SOFTWARE RELEASE NOTICE

01. SRN Number: PA-SRN-019

02. Project Title:
CANT2, Temperature Model for canisters at Yucca Mountain site, subroutine for TPA, CNWRA Version 1.1

Project No.
20-5702-723

03. SRN Title: CANT2

04. Originator/Requester:
Thomas J. Ratchford

Date:
03/09/94

05. Summary of Actions

- ■ Release of new code admitted to CM System (R.Janetzke)
- ☐ Release of modified code:
  - ☐ Enhancements made
  - ☐ Corrections made
- ■ Change of access code (Robert Baca)

06. Persons Authorized Access

Name	RO/RW	A/C/D

07. Element Manager Approval: Date:

08. Remarks:

A copy of the software package CANT2, CNWRA Ver. 1.1 was retained by the Principle Investigator for use in the CNWRA work center; therefore, a new release may not be necessary.

SOFTWARE SUMMARY FORM

01.Summary Date: 03/09/94	02. Summary prepared by(Name and Phone) T.J. Ratchford 522-3083	03. Summary Action: New
04. Software Date: 8/15/93	05. Short Title: CANT2	
06. Software Title: CANT2 - Temperature Model for Source Term.		07. Internal Software ID: NONE

08. Software Type:	09.Processing Mode:	10. APPLICATION AREA
☐ Automated Data System	☐ Interactive	A. General:
☐ Computer Program	☐ Batch	☐ Scientific/Engineering ☐ Auxiliary Analyses
■ Subroutine/Module	■ Combination	☐ Total System PA
		■ Subsystem PA ☐ Other
		b. Specific:

11. Submitting Organization and Address:	12. Technical Contact(s) and Phone:
CNWRA, SwRI, San Antonio, Texas	R. Janetzke, (210) 522-3318

13. Narrative:

The CANT2 code determines the temperature of individual canisters and average temperature in large zones at the Yucca Mountain site.

14. Computer Platform	15. Computer Operating System:	16. Programming Language(s):	17. Number of Source Program Statements:
CRAY/XMP	UNIX	FORTRAN	2,225 lines of code
18. Computer Memory Requirements:	**19. Tape Drives:**	**20. Disk/Drum Units:**	**21. Graphics:**
UNKNOWN	NONE	N/A	UNKNOWN

22. Other Operational Requirements

NONE

23. Software Availability:	24. Documentation Availability:
■ Available ☐ Limited ☐ In-House ONLY	■ Available ☐ Inadequate ☐ In-House ONLY

25. Submission Package Status:

Acceptance Criteria: Met ■ Not Met ☐ Software QA Assessment: Successful ■ Unsuccessful ☐

Code Custodian: T. J. Ratchford Date: 3/9/94

CNWRA Form TOP-4-1 (08/93)

CNWRA INFORMATION PROCESSING STANDARD SOFTWARE SUMMARY

01. Summary Date			02. Summary prepared by *(Name and phone)*	03. Summary action
Yr.	Mo.	Day	Janet Zke 3318	New ☐ Replacement ☒ Deletion ☐
92	12	17		
04. Software Date			**03. Software title**	Previous Internal Software ID
Yr.	Mo.	Day	CanT2	
92	11	20		

06. Short title	07. Internal Software ID
canT2	

08. Software type	09. Processing Mode	10. APPLICATION AREA	
		General	Specific
☐ Automated Data System	☐ Interactive	☐ Computer Systems Support/Utility ☐ Management/Business	
☐ Computer Program	☒ Batch	☒ Scientific/Engineering ☐ Process Control	
☐ Subroutine/Module	☐ Combination	☐ Bibliographic/Textual ☐ Other	

11. Submitting organization and address	12. Technical contact(s) and phone
NRC	Dick Codell

13. Narrative

New input file for CANT2.

canT2.in

14. Keywords

15. Computer manufacturer and model	16. Computer operating system	17. Programming language(s)	18. Number of source program statements
19. Computer memory requirements	**20. Tape drives**	**21. Disk/Drum units**	**22. Terminals**

23. Other operational requirements

24. Software availability			25. Documentation availability		
Available	Limited	In-house only	Available	Inadequate	In-house only
☐	☐	☒	☐	☐	☒

26. FOR SUBMITTING ORGANIZATION USE

CANT2 CRAY Listing

```
gemstone.4 ~/tpa/CANT2/VCS => ls -l
total 32
-rwxrwx---   1 tjr1     tjr1         1127 Jul  8  1993 s.Makefile*
-rwxrwx---   1 tjr1     tjr1         3653 Jul  8  1993 s.avtemp.F*
-rwxrwx---   1 tjr1     tjr1         6283 Jul  8  1993 s.cant2.F*
-rwxrwx---   1 tjr1     tjr1        20633 Jul  8  1993 s.cant2.cpp*
-rwxrwx---   1 tjr1     tjr1        20641 Jul  8  1993 s.cant2.fsourc*
-rwxrwx---   1 tjr1     tjr1         1680 Jul  8  1993 s.cant2.in*
-rwxrwx---   1 tjr1     tjr1         1730 Jul  8  1993 s.cant2.in.ori*
-rwxrwx---   1 tjr1     tjr1         2087 Jul  8  1993 s.cantemp.F*
-rwxrwx---   1 tjr1     tjr1          925 Jul  8  1993 s.driftpt.F*
-rwxrwx---   1 tjr1     tjr1          425 Jul  8  1993 s.erf.F*
-rwxrwx---   1 tjr1     tjr1          476 Jul  8  1993 s.presub.F*
-rwxrwx---   1 tjr1     tjr1          500 Jul  8  1993 s.presub2.F*
-rwxrwx---   1 tjr1     tjr1          941 Jul  8  1993 s.ran1.F*
-rwxrwx---   1 tjr1     tjr1         1338 Jul  8  1993 s.sethtab.F*
-rwxrwx---   1 tjr1     tjr1         2568 Jul  8  1993 s.sett.F*
-rwxrwx---   1 tjr1     tjr1         1642 Jul  8  1993 s.sett2.F*
-rwxrwx---   1 tjr1     tjr1          310 Jul  8  1993 s.subx.F*
-rwxrwx---   1 tjr1     tjr1          325 Jul  8  1993 s.subx2.F*
-rwxrwx---   1 tjr1     tjr1          453 Jul  8  1993 s.subz.F*
-rwxrwx---   1 tjr1     tjr1          284 Jul  8  1993 s.x.cant2.covr*
-rwxrwx---   1 tjr1     tjr1          704 Jul  8  1993 s.x.cant2.test*
gemstone.5 ~/tpa/CANT2/VCS =>
```

NNR 3/9/94

134.20.1.1 08:10:16

CANT2 Fortran Program
Static and Dynamic Analysis

June 28, 1993

Earl S. Marwil
John E. Tolli
Scientific Computing Unit
Idaho National Engineering Laboratory

1. Introduction

This analysis was performed on the Cray version of the software as provided by Southwest Research Institute (SwRI).

One sample problem was supplied along with the source code. The program was analyzed using the Craft (Cross Reference Analysis of Fortran) tool, FORWARN, the Fortran 77 analyzer, and PC-Metric. These tools provide static analysis, coverage analysis, and complexity analysis.

2. References

[1] N.H. Marshall and E.S. Marwil, Cross Reference Analysis of Fortran (CRAFT), EG&G-CATT-9198, EG&G Idaho, Inc., July 1991.
[2] Fortran 77 Analyzer User's Manual, National Bureau of Standards, NBS GCR 81-359, 1981
[3] FORWARN User's Guide, Quibus Enterprises, Inc., July 1991.
[4] PC-Metric User's Guide, SET Laboratories, Inc., 1987.

3. Functions

The CANT2 program contains 14 Fortran routines.

CANT2 has no alternate entry points.

4. Common Block Irregularities

There are no common blocks in the CANT2 program.

5. Interface Irregularities

Main program module "cant2" calls module "cantemp" with variable "tc" in argument position 19. This variable is dimensioned to 50 in "cant2", but is dimensioned to 100 in "cantemp".

6. Local Variable Irregularities

Parameter "nr" is assigned a value of 30 in "cant2" and "avtemp", but is assigned a value of 20 in "sethtab" and "sett".

Parameter "nr" is declared but unused in "sethtab".

Dummy argument "akm" is unused in "sethtab".

7. Fortran Extensions

All program modules contain some lower case alphabetic characters in their active Fortran.

Program modules "cant2", "cantemp", "driftpt", "presub2", "sethtab", "sett2", and "subz" contain entity names which are longer than 6 characters.

8. Optimization

The following table summarizes the performance data gathered from execution of the sample problem. Only those routines exercised by the sample problem are shown (see "Coverage Analysis" for a list of routines not exercised by the sample problem, i.e., coverage = 0%). The table lists all program modules in descending order according to CPU time. To optimize code execution time, emphasis should be placed on those modules which appear highest in the listing.

The performance data show that a high percentage of the overall execution time (81.729%) is spent in the first 4 routines listed (ERF, SUBX, SETT, SUBX2). This is due primarily to the following (applies to some or all of the 4 routines):

1) a low percentage of floating point operations which are performed in vector mode (%Vflops is small)

2) a high overhead factor for calls to the routines (IFact > 1)

3) a high level of memory conflicts (MC/MR > 1)

4) a high rate of instruction buffer fetches (IBFR > 1).

A detailed optimization analysis effort should focus on these 4 areas.

PERFORMANCE DATA FOR CANT2

ROUTINE NAME	Time	%ExTime	%AccumT	%Vflops	IFact	MC/MR	IBFR
ERF	3.327	25.744	25.744	0.00000	3088.42	4.133	0.236
SUBX	3.121	24.146	49.891	0.00051	485.72	3.110	0.979
SETT	2.973	23.005	72.895	0.00231	0.00	1.808	1.316
SUBX2	1.142	8.833	81.729	0.00000	120.97	3.037	1.042
SUBZ	0.939	7.265	88.994	0.00000	403.58	2.981	0.450
PRESUB	0.755	5.841	94.835	61.29032	502.00	1.802	0.560
SETT2	0.410	3.169	98.004	0.00528	0.00	2.004	1.095
DRIFTPT	0.116	0.894	98.897	99.02768	0.01	0.898	0.014
AVTEMP	0.079	0.613	99.511	69.65018	0.00	0.658	0.932
CANT2	0.053	0.408	99.919	46.25665	0.00	1.128	0.936
SETHTAB	0.003	0.026	99.945	0.78864	0.00	1.803	1.165
PRESUB2	0.003	0.024	99.968	63.33333	2.01	1.943	0.557
RAN1	0.003	0.022	99.990	0.00000	0.83	5.624	0.553
CANTEMP	0.001	0.010	100.000	56.96263	0.00	0.992	1.002
Totals (All Traced Routines)	12.925	100.000	100.000	26.69263	2827.71	2.571	0.802

Key:

- %AccumT = accumulated percentage of total CPU time
- %ExTime = percentage of total CPU time
- %Vflops = percentage of floating point operations due to vector floating point operations
- IBFR = Instruction Buffer Fetch Rate (megafetches/sec)
- IFact = Inline Factor (total calls to routine / average time spent in routine for each call)
- MC = number of memory conflicts
- MR = number of memory references
- Time = total CPU time (sec)

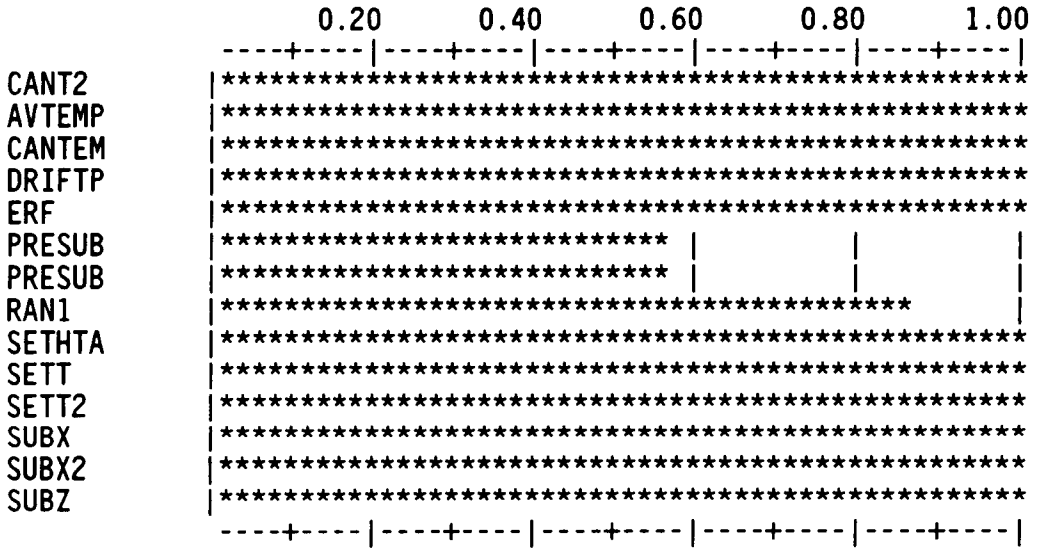
9. Coverage Analysis

One sample problem was supplied. A coverage analysis shows that this problem yielded a 92% segment coverage of CANT2. Sample problems provided with simulation programs typically achieve 35% to 50% coverage. A statement of software quality cannot be made for routines that have low coverage, i.e., large portions of the code are untested.

Two routines achieve 40%-59% coverage, 1 routine achieves 80%-99% coverage, and 11 routines achieve 100% coverage.

The following table shows the percent coverage for each routine.

Module Name	Number of Segments in module	Number of Segments Executed	Percent Segment Coverage
CANT2	17	17	100.0
AVTEMP	35	35	100.0
CANTEM	3	3	100.0
DRIFTP	8	8	100.0
ERF	6	6	100.0
PRESUB	9	5	55.6
PRESUB	9	5	55.6
RAN1	7	6	85.7
SETHTA	3	3	100.0
SETT	9	9	100.0
SETT2	6	6	100.0
SUBX	1	1	100.0
SUBX2	1	1	100.0
SUBZ	5	5	100.0
Totals	119	110	92.4



0.40 <= coverage < 0.60	PRESUB	PRESUB				
0.85 <= coverage < 0.90	RAN1					
coverage = 1.00	CANT2	AVTEMP	CANTEM	DRIFTP	ERF	
	SETHTA	SETT	SETT2	SUBX	SUBX2	
	SUBZ					

Program coverage for this run =0.92

10. Complexity Analysis

Some key metrics are the number of executable statements (sloc), the number of non-blank comments (ncomt), McCabe's extended cyclomatic complexity (vg2), the number of branching statements (cgoto, ugoto, bIF, and lIF), and Halstead's predicted number of errors in (re)writing the code (bhat). Measures are normalized per 100 executable statements for ease of comparison and are listed in the table below.

The branching measures for this code indicate few unconditional GO TO statements and logical IFs for most program modules. This code appears to be well structured.

Most routines have a low ratio of non-blank comments to source code. Additional comments would be helpful.

McCabe's extended cyclomatic complexity (vg2), normalized per 100 lines of source code, indicates moderate to high values. Generally, the routines with the highest complexity are those most likely to have defects. As a guideline, normalized measures of 15 or greater should be considered complex. A software maintenance program should focus on those routines with the highest measures.

Complexity Report by Subprogram for CANT2

Name	loc	sloc	cmnt	ncomt	ncomt /sloc	vg2 /sloc	cgoto	cgoto /sloc	ugoto	ugoto /sloc	bIF	bif /sloc	lIF	lif /sloc	Bhat
cant2	154	91	43	41	45.1	9.9	0	0.0	0	0.0	0	0.0	0	0.0	2
avtemp	97	83	13	11	13.3	20.5	0	0.0	0	0.0	0	0.0	4	4.8	2
cantemp	48	8	33	26	325.0	25.0	0	0.0	0	0.0	0	0.0	0	0.0	0
driftpt	27	18	6	6	33.3	22.2	0	0.0	0	0.0	1	5.6	0	0.0	0
erf	14	13	1	1	7.7	23.1	0	0.0	1	7.7	0	0.0	2	15.4	0
presub	13	13	1	1	7.7	38.5	0	0.0	1	7.7	0	0.0	3	23.1	0
presub2	13	13	1	1	7.7	38.5	0	0.0	1	7.7	0	0.0	3	23.1	0
ranl	29	24	0	0	0.0	25.0	0	0.0	0	0.0	1	4.2	1	4.2	0
sethtab	37	18	11	11	61.1	11.1	0	0.0	0	0.0	0	0.0	0	0.0	0
sett	76	53	12	12	22.6	9.4	0	0.0	0	0.0	1	1.9	0	0.0	1
sett2	55	41	10	10	24.4	7.3	0	0.0	0	0.0	0	0.0	0	0.0	1
subx	9	7	1	1	14.3	14.3	0	0.0	0	0.0	0	0.0	0	0.0	0
subx2	10	8	1	1	12.5	12.5	0	0.0	0	0.0	0	0.0	0	0.0	0
subz	15	12	3	3	25.0	25.0	0	0.0	0	0.0	0	0.0	2	16.7	0

Legend of Metrics in Report

loc -- lines of code
sloc -- number of executable statements
cmnt -- total number of commnts
ncomt -- number of non-blank COMMENT statements
100*ncomt/sloc -- percent, nonblank comments to number of executable statements
100*vg2/sloc -- percent, extended complexity of number of executable statements
cgoto -- number of COMPUTED GO TO statements
100*cgoto/sloc -- percent, computed GOTO's to number of executable statements
ugoto -- number of UNCONDITIONAL GO TO statements
100*ugoto/sloc -- percent, unconditional GOTO's to number of executable statements
bIF -- number of BLOCK IF statements
100*bif/sloc -- percent, Block IF statements to number of executable statements
lIF -- number of LOGICAL IF statements
100*lif/sloc -- percent, logical IF statements to number of executable statements
Bhat -- Halstead's predicted number of errors in writing code