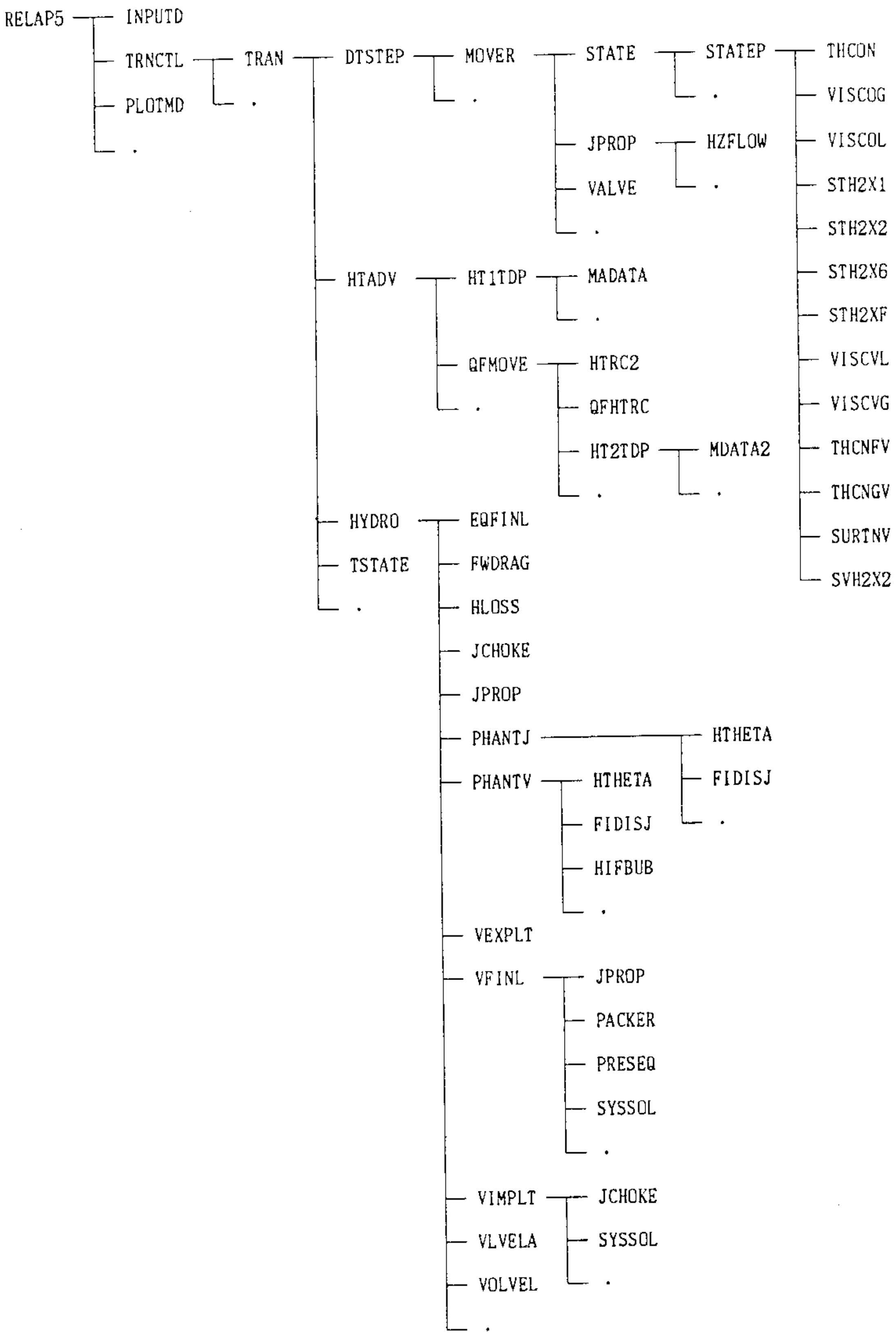


Fig. 3.3.2 Program tree of the time-consumming subroutines

```
  NO.    ROUTINE UNITS LINES ERR.   EXECUTIONS        COST      %      0..................................................

+-------------------------------------------------------------------------------------------------------------------------------------------+
I 0001   PHANTJ     1   1735    0          686     138081087    9.1    I**************************************************                  I
I 0002   PHANTV     1   1667    0          686     107479638    7.0    I**************************************                              I
I 0003   PRESEQ     1   1113    0         1268     102591187    6.7    I************************************                                I
I 0004   VEXPLT     1   1301    0          686      74276255    4.9    I**************************                                          I
I 0005   FWDRAG     1    709    0          686      72075775    4.7    I*************************                                           I
I 0006   SVH2X2     1    424    0         2085      67419228    4.4    I************************                                            I
I 0007   STH2X6     1    602    0         1893      65609402    4.3    I***********************                                             I
I           STH2XF                       92504                         I                                                                    I
I 0008   EQFINL     1   1160    0          686      58207022    3.8    I********************                                                I
I 0009   PMINVF     1    164    0         1266      56977226    3.7    I********************                                                I
I 0010   MADATA     1    379    0        57765      52981836    3.5    I*******************                                                 I
I 0011   STATEP     1   1847    0          695      48616243    3.2    I*****************                                                   I
I 0012   MAJOUT     1   2103    0            6      41222499    2.7    I**************                                                      I
I 0013   HT1TOP     1    824    0        56938      33420664    2.2    I************                                                        I
I 0014   VOLVEL     1    764    0          686      32002146    2.1    I***********                                                         I
I 0015   JCHOKE     1   1135    0          686      31870209    2.1    I***********                                                         I
I 0016   INP        1    298    0            2      30272925    2.0    I**********                                                          I
I 0017   CONDEN     1    449    0        41094      25240810    1.7    I*********                                                           I
I 0018   INP9       1    107    0          131      24744536    1.6    I********                                                            I
I 0019   PREDNB     1    486    0        18291      24111484    1.6    I********                                                            I
I 0020   HTHETA     1     58    0        64360      23685608    1.6    I********                                                            I
I 0021   KATOKJ     1    349    0        40808      22259742    1.5    I********                                                            I
I 0022   HLOSS      1    646    0          686      22082427    1.4    I*******                                                             I
I 0023   VISCVL     1    230    0         2085      19876305    1.3    I*******                                                             I
I 0024   VFINL      1    645    0          686      19729822    1.3    I*******                                                             I
I 0025   VLVELA     1    606    0          627      19437896    1.3    I*******                                                             I
I 0026   EPRIJ      1    429    0        17674      18783985    1.2    I******                                                              I
I 0027   JPROP      1    706    0         2744      18361544    1.2    I******                                                              I
I 0028   FIDIS      1    180    0        92565      17753499    1.2    I******                                                              I
I 0029   PMINVM     1     79    0         1268      16377892    1.1    I******                                                              I
I 0030   HIFBUB     1    180    0        58687      16377296    1.1    I******                                                              I
I 0031   DITTUS     1    295    0        68494      15341590    1.0    I*****                                                               I
I 0032   HZFLOW     1   1604    0         2744      13790201    0.9    I****                                                                I
I 0033   FIDIS2     1    280    0        42672      12281772    0.8    I****                                                                I
I 0034   FTBMOV     1     17    0        53190      11969837    0.8    I****                                                                I
I 0035   FIDISJ     1    735    0        58482      10627947    0.7    I***                                                                 I
I 0036   CHFCAL     1   2689    0        18291       9713323    0.6    I***                                                                 I
I 0037   THCNGV     1     53    0         2085       9484665    0.6    I***                                                                 I
I 0038   HTCOND     1    428    0        57765       8944209    0.6    I***                                                                 I
I 0039   PSET       1     78    0       175974       8396112    0.5    I**                                                                  I
I 0040   PMINVR     1     61    0         1266       7569745    0.5    I**
I 0041   HTRC1      1    431    0        88392       7280414    0.5    I**
I 0042   DTSTEP     1    822    0          687       6773837    0.4    I**
I 0043   STH2XO     1     32    0        56938       6604808    0.4    I**
I 0044   STATE      1    623    0          695       6404889    0.4    I**
I 0045   CVIC       1    403    0         1520       6101568    0.4    I**
I 0046   PACKER     1    393    0          686       5460903    0.4    I**
I 0047   HTADV      1    414    0          686       5324265    0.3    I*
I 0048   VISCVG     1    195    0         2085       4870560    0.3    I*
I 0049   TIMSET     1     31    0            1       4419106    0.3    I*
I           TIMEL                       137409                         I
I           TIMER                          687                         I
I 0050   MOVER      1    444    0          686       4376835    0.3    I*
I 0051   TSTATE     1   1120    0          686       4376021    0.3    I*
I 0052   PSATPD     1     97    0        19663       3948686    0.3    I*
I 0053   THCNFV     1     78    0         2085       3886120    0.3    I*
```

Fig. 3.3.3 Dynamic profil of RELAP5/MOD3 code

```
I 0054   IEDIT       1    743   0        1    3624709   0.2  I*                                          I
I 0055   RKIN        1    925   0      627    3420345   0.2  I*                                          I
I 0056   RPIPE       1   1582   0       23    2717330   0.2  I*                                          I
I 0057   WRPLID      1    785   0        1    2711380   0.2  I*                                          I
I 0058   SURTNV      1     44   0     2085    2454045   0.2  I*                                          I
I 0059   RHTCMP      1   1346   0        1    2367075   0.2  I*                                          I
I 0060   STH2X1      1    213   0     3487    2153658   0.1  I                                           I
I          STH2XB                        0                   I                                           I
I          STH2X2                      258                   I                                           I
I 0061   SQOZ        1     31   0      101    1601961   0.1  I                                           I
I 0062   RBRNCH      1   1638   0       15    1474371   0.1  I                                           I
I 0063   CONVAR      1    579   0      627    1297890   0.1  I                                           I
I 0064   POLAT       1     67   0    19309    1242658   0.1  I                                           I
I 0065   RCONVR      1   1699   0        1    1223267   0.1  I                                           I
I 0066   IHTCMP      1   1249   0        1    1129106   0.1  I                                           I
I 0067   CCFL        1    420   0      686    1096228   0.1  I                                           I
I 0068   ACCUM       1    783   0      686    1049772   0.1  I                                           I
I 0069   PLTWRT      1    774   0      101     924958   0.1  I                                           I
I 0070   HT1INP      1   1027   0       16     918397   0.1  I                                           I
I 0071   IMLP        1   1081   0        1     905142   0.1  I                                           I
I 0072   STACC       1    791   0      695     829914   0.1  I                                           I
I 0073   RSNGJ       1    788   0       16     808924   0.1  I                                           I
I 0074   PMINVD      1    507   0        2     757000   0.0  I                                           I
I 0075   TRAN        1    279   0        1     739485   0.0  I                                           I
I 0076   RTMDV       1   1167   0       14     738832   0.0  I                                           I
I 0077   IELVTN      1    580   0        1     603104   0.0  I                                           I
I 0078   MIREC       1    137   0      102     582829   0.0  I                                           I
I 0079   TRIP        1    180   0      627     462655   0.0  I                                           I
I 0080   INPLNK      1     82   0     2803     413140   0.0  I                                           I
I 0081   RSNGV       1    907   0       11     392196   0.0  I                                           I
I 0082   FTBLCT      1    116   0     1225     384606   0.0  I                                           I
I 0083   HT1SST      1    651   0       83     378936   0.0  I                                           I
I 0084   VALVE       1    897   0      745     376822   0.0  I                                           I
I 0085   RTMDJ       1    736   0       10     374660   0.0  I                                           I
I 0086   INPMOD      1     94   0     1672     321815   0.0  I                                           I
I 0087   TSETSL      1    668   0        1     278396   0.0  I                                           I
I 0088   INVJT       1    380   0        1     268004   0.0  I                                           I
I 0089   RVALVE      1   1658   0        4     255269   0.0  I                                           I
I 0090   SCNREQ      1   1644   0      186     252127   0.0  I                                           I
I 0091   RPMVNJ      1   1173   0        2     242708   0.0  I                                           I
I 0092   INP2        1    155   0     1244     221758   0.0  I                                           I
I 0093   PUMP        1    465   0      686     197764   0.0  I                                           I
I 0094   PUMP2       1    410   0     1374     189612   0.0  I                                           I
I 0095   TRNFIN      1    119   0        1     187395   0.0  I                                           I
I 0096   RACCUM      1   1385   0        2     173220   0.0  I                                           I
I 0097   RRKIN       1   1672   0        1     150504   0.0  I                                           I
I 0098   ICONVR      1    973   0        1     146998   0.0  I                                           I
I 0099   TRNSET      1   1725   0        1     144261   0.0  I                                           I
I 0100   RMADAT      1    778   0        1     137124   0.0  I                                           I
I 0101   INP5        1    165   0      309     130887   0.0  I                                           I
I 0102   RSTREC      1    141   0        3     130403   0.0  I                                           I
I 0103   ITRIP       1    382   0        1     129065   0.0  I                                           I
I 0104   DMPLST      1     72   0        1     125674   0.0  I                                           I
I 0105   SYSSOL      1    367   0     1268     121190   0.0  I                                           I
I 0106   SETNDF      1     17   0      135     119713   0.0  I                                           I
I 0107   POLATR      1     84   0     1374     113354   0.0  I                                           I
I 0108   STH2X3      1    381   0      161     106349   0.0  I                                           I
I 0109   RPMPMD      1    434   0        1     100458   0.0  I                                           I
I 0110   PHINV4      1     31   0        4      95660   0.0  I                                           I
I 0111   HYDRO       1    181   0      686      91729   0.0  I                                           I
I 0112   ISTATE      1   1817   0        1      91247   0.0  I                                           I
I 0113   INPUPK      1     34   0     1672      84679   0.0  I                                           I
```

Fig. 3.3.3 (Continued)

3.4 Vectorization methods for heat transfer calculation

The heat transferred across solid boundaries of hydrodynamic volumes is calculated using heat structures[*4]. The heat structures are able to model fuel pins or plates with nuclear or electrical heating, heat transfer across steam generator tubes, and heat transfer from pipe and vessel walls, and so on. The heat structures are represented by one-dimensional heat conduction. Finite differences are used to advance the heat conduction solutions.

Fig 3.4.1 illustrates the layout of mesh points at temperatures to be calculated. The finite difference approximation for mesh points leads to a tri-diagonal system of M_i equations, where M_i denotes the number of mesh points for the heat structure i. In the original code, these equations were solved using the Gaussian elimination method. And the heat transfer rate κ, and the heat capacity coefficient ρCp are obtained by table look-up.

The present vectorized subroutines at JAERI are summarized in Table 3.4.1. The vectorization methods used are summarized as follows[*9]:

(1) Since the heat advancement can be calculated in parallel for all the heat structure, the heat structure DO-loop in the subroutine HTADV, which is the top-level subroutine for heat transfer calculation, is transferred into the invoked subroutine HT1TDP (see Fig. 3.2.1).

(2) Program block which contains a calculation on heat mesh points or intervals is doubly nested DO-loops in nature since heat structure loop exists in the outer. In this case, the double DO-loops are reformed as a single DO-loop, if possible, in order to enlarge the vector length.

<Index transformation from doubly-nested DO-loop to a single loop>

The heat structure geometry data and tempretures are treated for each mesh point or mesh interval. The calculated data were stored in a two-dimensional array XXX¥(ℓ, i) as shown in Fig. 3.4.2. In order to process these data efficiently, doubly nested DO-loops are transformed into a single loop as follows, where the arrays XXX¥(L, I) and ZXXX¥(LCOLS ¥(J)) share a memory by EQUIVALENCE statement:

```
      DO 1   I=1,¥NH                      DO 1   J=1, ¥LCOLS
       DO 1  L=1,COLS ¥(1)        ===>   1   ZXXX¥( LCOLS¥(J)) = ····
    1    XXX¥(L,I) = ····
```

(3) The original method which was applied to the solution of heat transfer equations was the Gaussian elimination which includes unvectorizable recursive formula. In the vectorized code, the inner/outer DO-loops are changed in position and then the heat structure-loop is vectorized.

<Vectorized solution of the heat transfer equation>

The data structure of the coefficient matrix of heat transfer equations is not changed in the vector version. Use of the data area of coefficient matrix for a heat structure is shown in Fig. 3.4.3. Here three arrays HTE¥, HTB¥, and HTF¥ are used for keeping values "a and c", "b", and "d" when general form

$$a_j T_{j-1} + b_j T_j + c_j T_{j+1} = d_j$$

is assumed, where the matrix is not necessarily symmetric because values c_1 and a_M become zero depending on the heat structure geometry.

The vectorization method applied to solve the one-dimensional heat conduction calculation is shown in Fig. 3.4.4. In the original code, the program is coded as doubly nested DO-loops. The inner loop index m is for heat mesh points, and the outer loop index i is for heat structures as shown in Fig. 3.4.4(a). The maximum number of heat mesh points and heat structures is indicated ¥MNN and ¥NH, respectively.

This nested DO-loops solved N-set of tri-diagonal system of equations for heat mesh points of N-heat structures (= ¥NH). For the vectorization, the DO-loop indices are inverted because the DO-loop for the heat mesh points is recursive and the vector length is very small (3～10), but the heat structures have no data dependency each other.

Besides, the indices of heat structure "i" are changed in accending order of number of mesh points in heat structures as shown in Fig. 3.4.4(b). A list vector is provided to keep the maximum heat mesh numbers corresponding to heat mesh indices. By using this list vector, we could avoid the IF tests in the inner loop when number of mesh points is different among heat structures.

(4) The forced convection heat transfer correlations and so on were originally calculated on the left and right boundaries of each heat structure in serial. The program is reformed in order to calculate on left and right boundaries over all heat structures in parallel. From this reform, the vector length becomes double.

<Integration of the left and right boundary treatments>

Data structure of boundaries is illustlated as Fig. 3.4.5, where the left and right boundaries are arranged alternatively for each heat structure. By applying this layout, the calculation order does not changed in vector version. The change of calculation order in heat boundaries sometimes brings a discrepancy of the computed results. Especially, in the top-level subroutine HTADV (see Fig. 3.3.1), heat transfers from heat boundaries are added into the hydrodynamic volumes. The order of the summation at this time is one of the factors which cause the difference of computed results between vector and scalar processings.

The list vectors generated for the boundary treatments are as follows:

LB1¥(k)	to transform the boundary indices assigned as Fig. 3.4.5 into the indices for a heat structure,
LBL¥(k)	to extract boundary mesh points,
LBN¥(k)	to extract boundary intervals,
LBNO ¥(k)	to distinguish the left and right boundaries (left = 0, right =1).

(5) List vectors are generated once and later used when IF branch conditions in a DO-loop are fixed during a time step. In the vectorized version, a DO-loop is constructed corresponding to a branch condition. The data reference or update in the DO-loop is replaced to indirectly addressed ones using the list vector which satisfies the branch condition (see Fig. 3.2.4).

(6) The invariable list vectors which are frequently used are reserved as one-dimensional tables. We make tables for two purposes: for the index transformation from doubly nested DO-loop to a single loop as described in (2) and for the integration of the left and right boundary treatments as described in (4).

Table 3.4.1 Vectorized subroutines for heat transfer calculation

Subroutine name		Contents	Vectorized index
Original	Vectorized		
HTADV	HTADV	Controls advancements of heat structures and computes heat added to hydrodynamic volumes.	Left and right heat boundaries
HT1TDP	HT1TDV	Advance one heat structure one time step by advancing the transient heat transfer equations.	Heat structures, Heat meshes, etc.
MADATA	MADATV	Computes thermal conductivity and volumetric heat capacity for each mesh intervals.	Total mesh-intervals
HTCOND	HTCONV	Returns left and right boundary conditions for a heat structure.	Heat boundaries
HTRC1	HTRC1V	Computes heat transfer coefficient from correlations.	Heat boundaries connected with hydrodynamic volumes
DITTUS	DITTSV	Computes Dittus-Boelter forced convection heat transfer correlation.	〃
CONDEN	CONDEV	Computes condensation heat transfer correlations.	〃
PREDNB	PREDNV	Computes Pre-DNB forced convection heat correlations.	〃
CHFCAL	CHFCAV	Computes the critical heat transfer flux using ZUBER and BIASI CHF correlations.	〃

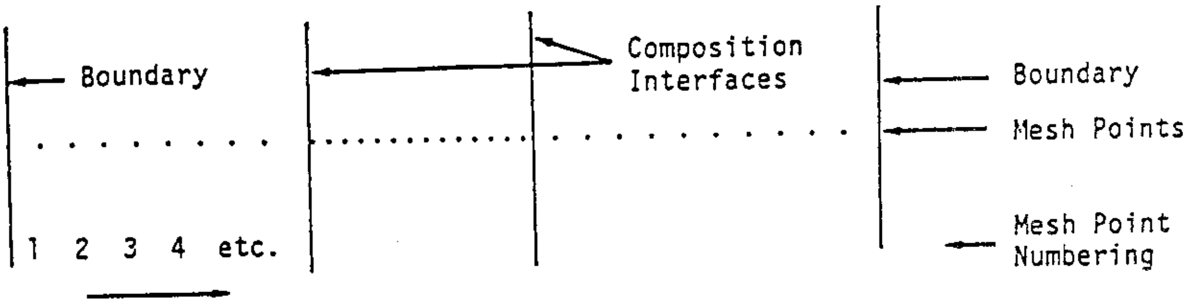
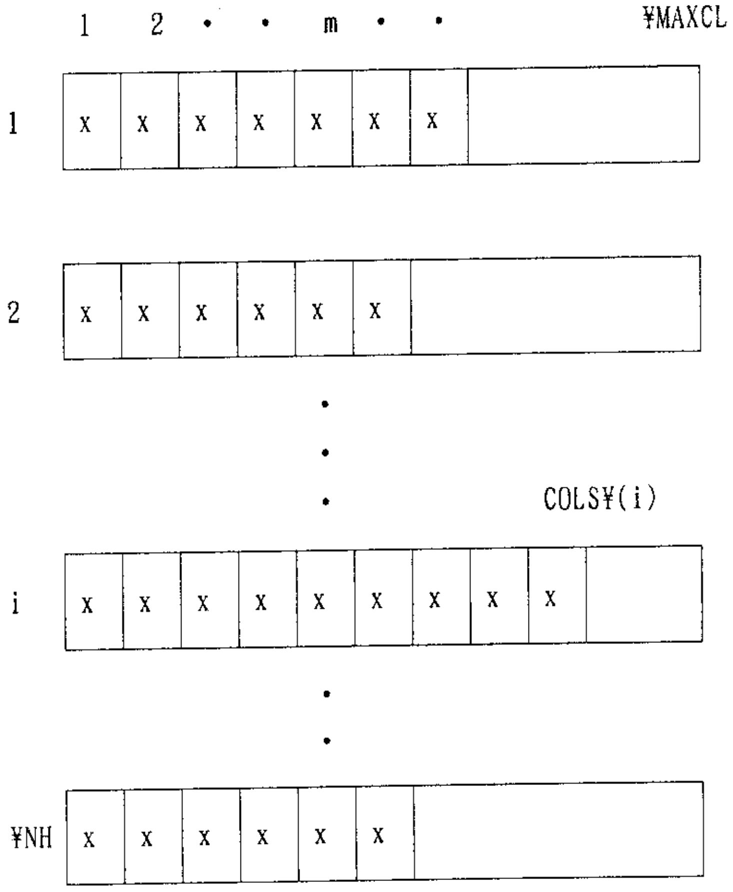


Fig. 3.4.1 Mesh point layout



¥MAXCL: Upper limit of the mesh number of heat
structure (not large)

¥NH: Number of heat structures

COLS ¥(i): Number of mesh points for a heat
structure i

Fig. 3.4.2 Data structure of heat mesh points in the original code

	1	2	3	······ M-1	M	M+1
HTE¥	c_1,	$a_2=c_3$	$a_3=c_4$	$a_{M-1}=c_{M-2}$	c_{M-1}	a_M
HTB¥	b_1	b_2	b_3		b_M	
HTF¥	d_1	d_2	d_3		d_M	

Fig. 3.4.3 Use of data area for coefficient matrix of heat transfer equations

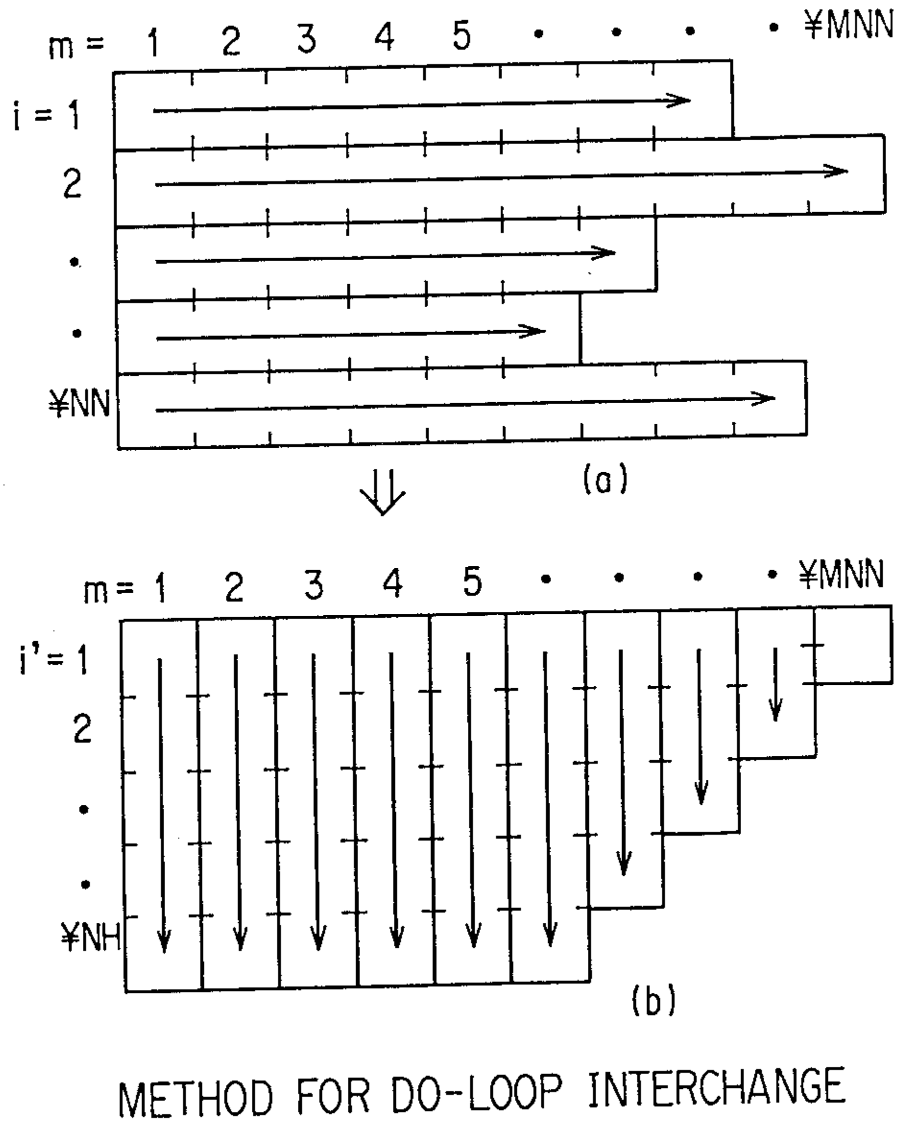


Fig. 3.4.4 Method of DO-loop inversion

1	2	3	4	••• k •••	¥NH2-1	¥NH
left1	right2	left2	right2		lt ¥NH	rt ¥NH

¥NH2: = number of boundaries
= number of heat structures(¥NH) * 2

Fig. 3.4.5 Data structure of boundaries

3.5 Vectorization method for hydrodynamic calculation

The RELAP5 hydrodynamic model *4 is a one-dimensional, transient two-fluid model for flow of two-phase steam water nature which can contain a noncondensible component in the steam phase and/or nonvolatile component in the water. Six equation model with an additional equation for noncondensible gas components is applied as the field equation.

The two-fluid equations of motion which are used as the basis for the RELAP5 hydrodynamic model are formulated in terms of space and time averaged parameters of flow. The system model is solved numerically using a semi-implicit finite difference method *4.

Program structure of the hydrodynamic model is rather serial and each submodels is called sequentially (see Fig. 3.3.2). Most of the calculation can be performed in parallel for spatial meshes. Since the staggered mesh method is applied, nodes (=volumes) or junctions can become vectorizable indices. For most part of the DO-loops, vectorizable codings were already prepared by INEL.

The vectorized subroutines at JAERI are summarized in Table 3.5.1.
The vectorization method of each subroutine is as follows *9:

(1) Subroutine EQFINL

Most of DO-loops in this subroutine can be vectorized on junctions or volumes without restructuring. Only problem is several junction loops which contains multiple assignment statements. Such recursive loops appear for adding the convective terms to the source terms and for computing the Boron density, and so on. These loops are, therefore, isolated and scalar option is specified.

(2) Subroutine FWDRAG

This subroutine consists of volume-loops, so the vectorization is very easy. Insert of compiler directives (*VOCL on FACOM vectorizing compiler) is the useful means.

(3) Subroutine HLOSS

This subroutine consists of a junction-loop. The calculation is skipped for most junctions by an IF branch appearing at the top of the program. Therefore, in the vectorized version, the DO-loop is divided into several blocks before the IF statements and list vectors are generated if sufficiently large vector length is expected.

(4) Subroutine JCHOKE

The last half of this subroutine contains a type of recursive formula to compute the flow with sound speed but the computing time spent in this part is very small. So, we vectorize only the first half by making use of the parallelizm of junctions.

(5) Subroutine JPROP

As similar as subroutine FWDRAG, modification is not so much. The compiler directives are inserted.

(6) Subroutine HZFLOW

This subroutine consists of a junction-loop with big program body but the most part of calculation is skipped for all most all junctions, except for the last program sequence. Therefore, the last program sequence is isolated as a DO-loop and vectorized on the junctions.

(7) Subroutine PHANTJ

The subroutines PHANTJ and FHANTV are new ones in the RELAP5/MOD3 code. The PHAINT subroutine of RELAP5/MOD2 are divided into PHANTJ and PHANTV which mainly contains junction-loops and volume-loops, respectively.

PHANTJ consists of doubly nested DO-loops of plant-components and their junctions. In the vectorized version, this double loops are reformed into a single loop. Number of statements in the DO-loop is so many that can be effectively vectorized. So, we divide the loop into several blocks. Flow of the vectorized PHANTJ is illustrated in Fig. 3.5.1.

Subroutine calls for HTHETA in the DO-loop are isolated and the DO-loop are transferred into the invoked subroutine HTHETA to be vectorized in it. A backward GOTO statement exists for some junctions when the horizontal flow map calculation (HMAP) is required (see Fig. 3.5.1). We generate two kinds of list vectors; one is for the junctions used in usual program path and another is for the junctions on which the HMAP option is specified. If short vector length is expected such as dry wall correlation calculation, scalar loops are applied.

(8) Subroutune HTHETA

This subroutine is called from PHANTJ and PHANTV. The volume loop existing in both subroutines are transferred and vectorized on the volumes. The convergence calculation is explicitly written down in the vectorized DO-loop and solved for all volumes in parallel.

Different assignment statements are used in the subroutines PHANTJ and PHANTV after the calls for HTHETA. This assignment satement is to be moved into each child subroutine. Accordingly, we prepare two subroutines of vectorized HTHETA; one is named HTHEVJ for the subroutine PHANTJ and another is HTHETV for PHANTJ. Flow of the HTHETV is shown in Fig. 3.5.2.

(9) Subroutine PHANTV

As similar as the subroutine PHANTJ, doubly-nested DO-loops of components and volumes are reformed into a single DO-loop and the single loop is divided into several blocks in order for efficient vectorization. The subroutine calls for HTHETA, FIDIS and HIFBUS in the DO-loops are vectorized in each slave routines.

(10) Subroutine FIDIS

There is no DO-loop in the original FIDIS subroutine. The volume loop in the subroutine PHANTV is transferred into the FIDIS. Flow of the vectorized subroutine is shown in Fig. 3.5.3. In the subroutine PHANTV, the vectorized FIDIS (= FIDISV) is called when large vector length is expected but otherwise, such as dry wall correlation calculation, the original FIDIS is called.

(11) Subroutine HIFBUS

The vectorization method of this subroutine is similar to subroutine FIDIS. Both the vectorized subroutine HIFBUV and the original scalar subroutine HIBUS are used accoding to the vector length. Flow of the vectorized subroutine is shown in Fig. 3.5.4.

(12) Subroutine PRESEQ

Most of DO-loops of this subroutine are vectorizable on junctions or volumes. The problem is the multiple assignment statements in DO-loops to compute the coefficient matrix and source terms of pressure equation, where convective terms on adjacent volumes ("to volume" and "from volume") are added to each junction. The loops can not be vectorized. So, we leave them as scalar.

(13) Subroutine VEXPLT

Compiler directives are inserted.

(14) Subroutine VFINL

Compiler directives are inserted.

(15) Subroutine PACKER

WRITE statements in the DO-loop are moved outside and gathered as another loop so that the remaining part of the program can be vectorized

(16) Subroutine VLVELA

Compiler directives are inserted.

(17) Subroutine VOLVEL

Compiler directives are inserted.

Table 3.5.1 Vectorized subroutines for hydrodynamic calculation

Subroutine names Original	 Vectorized	Contents	Vectorized index
EQFINL	EQFINL	This subroutine computes the new time pressure and carries out the back substitution to obtain the new time liquid specific internal energy, vapor specific internal energy, void fraction internal noncondensible quality, and boron density.	VOLUMES JUNCTIONS
FIDIS	FIDISV	Computes interphase drag terms.	VOLUMES
FWDRAG	FWDRAG	Computes wall drag terms...include flow regimes and correlation.	VOLUMES
HIFBUB	HIFBUV	Computes liquid HIF for bubbly flow.	VOLUMES
HLOSS	HLOSS	Calculates void fractions at throat and downstream of an abrupt area change and assosiated head loss terms.	JUNCTIONS
HTHETA	HTHEAV HTHEVJ	Calculation of horizonal stratification angle.	VOLUMES JUNCTIONS
HZFLOW	HZFLOW	Vapor pull-through and liquid entrainment model for stratified horizontal flow.	JUNCTIONS
JCHOKE	JCHOKE	Computation of choking theory.	JUNCTIONS
JPROP	JPROP	Donors junction properties from adjacent volume quantities.	JUNCTIONS
PACKER	PACKER	This subroutine determines if water packing occurs, and if it does, modifies terms used in the velocity equations.	VOLUMES
PHANTJ	PHANTJ	Computes interphase drag and also calculates some information for VEXPLT. (for junction loop)	JUNCTIONS
PHANTV	PHANTV	Computes heat transfer and also calculates some information for VEXPLT.(for volume loop)	VOLUMES
PRESEQ	PRESEQ	Using the phasic equations for mass and energy to eliminate liquid specific internal energy, vapor specific internal energy, void fraction, and noncondensible quality, this subroutine builds the matrix elements and the source vector elements for the resultant pressure equation.	VOLUMES JUNCTIONS
SYSSOL	SYSSOL	Solves system of equations using sparse matrix subroutines.	Non-zero matrix elements
VEXPLT	VEXPLT	Computes the explicit liquid and vapor velocities and the pressure gradient coefficients needed for the inplicit pressure solution.	VOLUMES JUNCTIONS
VFINL	VFINL	Computes new velocities from the pressure solution.	JUNCTIONS
VLVELA	VLVELA	Calculates average volume velocities by averaging. The average junction velocities in and out of the volume.	VOLUMES
VOLVEL	VOLVEL	Calculates average volume velocities by averaging. The average junction velocities in and out of the volume.	JUNCTIONS

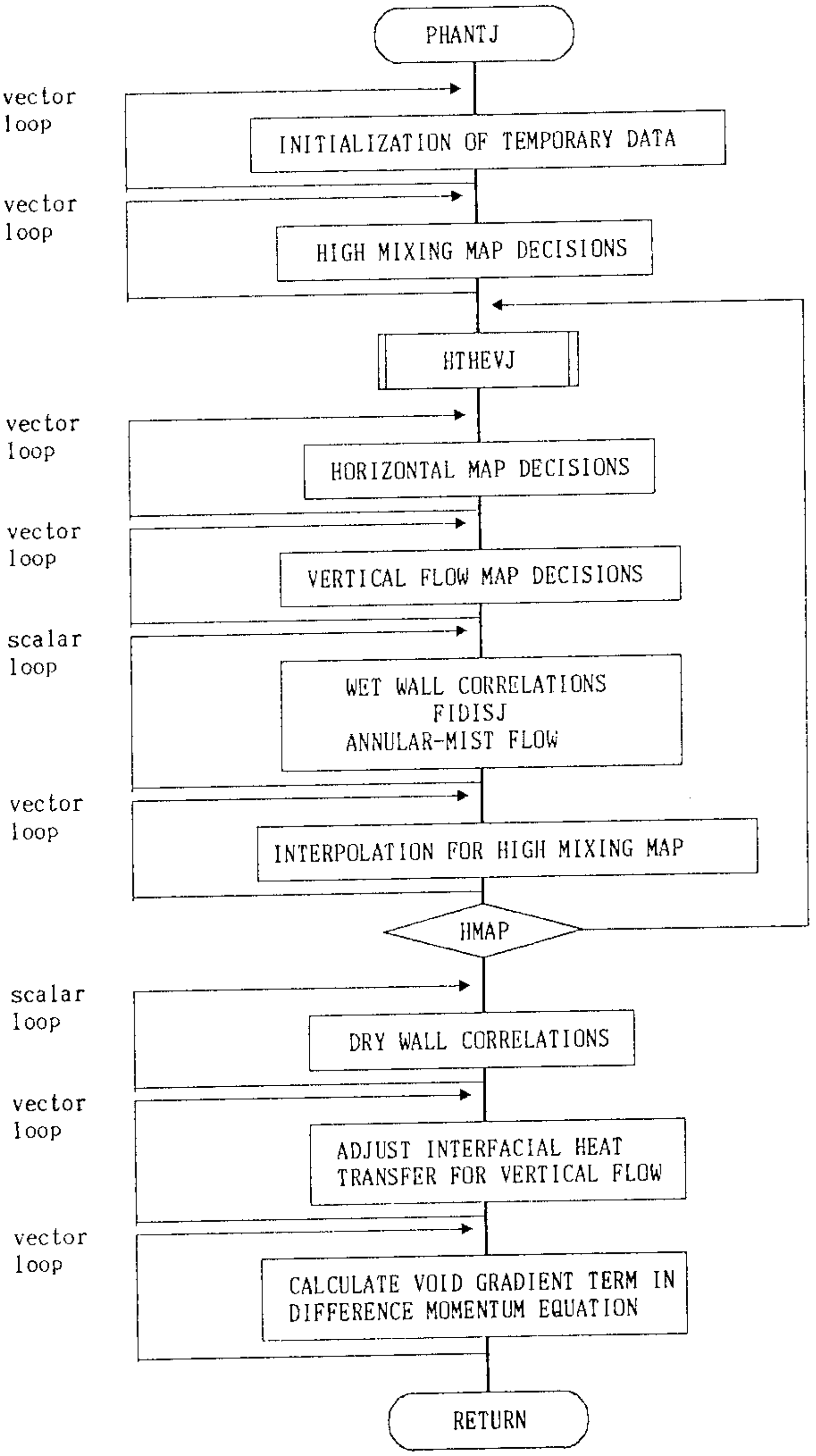


Fig. 3.5.1 Flow diagram of subroutine PHANTJ in vectorized version

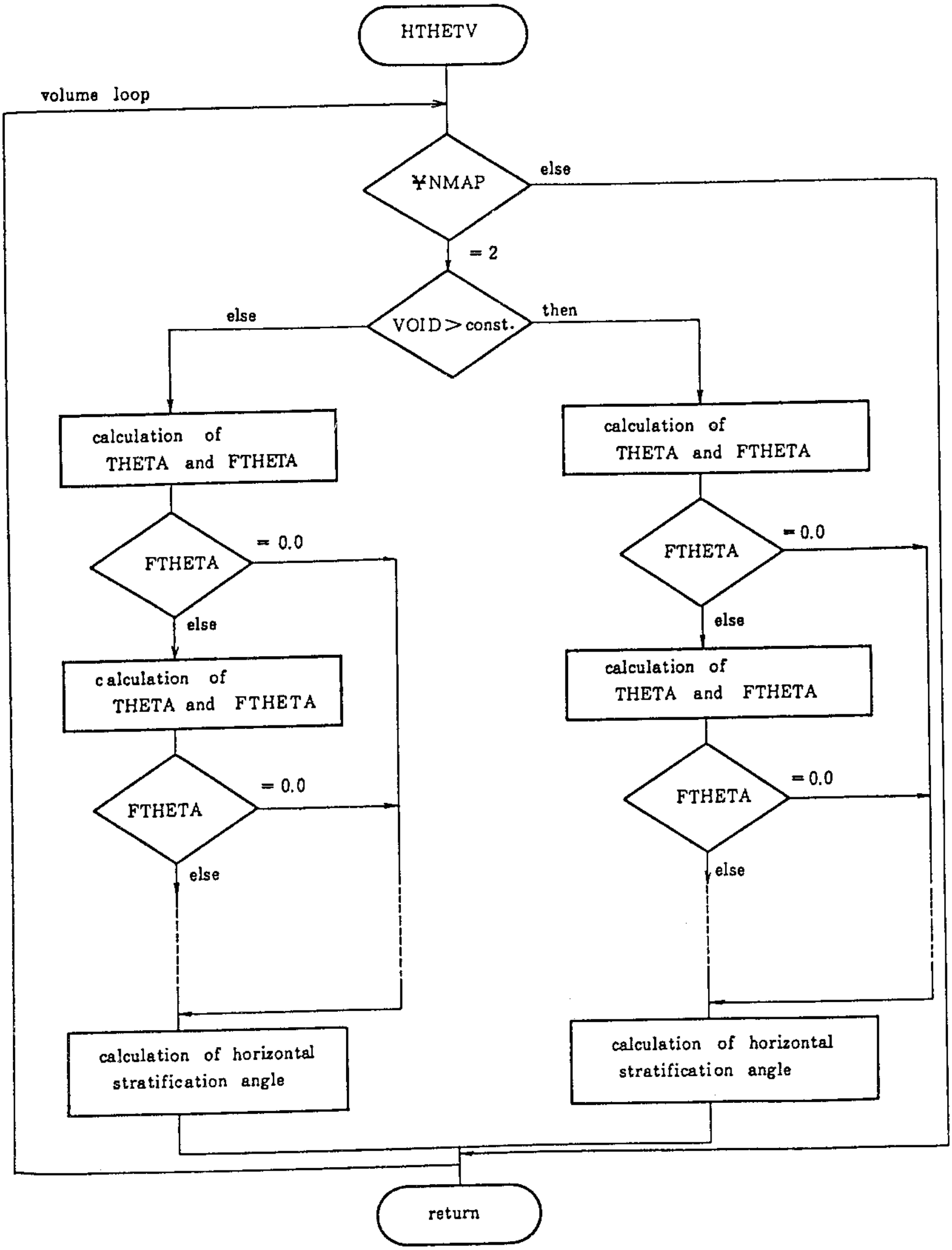


Fig. 3.5.2 Flow diagram of subroutine HTHETA in vectorized version

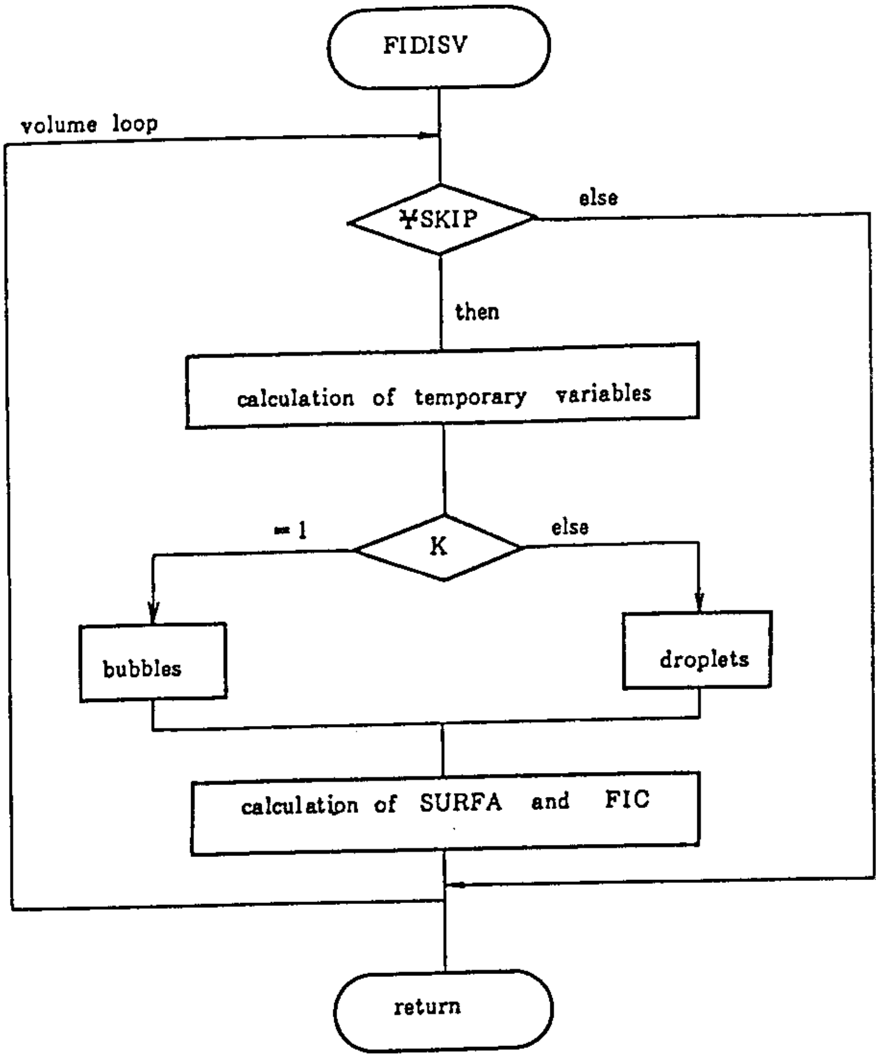


Fig. 3.5.3 Flow diagram of subroutine FIDIS in vectorized version

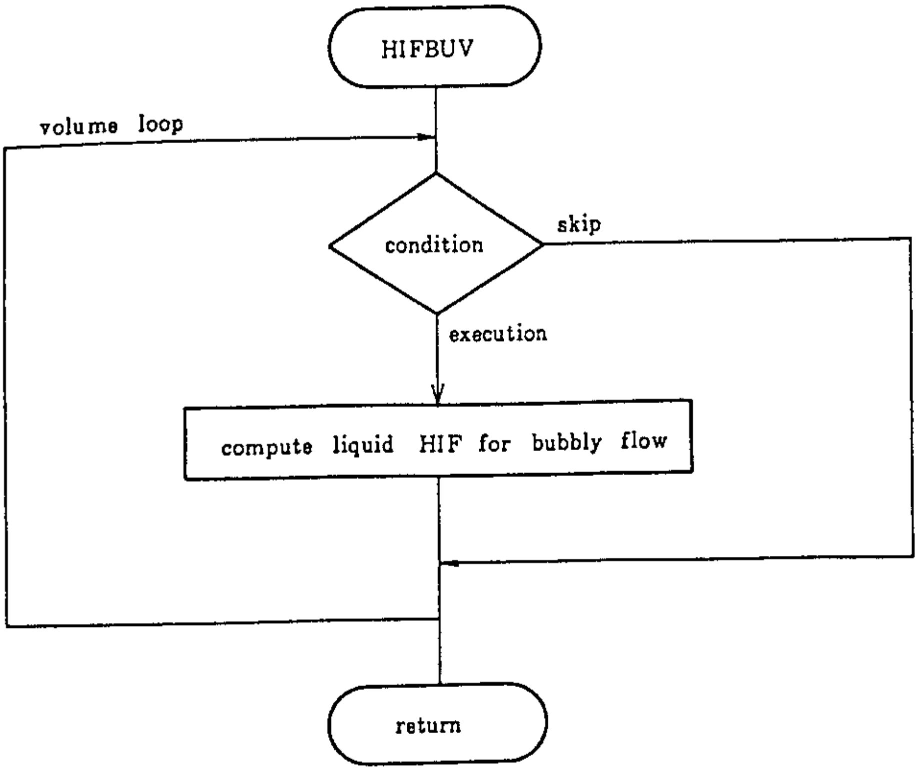


Fig. 3.5.4 Flow diagram of subroutine HIFBUS in vectorized version

3.6 Vectorization method for state relationship calculation

The six equation model with an additional equation for noncondensible gas component has the five independent variables; pressure, void fraction, vaper and liquid internal energies, noncondensible vaper phase mass fraction. All the remaining thermohydraulic variables; temperature, densities, partial pressure, quality, etc. are expressed as function of these five independent properties.

The state relationship calculation evaluates the dependent variables and their derivatives from equations of state *4.

The main program structure was doubly-nested loops of components and volumes. But in the JAERI vectorized version, the double loops are reformed into a single loop over the total volumes. The calculation mostly depends on the fluid state, that is, different subroutines are called depending on the fluid state. These lower-level subroutines can be vectorized on the volume-loop which is transferred from parent routine. For this purpose, list vector for each fluid state was alrealy provided by INEL for most part of the program. We only turn up the DO-loop structure to vectorize on the single loop for all the volumes.

The vectorized subroutines are summarized in Table 3.6.1.
The vectorization method for each subroutine is as follows:

(1) Subroutine STATE

All the volume indices are gathered and reserved as a list vector NVOLS¥ at the first time-step. This list vector is used in the slave subroutines lower than the STATEP. The volume indices which compose a coolant loop system (preliminary and secondary loops, etc) are invariant throughout the calculation. Hence, list vectors are generated only once at the first time-step to keep the volume indices composing of systems. These list vectors are used in later time-steps to compute the system mass errors.

(2) STATEP

Flow diagram of subroutine STATEP in the original code is shown in Fig. 3.6.1. The outer component loop and inner volume loop are reformed into a single loop for all the volumes. Two list vectors were already created in the original code; one is for the volumes without air and another is for the volumes with air. Since the number of volumes with air is very small, a scalar loop is applied. But the volume-loop without air is vectorized. The DO-loop is divided into several blocks to isolate the subroutine calls to STH2X6 (= STH2XF). These subroutine calls are vectorized in the subroutine STH2X6 by transferring the DO-loop from STATEP to STH2X6.

(3) STH2X6

Flow diagram of subroutine STH2X6 in the original code is shown in Fig. 3.6.2. The volume-loops transferred from STATEP were originally divided into several groups by the fluid states as shown in Fig. 3.6.2. Each calculational block is vectorized using the related list vector.

Table search of the steam table is newly vectorized. The original DO-loop structure could not be vectorized because of iterative GOTO loops which were used to table look-up. In the vectorized version, the GOTO loops were moved outside as the outer loop and, on the contrary, table search over volumes is vectorized. If the volume is found, it should be deleted at the next search. Accordingly, the list vector must be regenerated in every search time. The method is illustrated in Fig. 3.6.3. In order to clearfy the method, new and old program flows are compared in Fig. 3.6.4.

(4) SVH2X2

Speedup of table search is not expected in this routine because of short vector length and then the remaining part of programs are vectorized.

(5) VISCVL, VISCGV, THCNGV, THCNFV

In the parent subroutine STATEP, list vectors were generated in each time-step to gather the volume indices according to the fluid states. After then, the fluid state calculations, which may be carried out by calling one of above subroutines, are vectorized using list vectors.

(6) SURTNV

This subroutine is vectorized for all the volumes.

Table 3.6.1 Vectorized subroutines for state relationship calculation

Subroutine names Original	 Vectorized	Contents	Vectorized index
STATE	STATE	Controls the evaluation of the equation of state for all components.	Volumes
STATEP	STATEV	Computes equation of state and derivatives for time advanced volumes.	Volumes
STH2X6	STH2X6	Computes water thermohydraulic properties as function of pressure and internal energy.	Volumes
SVH2X2	STH2X2	Computes water thermohydraulic properties as a function of pressure and quality.	Volumes
VISCVL	VISCVL	Calculate water liquid viscosity	Volumes
VISCVG	VISCVG	Calculate water vapor viscosity	Volumes
THCNGV	THCNGV	Computes thermal conductivities of saturated or subcooled liquid.	Volumes
THCNFV	THCNFV	Computes thermal conductivities of saturated or superheated vapor.	Volumes
SURTNV	SURTNV	Determins H_2O surface tension from the fluid temperature.	Volumes

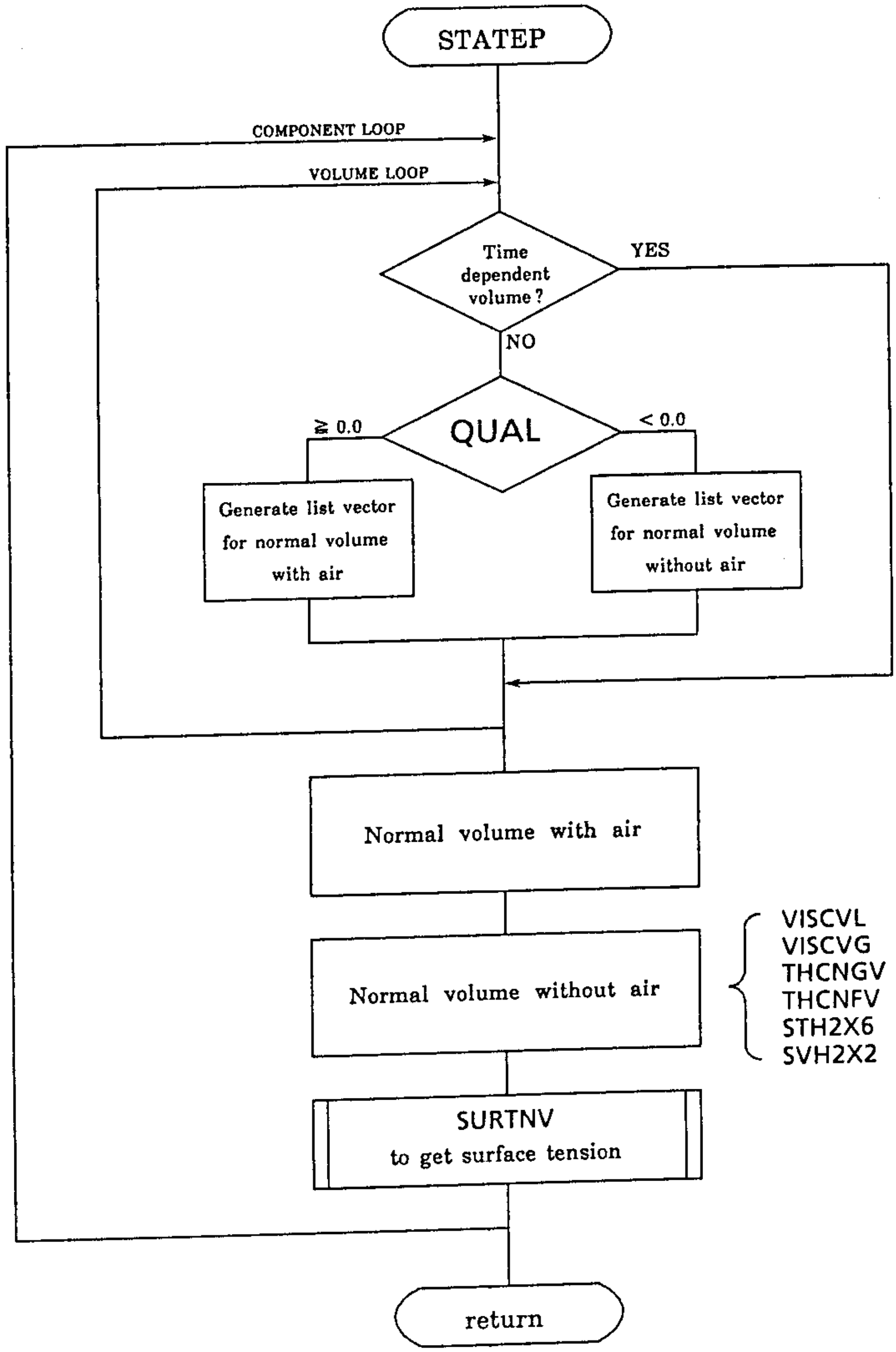


Fig. 3.6.1 Flow diagram of subroutine STATEP in the original code

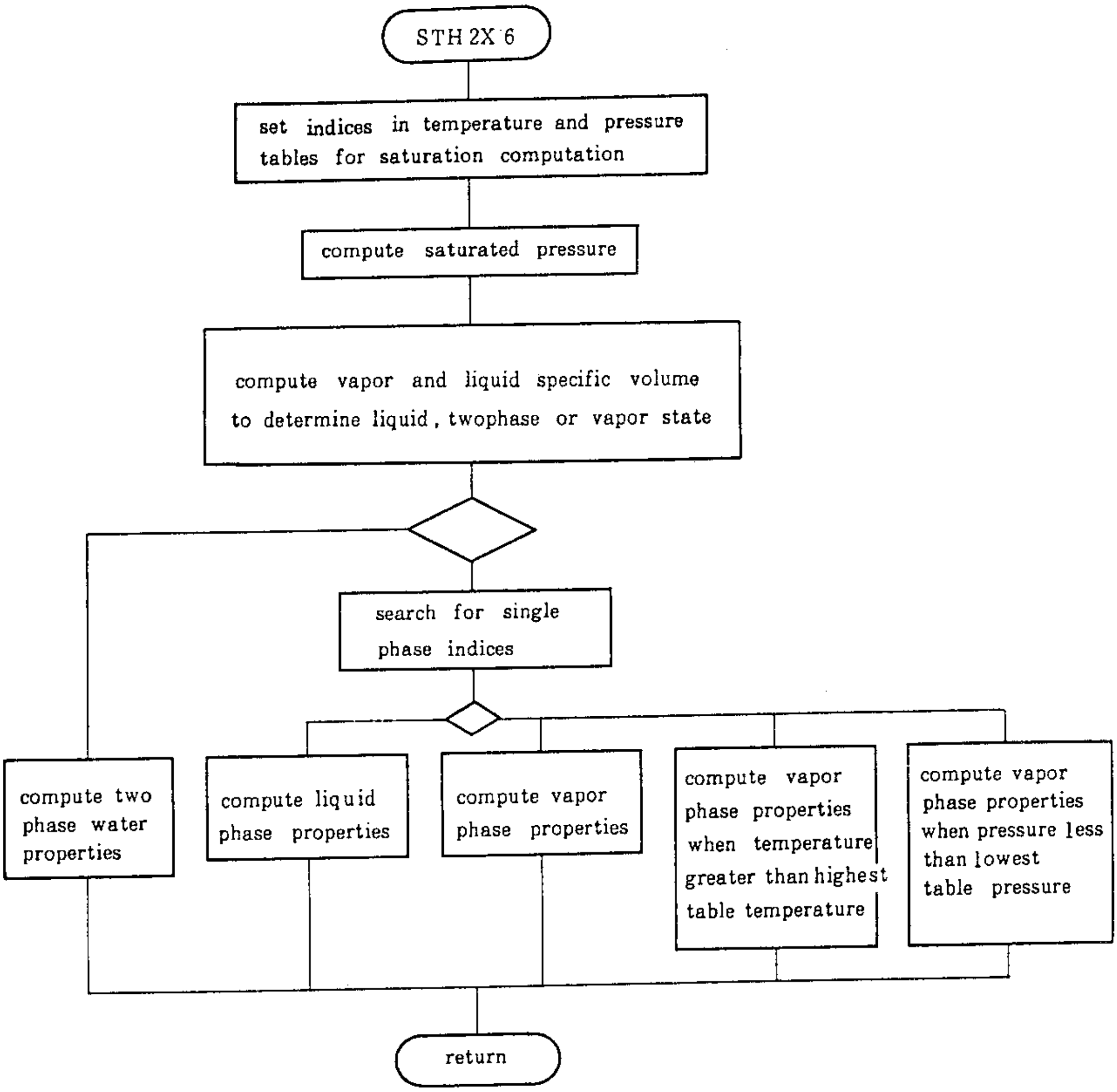


Fig. 3.6.2 Flow diagram of subroutine STH2X6 in the original code

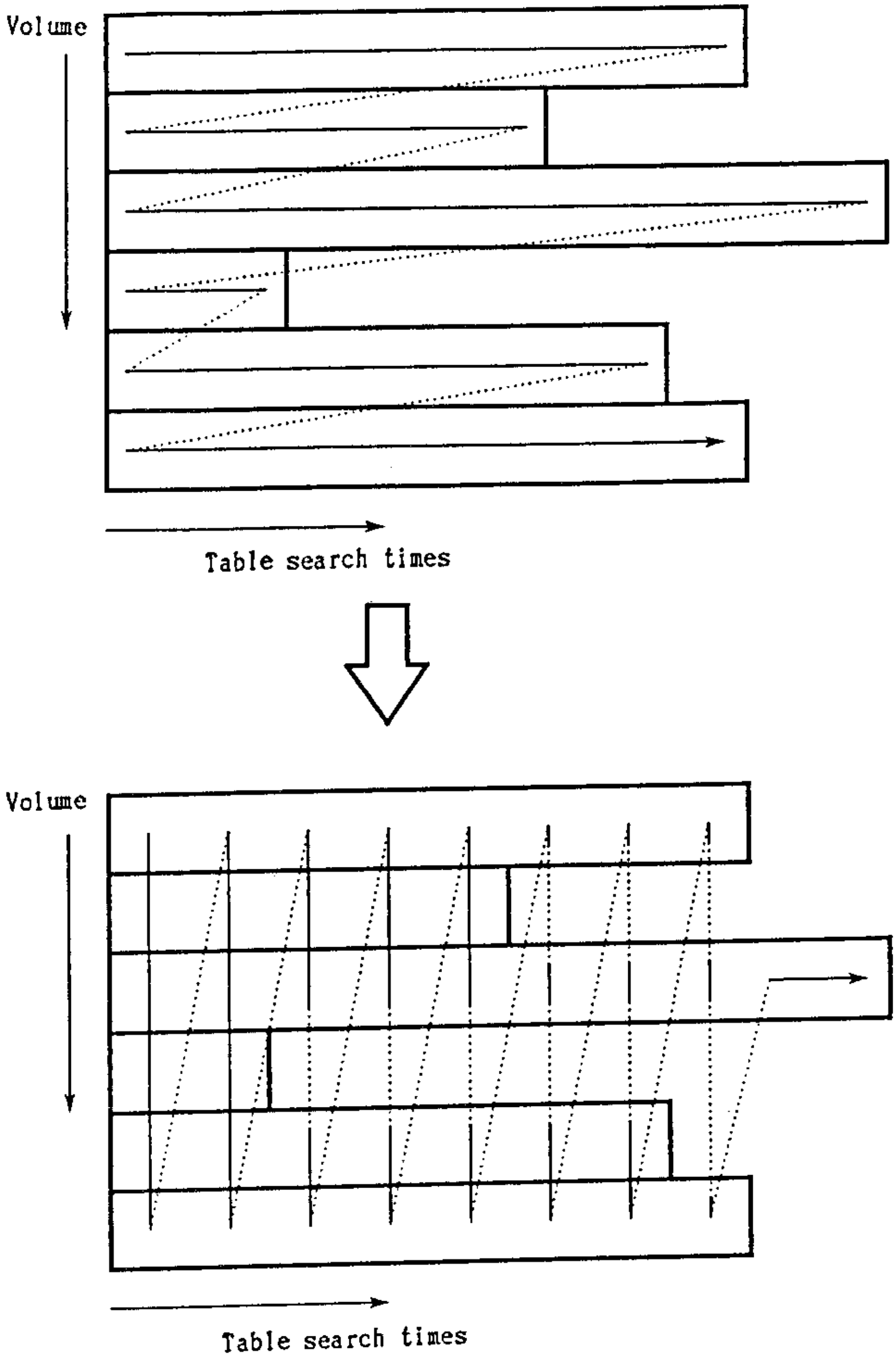


Fig. 3.6.3 Restructure for table look-up of steam table

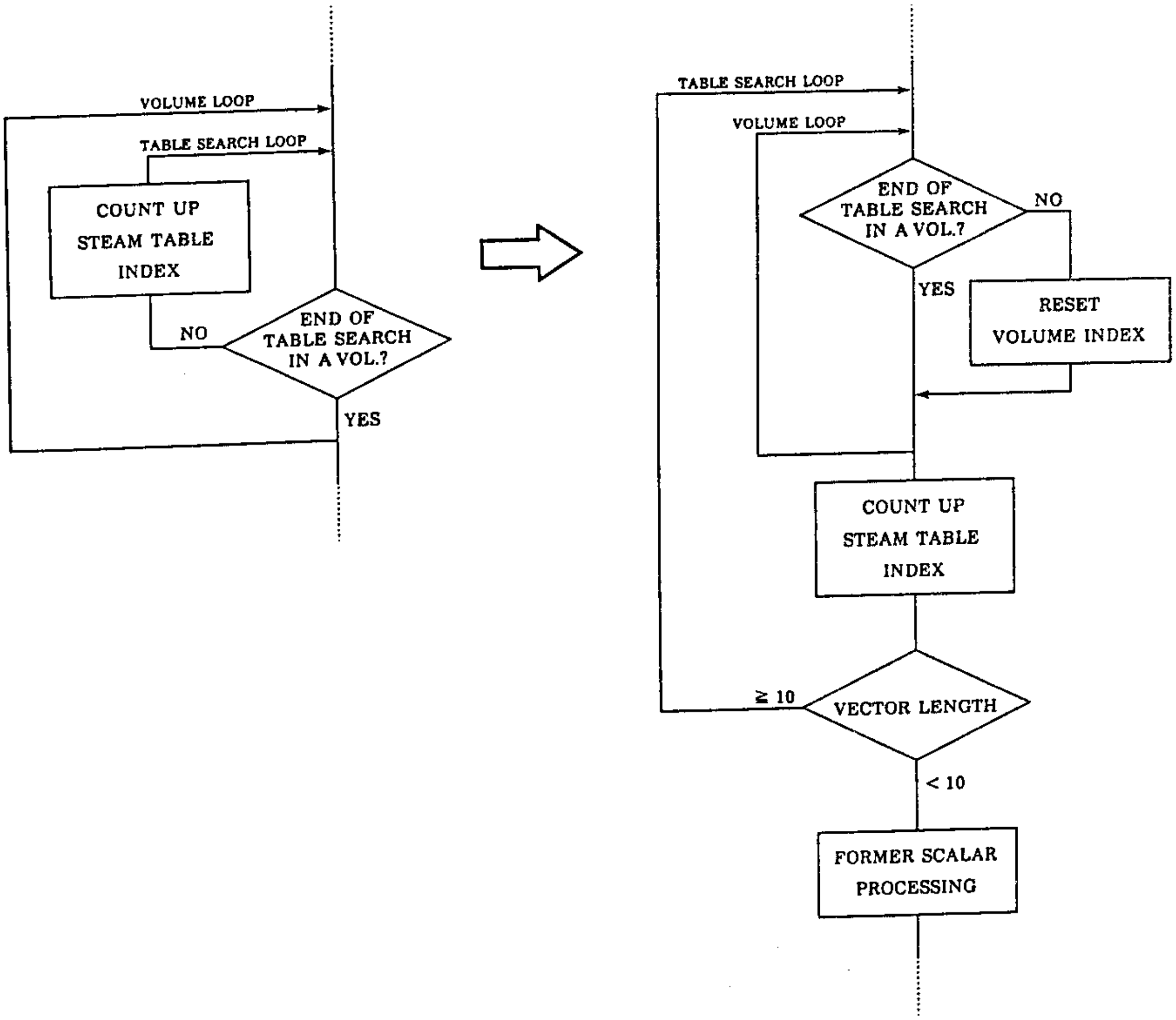


Fig. 3.6.4 Change of program flow for vectorized table look-up

3.7 Performance of vectorization

By these vectorization efforts, the vectorization ratio 78% was achieved. The vectorized RELAP5/MOD3 runs in the vector mode 2.8 times faster than in the scalar mode on FACOM VP2600 for the type TYPPWR problem (100 sec. small break loss of coolant flow accident). These performance numbers are obtained from the following measured data:

SS	Nonvectorized FACOM version in scalar mode	37.17 sec.
VS	Vectorized FACOM version in scalar mode	38.23 sec.
VV	Vectorized FACOM version in vector mode	13.88 sec. (VU time 5.24)

Here VU time is the time spent on the vector unit of VP2600 (see Fig. 1.1.1). Vectorization ratio (V) is derived using Amdahl's law [*10] as folows:

$$P = 1 / (1 - V + V / \alpha) \tag{1}$$

where P is the speedup ratio compared with the scalar mode and interprated as

$$P = (\text{time on VS})/(\text{time on VV}) = 2.8, \tag{2}$$

and α is acceleration ratio in vector mode.

The ratio

$$VU = (\text{VU time})/(\text{time on VV}) = 0.378 \tag{3}$$

is as called as VU ratio. Then the VU means in the Amdahl's law as

$$VU = V/\alpha/(1-V + V/\alpha), \tag{4}$$

it follows:

$$V/\alpha = (1- V)/(1-VU) \ast VU. \tag{5}$$

From eqs.(4) and (5)

$$P = VU/(V/\alpha) = (1-VU)/(1-V). \tag{6}$$

Consequently, we have

$$V = 1 - (1 - VU)/P = 1- (1- 0.378) \ast 0.363 = 0.78 \tag{7}$$

$$\alpha = P \ast V/VU = V \ast (1-VU)/VU \ast (1-V) = 5.7 \tag{8}$$

Tables 3.7.1, 3.7.2 and 3.7.3 represents the timming data for each calculational block of SS, VS, and VV, respectively, for the TYPPWR problem. Fig. 3.7.1 shows the program flow of the vectorized RELAP5/MOD3 accompanied with the time percents and speedup ratio to vector mode to scalar mode calculations for the same problem.

Table 3.7.4 is the summary of computing time of the vectorized RELAP5/MOD3, in scalar and vector modes. Here the vector to scalar speedup ratios in the vectorized version are presented in parentheses. The discrepancy of time step numbers among calculations is caused from the difference of arithmetic operation order between nonvectorized and vectorized codes, and between vector and scalar modes such as summation.

From above performance tables and figure, we can see that the vectorized RELAP5/MOD3 runs 7.2 times faster than the real time (100sec.) on the FACOM VP2600. In the TYPPWR problem, the volume nubmer 139, junction number 142, heat structure number 83, and heat mesh point or interval number 393 are calculated. But in the practical calculation at JAERI these numbers becomes larger and then higher speedup ratio is expected.

Table 3.7.1 CPU time table of nonvectorized code in scalar mode (SS)

```
TRANSIENT TIMING INFORMATION
SUBROUTINE     ENTRIES   TIME(SEC)     FRACTION
    DTSTEP         687      1.3126       0.0364
  STATE(R)          59      0.4602       0.0128
      TRIP         627      0.0188       0.0005
     HTADV         686      2.3939       0.0664
    HTCOND       56938      2.3350       0.0648
  STATE(T)         686      0.0775       0.0022
     JPROP        2803      0.5986       0.0166
    VOLVEL         686      0.7893       0.0219
    PACKER         686      0.1656       0.0046
    PHAINT         686      7.1020       0.1970
    HZFLOW        2744      0.3647       0.0101
    FWDRAG         686      1.7796       0.0494
     HLOSS         686      0.4560       0.0127
    VEXPLT         686      2.4895       0.0691
    VLVELA         627      0.5548       0.0154
    PRESEQ        1268      3.9568       0.1098
    SYSSOL        1268      1.6744       0.0465
     VFINL         686      0.7108       0.0197
    EQFINL         686      2.3719       0.0658
  STATE(A)         636      5.2576       0.1459
      RKIN         627      0.0901       0.0025
    CONVAR         627      0.0595       0.0017
    SBTRAN           0      0.0          0.0
    SBNTAC           0      0.0          0.0
    JCHOKE         686      1.0001       0.0277
    FUELAN           0      0.0          0.0
     CYLIN           0      0.0          0.0
     SLABC           0      0.0          0.0
     FPRTR           0      0.0          0.0
     TSINT           0      0.0          0.0
     ZSOLV           0      0.0          0.0
     DERV1           0      0.0          0.0
     DERV3           0      0.0          0.0
      FMIX           0      0.0          0.0
     VALVE         686      0.0245       0.0007
TOTAL                      36.0436
NO PLOTS MADE
```

Table 3.7.2 CPU time table of vectorized code in scalar mode (SV)

```
TRANSIENT TIMING INFORMATION
SUBROUTINE     ENTRIES    TIME(SEC)      FRACTION
     DTSTEP        655       1.3034        0.0351
   STATE(R)         44       0.3818        0.0103
       TRIP        610       0.0191        0.0005
      HTADV        654       2.4596        0.0662
     HTCOND        654       1.9650        0.0529
   STATE(T)        654       0.0731        0.0020
      JPROP       2660       0.5800        0.0156
     VOLVEL        654       0.7541        0.0203
     PACKER        654       0.1721        0.0046
     PHAINT        654       8.6012        0.2314
     HZFLOW       2616       0.3383        0.0091
     FWDRAG        654       1.6953        0.0456
      HLOSS        654       0.4646        0.0125
     VEXPLT        654       2.3750        0.0639
     VLVELA        610       0.5414        0.0146
     PRESEQ       1208       3.9361        0.1059
     SYSSOL       1208       1.5945        0.0429
      VFINL        654       0.6801        0.0183
     EQFINL        654       2.2787        0.0613
   STATE(A)        617       5.6875        0.1530
       RKIN        610       0.0882        0.0024
     CONVAR        610       0.0534        0.0014
     SBTRAN          0       0.0           0.0
     SBNTAC          0       0.0           0.0
     JCHOKE        654       1.0942        0.0294
     FUELAN          0       0.0           0.0
      CYLIN          0       0.0           0.0
      SLABC          0       0.0           0.0
      FPRTR          0       0.0           0.0
      TSINT          0       0.0           0.0
      ZSOLV          0       0.0           0.0
      DERV1          0       0.0           0.0
      DERV3          0       0.0           0.0
       FMIX          0       0.0           0.0
      VALVE        654       0.0270        0.0007
 TOTAL                      37.1637
 NO PLOTS MADE
```

Table 3.7.3 CPU time table of vectorized code in vector mode (VV)

```
TRANSIENT TIMING INFORMATION
SUBROUTINE      ENTRIES   TIME(SEC)     FRACTION
    DTSTEP          656      0.9828      0.0745
  STATE(R)           44      0.1173      0.0089
      TRIP          611      0.0174      0.0013
     HTADV          655      0.4835      0.0367
    HTCOND          655      0.7107      0.0539
  STATE(T)          655      0.0733      0.0056
     JPROP         2664      0.1922      0.0146
    VOLVEL          655      0.1140      0.0086
    PACKER          655      0.0741      0.0056
    PHAINT          655      3.3333      0.2527
    HZFLOW         2620      0.1091      0.0083
    FWDRAG          655      0.2532      0.0192
     HLOSS          655      0.3480      0.0264
    VEXPLT          655      0.3756      0.0285
    VLVELA          611      0.1088      0.0082
    PRESEQ         1246      0.8531      0.0647
    SYSSOL         1246      1.6057      0.1217
     VFINL          655      0.1230      0.0093
    EQFINL          655      0.4226      0.0320
  STATE(A)          618      1.9112      0.1449
      RKIN          611      0.0865      0.0066
    CONVAR          611      0.0384      0.0029
    SBTRAN            0      0.0         0.0
    SBNTAC            0      0.0         0.0
    JCHOKE          655      0.8327      0.0631
    FUELAN            0      0.0         0.0
     CYLIN            0      0.0         0.0
     SLABC            0      0.0         0.0
     FPRTR            0      0.0         0.0
     TSINT            0      0.0         0.0
     ZSOLV            0      0.0         0.0
     DERV1            0      0.0         0.0
     DERV3            0      0.0         0.0
      FMIX            0      0.0         0.0
     VALVE          655      0.0237      0.0018
TOTAL                       13.1902
NO PLOTS MADE
```

Table 3.7.4 Timming comparison of vectorized RELAP5/MOD3 for TYPPWR 100 sec. small break LOCA

Version	Time-steps	Heat	Hydro	State	Others	Total
Nonvectorized scalar (SS)	687	4.7 (0.94)	23.1 (1.04)	5.3 (1.07)	2.8 (1.04)	36.0 (1.03)
Vectorized scalar (VS)	655	4.4 (1.0)	24.1 (1.0)	5.7 (1.0)	2.9 (1.0)	37.1 (1.0)
Vectorized Vector (VV)	656	1.19 (3.7)	7.98 (3.2)	1.91 (3.0)	2.10 (1.4)	13.19 (2.8)

Calculated problem: 139 volumes, 142 junctions, 83 heat structures, and 393 heat mesh points or intervals. Vectorization ratio 78%.

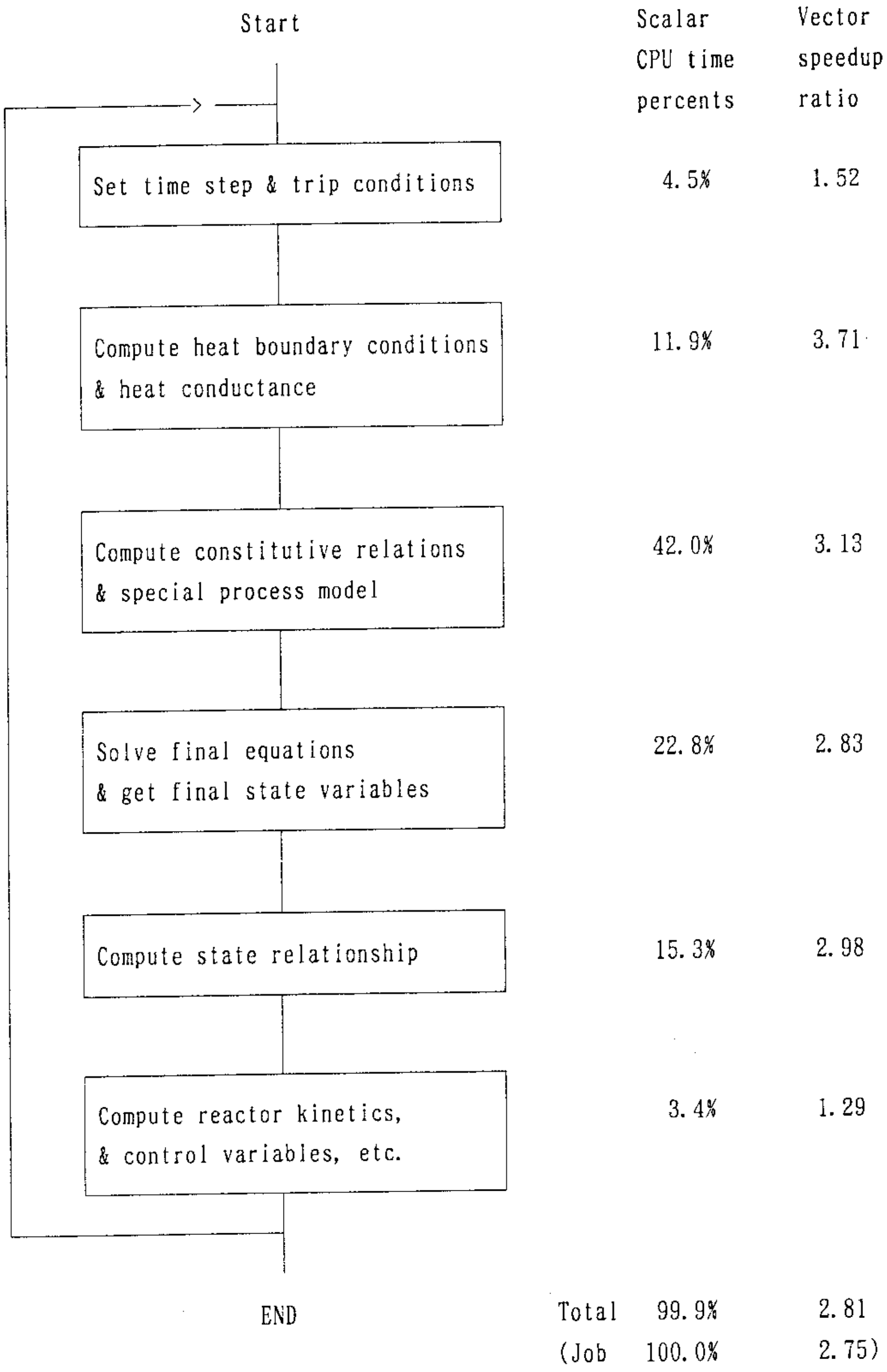


Fig. 3.7.1 Program flow, CPU time ratio in scalar mode, and speedup ratio of vector mode to scalar mode in vectorized RELAP5/MOD3

3.8 Verification of the computed results

In order to validate the vectorized RELAP5/MOD3, computed results are compared between INEL and JAERI codes and between nonvectorized and vectorized codes.

The computed results that were packed in the INEL transmittal tape are not always useful for verifying the JAERI RELAP5/MOD3 since out of seven results of problems only the EDHTRK problem was run with RELAP5/MOD3 on CRAY X-MP but the rest six problems were run with the preliminary code RELAP5/MOD2.5. Therefore, the results of EDHTRK have been compared between INEL CRAY and JAERI FACOM versions. Fig. 3.8.1 and Fig. 3.8.2 shows the comparison of water temperature and pressure at the outlet of the pipe, respectively. Here the nonvectorized scalar mode calculation (SS) is applied as the JAERI code. The computed results have a good agreement between two calculations.

Figs. 3.8.3 and 3.8.4 show the comparison between nonvectorized scalar (SS) and vectorized vector (VV) mode calculations of RELAP5/MOD3 of JAERI for the TYPPWR problem. In Fig. 3.8.2 and Fig. 3.8.3, the water temperature and pressure at the inlet of hotleg of regular side of the typical PWR is pictured, respectively. Agreement of the water temperatures is two figures. The discrepancy is caused from the difference of arithmetic operation order between scalar and vector operations such as summation, mathematical function evaluation (SIN, COS, LOG ·····), division, etc.*5 The computed results obtained from vectorized scalar mode calculation (VS) have the similar agreement with that of SS or VV.

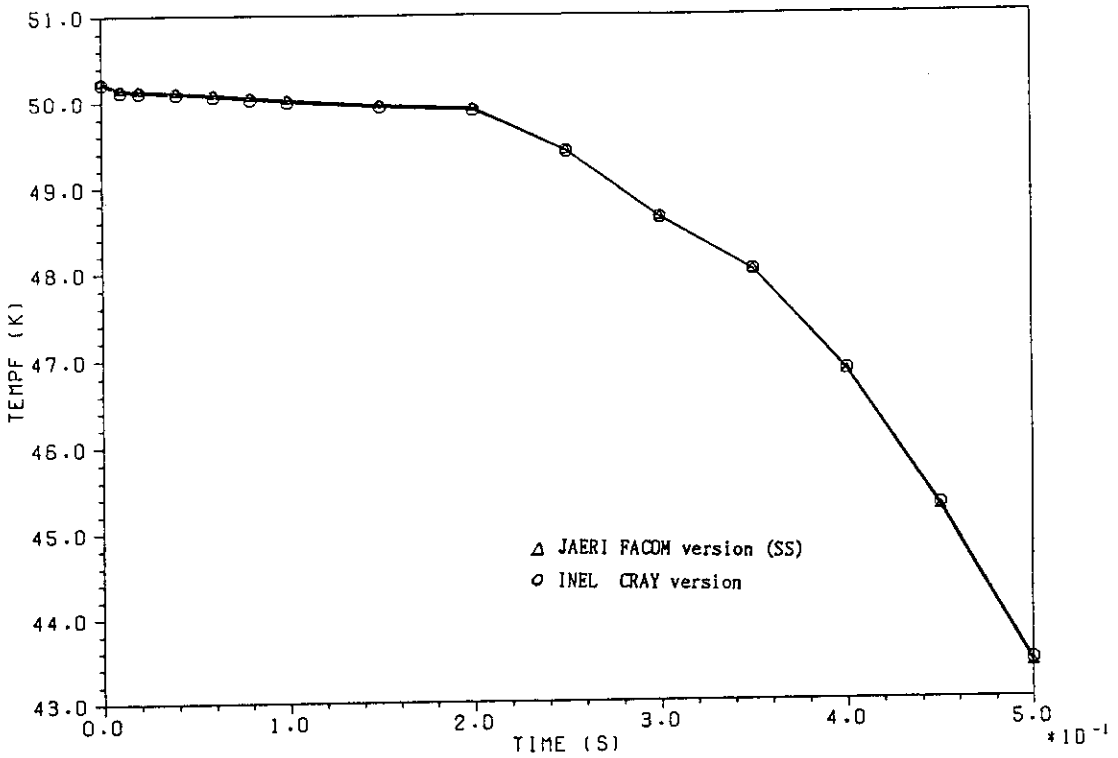


Fig. 3.8.1 Comparison of water temperatures between the INEL CRAY and JAERI FACOM versions for the problem EDHTRK

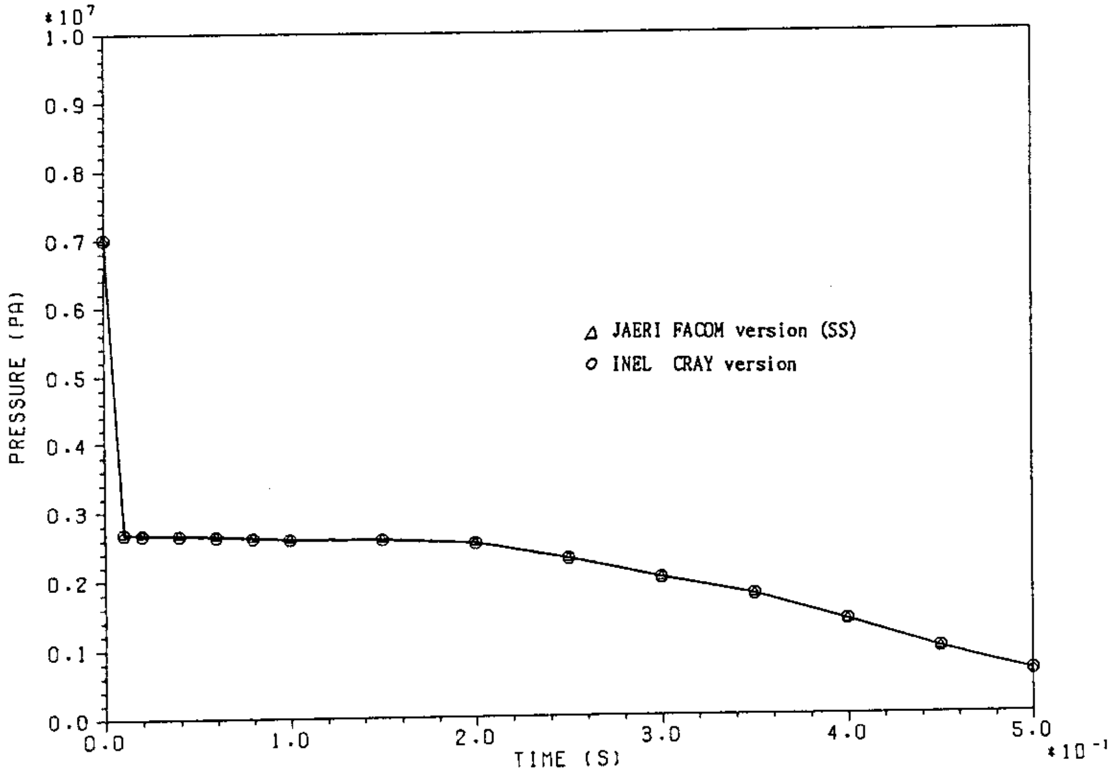


Fig. 3.8.2 Comparison of pressures between the INEL CRAY and JAERI FACOM versions for the problem EDHTRK

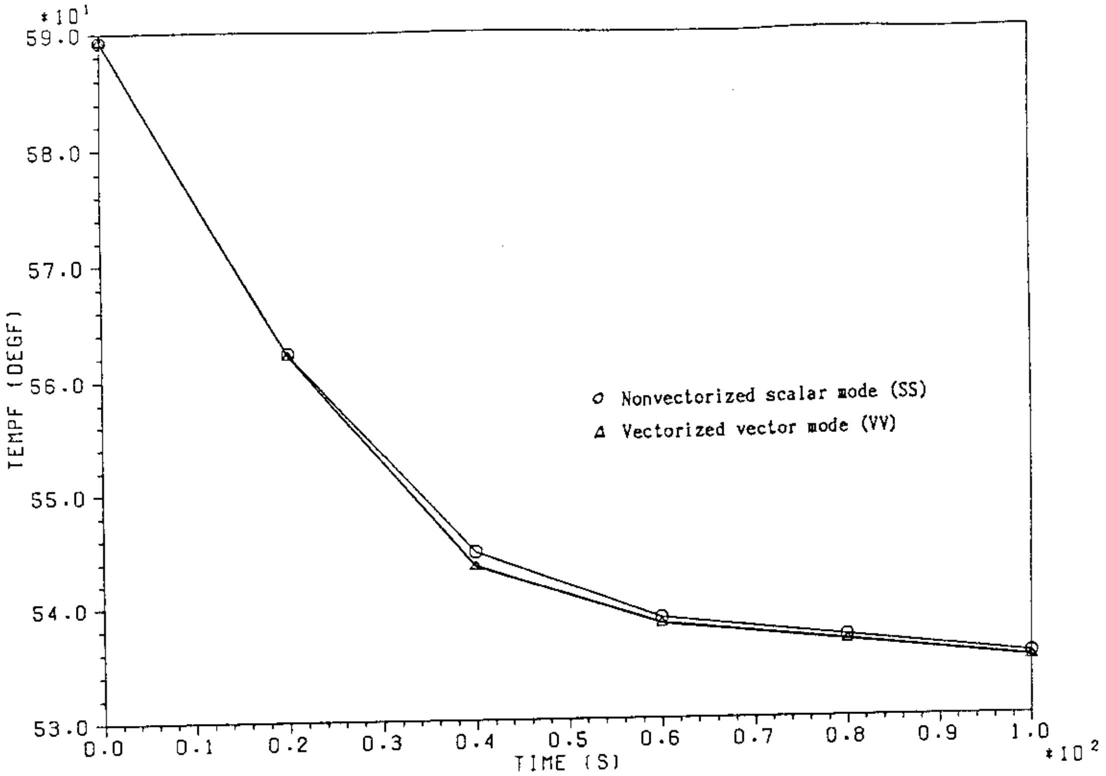


Fig. 3.8.3 Comparison of water temperatures between the JAERI nonvectorized scalar (SS) and JAERI vectorized vector (VV) mode calculations for the problem TYPPWR

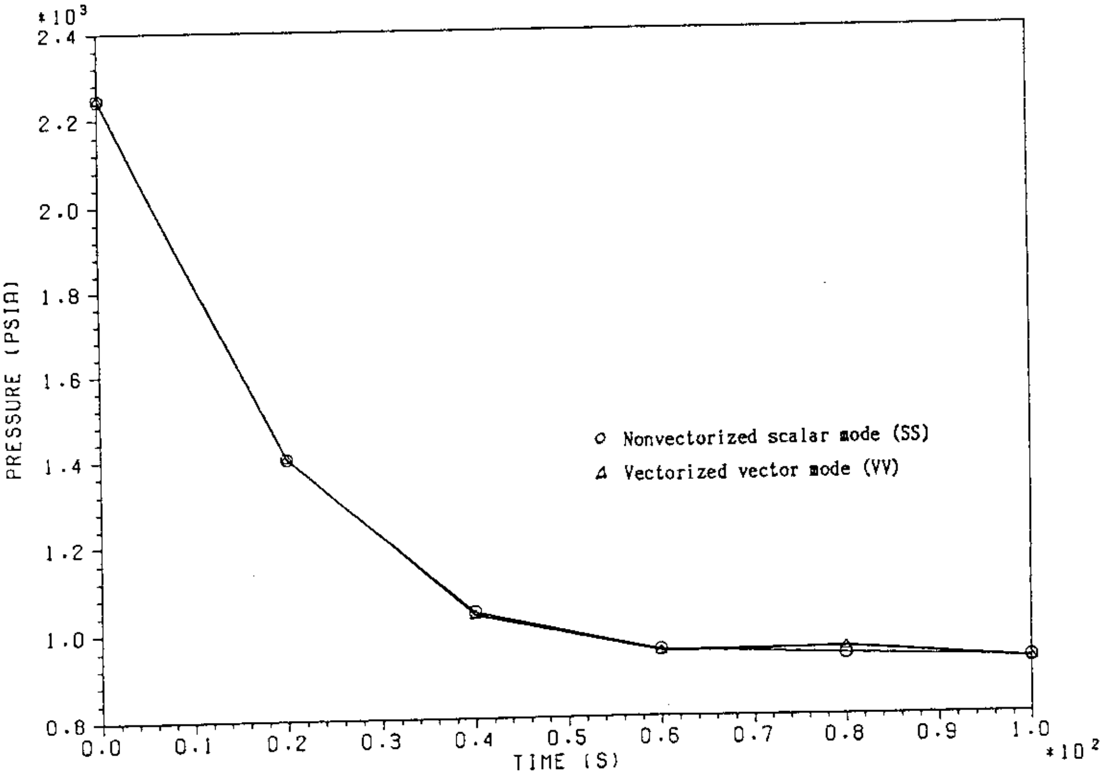


Fig. 3.8.4 Comparison of pressures between the JAERI nonvectorized scalar (SS) and JAERI vectorized vector (VV) mode calculations for the problem TYPPWR

4. Concluding Remarks

The conversion from INEL IBM version to FACOM version was accomplished. The FACOM version has been vectorized for using FACOM VP2600 supercomputer. The vectorization ratio 78% is achieved and the obtained speedup ratio is 2.8 times compared with scalar calculation for the typical PWR problem (TYPPWR) which was provided by INEL. This ratio will increase for the practical problems at JAERI because longer vector length is expected. The computed results of the FACOM vectorized RELAP5/MOD3 have a good agreement with that of INEL CRAY version. The computed results also have a good agreement between scalar and vector mode calculations of the FACOM vectorized RELAP5/MOD3.

The manpower used for implementing FACOM nonvectorized version was three months by one person and another three months by two persons were required for the vectorization.

This report is firstly aimed at the information exchange between USNRC and JAERI under the ROSA-IV project to show the methods applied at JAERI and to help other IBM or FACOM users of the world to implement own version.

We also hope that the descriptions in this report are useful for the future code conversion and vectorization at JAERI.

Acknowledgement

Authors wish to thank Mr.T. Watanabe and other members of Thermohydraulic Safety Engineering Laboratory of JAERI for their helpful comments and suggestions on the verfication of computed results.

4. Concluding Remarks

The conversion from INEL IBM version to FACOM version was accomplished. The FACOM version has been vectorized for using FACOM VP2600 supercomputer. The vectorization ratio 78% is achieved and the obtained speedup ratio is 2.8 times compared with scalar calculation for the typical PWR problem (TYPPWR) which was provided by INEL. This ratio will increase for the practical problems at JAERI because longer vector length is expected. The computed results of the FACOM vectorized RELAP5/MOD3 have a good agreement with that of INEL CRAY version. The computed results also have a good agreement between scalar and vector mode calculations of the FACOM vectorized RELAP5/MOD3.

The manpower used for implementing FACOM nonvectorized version was three months by one person and another three months by two persons were required for the vectorization.

This report is firstly aimed at the information exchange between USNRC and JAERI under the ROSA-IV project to show the methods applied at JAERI and to help other IBM or FACOM users of the world to implement own version.

We also hope that the descriptions in this report are useful for the future code conversion and vectorization at JAERI.

Acknowledgement

Authors wish to thank Mr.T. Watanabe and other members of Thermohydraulic Safety Engineering Laboratory of JAERI for their helpful comments and suggestions on the verfication of computed results.

References

1) Wagner, R.J. and Singer, G.L., Instllation Notes for Integrated Code RELAP5/MOD2.5 and SCDAP-RELAP/MOD2, dated June 16, 1989.

2) Kukita, Y. et al., Developmental Assessment of RELAP5/MOD3 Code against ROSA-IV/TPTF Horizontal Two Phase Flow Experiments, JAERI-M 90-053 (1990).

3) Wagner, R. J. et al. (Private communication) (1987).

4) Ranson, V.H. et al., RELAP5/MOD2 Code Manual, Volume 1 and Volume 2, NUREG/CR 4312, EGG-2396 (1985).

5) Ishiguro, M., et al., Vectorization of the Light Water Reactor Transient Analysis Code RELAP5, Nuclear Science & Engineering, Vol. 92 (1985).

6) Ishiguro, M., Vector and Parallel Processing for Nuclear Reactor Transient Analysis Code RELAP5, Proc. of Supercomputing' 88 (1988).

7) FUJITSU, VP-2000 series Supercomputers (1990).

8) Makowitz, H., The CERBERUS Code: Experiments with Parallel Processing Using RELAP5/MOD3, Proc. of the ANS Winter Meeting (1989)

9) Shinizawa,N, Kondo, K., Wada, Y., and Ishiguro, M., Vectorization of the LWR Transient Analysis Code RELAP5/MOD2/CYCLE36, JAERI-M 86-019 (in Japanese) (1986)

10) Hockney, R.W. and Jesshope, C.R., Parallel Computers, Adam Hilger (1981).