SOFTWARE RELEASE NOTICE

01. SRN Number: PA-SRN-013

02. Project Title:
Volcano - A Module for Simulation of Magmatic Scenario.

Project No.

03. SRN Title: VOLCANO Version 1.1

04. Originator/Requester:
Thomas J. Ratchford

Date:
02/01/94

05. Summary of Actions

- ■ Release of new code admitted to CM System
- □ Release of modified code:
  - □ Enhancements made
  - □ Corrections made
- □ Change of access code

06. Persons Authorized Access

Name	RO/RW	A/C/D

07. Element Manager Approval: Date:

08. Remarks:

CNWRA Form TOP-6 (08/93)

SOFTWARE RELEASE NOTICE

01. SRN Number: PA-SRN-013

02. Project Title:
Volcano - A Module for Simulation of Magmatic Scenario.

Project No.

03. SRN Title: VOLCANO Version 1.1

04. Originator/Requester:
Thomas J. Ratchford

Date:
02/1/94

05. Summary of Actions

- ■ Release of new code admitted to CM System
- □ Release of modified code:
  - □ Enhancements made
  - □ Corrections made
- ■ Change of access code

06. Persons Authorized Access

Name	RO/RW	A/C/D

07. Element Manager Approval: Date:

08. Remarks:

CNWRA Form TOP-6 (08/93)

SOFTWARE SUMMARY FORM

01.Summary Date: 02/01/94	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: VOLCANO	
06. Software Title: Volcano - A module for simulation of magmatic scenario.		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	B. Baca, (210) 522-3805

13. Narrative:
The Volcano code is used by the Total Performance Assessment Code to simulate the magma scenarios in the Yucca Mountain region arround the potential repository.

14. Computer Platform	15. Computer Operating System:	16. Programming Language(s):	17. Number of Source Program Statements:
CRAY/XMP	UNIX	FORTRAN	15,639 lines of code
18. Computer Memory Requirements: UNKNOWN	19. Tape Drives: NONE	20. Disk/Drum Units: N/A	21. Graphics: 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: 2/1/94

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

Volcano Cray Directory Listing

```
-rwxrwx---   1 tjr1     tjr1         2209 Jan 31 08:50 Makefile*
-rwxrwx---   1 tjr1     tjr1         2385 Jan 31 08:50 TPA_VOL.VGD*
-rw-r-----   1 tjr1     tjr1            0 Jan 31 09:09 VALCANO.DIR
-rwxrwx---   1 tjr1     tjr1         4375 Jan 31 08:50 airout.F*
-rwxrwx---   1 tjr1     tjr1          473 Jan 31 08:50 betwn.F*
-rwxrwx---   1 tjr1     tjr1         1763 Jan 31 08:50 cdamag.F*
-rwxrwx---   1 tjr1     tjr1          216 Jan 31 08:50 ceil.F*
-rwxrwx---   1 tjr1     tjr1          690 Jan 31 08:50 chkele.F*
-rwxrwx---   1 tjr1     tjr1         1264 Jan 31 08:50 cirint.F*
-rwxrwx---   1 tjr1     tjr1         1291 Jan 31 08:50 coin.F*
-rwxrwx---   1 tjr1     tjr1         1253 Jan 31 08:50 ddamag.F*
-rwxrwx---   1 tjr1     tjr1         3193 Jan 31 08:50 decay.F*
-rwxrwx---   1 tjr1     tjr1          150 Jan 31 08:50 dist.F*
-rwxrwx---   1 tjr1     tjr1         2151 Jan 31 08:50 ditout.F*
-rwxrwx---   1 tjr1     tjr1          295 Jan 31 08:50 dot.F*
-rwxrwx---   1 tjr1     tjr1         3272 Jan 31 08:50 echo1.F*
-rwxrwx---   1 tjr1     tjr1         1836 Jan 31 08:50 echo2.F*
-rwxrwx---   1 tjr1     tjr1         5025 Jan 31 08:50 echo3.F*
-rwxrwx---   1 tjr1     tjr1         1482 Jan 31 08:50 getara.F*
-rwxrwx---   1 tjr1     tjr1         1651 Jan 31 08:50 getbrk.F*
-rwxrwx---   1 tjr1     tjr1         1065 Jan 31 08:50 getdc.F*
-rwxrwx---   1 tjr1     tjr1         1948 Jan 31 08:50 getlen.F*
-rwxrwx---   1 tjr1     tjr1         1390 Jan 31 08:50 gettb.F*
-rwxrwx---   1 tjr1     tjr1          969 Jan 31 08:50 inside.F*
-rwxrwx---   1 tjr1     tjr1          631 Jan 31 08:50 int1.F*
-rwxrwx---   1 tjr1     tjr1          482 Jan 31 08:50 int2.F*
-rwxrwx---   1 tjr1     tjr1          372 Jan 31 08:50 int3.F*
-rwxrwx---   1 tjr1     tjr1          937 Jan 31 08:50 intr.F*
-rwxrwx---   1 tjr1     tjr1          570 Jan 31 08:50 left.F*
-rwxrwx---   1 tjr1     tjr1       344250 Jan 31 08:50 lhsoooo.out*
-rwxrwx---   1 tjr1     tjr1         8962 Jan 31 08:50 opnfil.F*
-rwxrwx---   1 tjr1     tjr1          422 Jan 31 08:50 order.F*
-rwxrwx---   1 tjr1     tjr1         1863 Jan 31 08:50 params.H*
-rwxrwx---   1 tjr1     tjr1          900 Jan 31 08:50 ran1.F*
-rwxrwx---   1 tjr1     tjr1         1878 Jan 31 08:50 ranc.F*
-rwxrwx---   1 tjr1     tjr1         1596 Jan 31 08:50 rand.F*
-rwxrwx---   1 tjr1     tjr1         3921 Jan 31 08:50 rdelem.F*
-rwxrwx---   1 tjr1     tjr1         1909 Jan 31 08:50 rdenv.F*
-rwxrwx---   1 tjr1     tjr1         1707 Jan 31 08:50 rdmap.F*
-rwxrwx---   1 tjr1     tjr1         2892 Jan 31 08:50 rdrun.F*
-rwxrwx---   1 tjr1     tjr1          260 Jan 31 08:50 remove.F*
-rwxrwx---   1 tjr1     tjr1         4062 Jan 31 08:50 setup.F*
-rwxrwx---   1 tjr1     tjr1         1014 Jan 31 08:50 solve.F*
-rwxrwx---   1 tjr1     tjr1         2592 Jan 31 08:50 sotcom.H*
-rwxrwx---   1 tjr1     tjr1          309 Jan 31 08:50 split.F*
-rwxrwx---   1 tjr1     tjr1         1874 Jan 31 08:50 test15.nuc*
-rwxrwx---   1 tjr1     tjr1          452 Jan 31 08:50 valid.F*
-rwxrwx---   1 tjr1     tjr1         1727 Jan 31 08:50 vlcano.F*
-rwxrwx---   1 tjr1     tjr1       198329 Jan 31 08:50 vlcano.f*
-rwxrwx---   1 tjr1     tjr1        75766 Jan 31 08:50 volcano.cpp*
-rwxrwx---   1 tjr1     tjr1         2502 Jan 31 08:50 volcano.in*
-rwxrwx---   1 tjr1     tjr1        78769 Jan 31 08:50 volcano.pre*
-rwxrwx---   1 tjr1     tjr1         1701 Jan 31 08:50 volcom.H*
-rwxrwx---   1 tjr1     tjr1          209 Jan 31 08:50 volmap.dat*
-rwxrwx---   1 tjr1     tjr1         5011 Jan 31 08:50 wrel.F*
-rwxrwx---   1 tjr1     tjr1          287 Jan 31 08:50 x.volcano.co*
-rwxrwx---   1 tjr1     tjr1          287 Jan 31 08:50 x.volcano.te*
```

Directory from ~/tpa/VOLCANO/WKDIR
1/31/94 T.J. Ratchford

VOLCANO Fortran Program Static and Dynamic Analysis

June 29, 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 VOLCANO program contains 42 Fortran routines.

VOLCANO has no alternate entry points and no unreferenced subroutines or functions.

4. Common Block Irregularities

There are 9 common blocks in the VOLCANO program.

All common block declarations are consistent, except for the declaration of common block "envpar" in subroutine "setup" where the variables "seed" and "cmeth" are typed real. These variables are typed integer everywhere else "envpar" is declared.

There are many instances of a common block being declared in a routine in which none of its elements are otherwise referenced.

Of the 27 elements in common block "cnucl6", 23 are never defined or used anywhere in the VOLCANO program.

5. Interface Irregularities

Argument number 6 to "opnfil" should be a 64-character string, but is typed real in calls from "airout", "ditout", "rdenv", "rdrun", "rdelem", and "rdmap".

Argument number 2 in "split" is an 8-character string, but "wrel" passes a 6-character string to "split" in this argument.

Routines "ditout", "int1", and "wrel" each have dummy arguments which are unused.

Subroutine "rdrun" calls function "ran1" several times. The first call passes the expression "-seed" to "ran1" as an argument. This argument is assigned a value of 1 by "ran1" before returning to "rdrun". While no harm is done by this argument usage, owing to the fact that the expression is never used again, it may be better to change the statement

```
u1=RAN1(-seed)
```

to

```
idum = -seed
u1 = ran1 (idum)
```

to avoid confusion.

6. Local Variable Irregularities

Parameter usage is consistent.

There are several instances of a parameter not being used in a module in which it is declared.

The proper functioning of "ran1" depends upon the local variables "r", "iff", "ix1", "ix2", and "ix3" retaining their assigned values from the previous call to "ran1". These variables should therefore be placed in a common block.

7. Fortran Extensions

All routines except "ran1" contain some lower case alphabetic characters in their active Fortran.

Several routines have entity names which are longer than 6 characters.

Routines "echo1", "echo2", and "echo3" all have format statements containing the "x" descriptor without a repeat count.

Routine "rdelem" has 2 instances of overlap in character assignment statements.

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.

In order to obtain meaningful statistics for performance evaluation, the program should execute for a reasonable amount of time. Note that the execution time for this sample problem is short (<< 10 sec) and that the resulting statistics may therefore not accurately reflect program performance for longer runs.

The performance data show that a high percentage of the overall execution time (85.118%) is spent in the first 7 routines listed (RDMAP, OPNFIL, RDELEM, AIROUT, RDENV, ECHO1, WREL). This is due primarily to the following (applies to some or all of the 7 routines):

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

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

A detailed optimization analysis effort should focus on these 2 areas.

PERFORMANCE DATA FOR VOLCANO

ROUTINE NAME	Time	%ExTime	%AccumT	%Vflops	IFact	MC/MR	IBFR
RDMAP	0.022	34.124	34.124	0.00000	0.00	0.116	0.757
OPNFIL	0.006	9.663	43.786	0.00000	0.00	0.371	0.547
RDELEM	0.006	9.575	53.361	0.00000	0.00	0.096	1.076
AIROUT	0.006	9.114	62.476	0.00000	0.00	0.457	1.027
RDENV	0.005	8.342	70.817	0.00000	0.00	0.204	1.118
ECHO1	0.005	8.122	78.939	0.00000	0.00	0.273	0.910
WREL	0.004	6.179	85.118	47.44875	0.00	0.097	1.072
ECHO3	0.004	6.175	91.294	80.14185	0.00	0.200	0.897
ECHO2	0.001	1.795	93.088	0.00000	0.00	0.267	0.991
RDRUN	0.001	1.034	94.123	0.00000	0.00	0.175	0.786
DITOUT	0.001	0.931	95.054	0.00000	0.00	0.301	1.093
DECAY	0.001	0.865	95.919	71.03448	0.00	0.070	0.430
SETUP	0.000	0.634	96.554	37.88301	0.00	0.150	1.308
INTR	0.000	0.609	97.163	0.00000	0.10	0.254	1.384
GETLEN	0.000	0.576	97.738	0.00000	0.01	0.285	1.615
INSIDE	0.000	0.544	98.283	0.00000	0.03	0.020	1.207
LEFT	0.000	0.318	98.600	0.00000	0.40	0.482	0.952
DIST	0.000	0.296	98.896	0.00000	0.20	1.300	0.709
CHKELE	0.000	0.289	99.185	0.00000	0.02	0.433	0.661
SOLVE	0.000	0.217	99.403	0.00000	0.28	0.729	0.966
VLCANO	0.000	0.216	99.619	0.00000	0.00	0.793	1.091
SPLIT	0.000	0.112	99.731	0.00000	0.06	0.464	1.345
DDAMAG	0.000	0.106	99.837	0.00000	0.00	0.090	1.321
DOT	0.000	0.076	99.914	0.00000	0.20	0.484	0.000
CEIL	0.000	0.054	99.968	0.00000	0.19	0.138	0.000
RAND	0.000	0.019	99.987	0.00000	0.00	0.000	0.412
GETDC	0.000	0.013	100.000	0.00000	0.00	0.395	0.930
Totals (All Traced Routines)	0.065	100.000	100.000	30.15386	0.03	0.216	0.884

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. The sample problem aborted while trying to raise a negative number to a real power. The trouble was traced to function "dist" in which the following statement appears:

```
dist=SQRT((x2-x1)**2. + (y2-y1)**2.)
```

When the program is compiled without optimization, this statement will cause execution failure any time one of the base numbers is negative or zero due to the exponents being specified as floating point numbers (2.). Failure does not occur with compiler optimization because the optimizer replaces the real-to-real function call with a real-to-integer function call.

A similar problem exists with several other statements in the program. The trouble was rectified by eliminating the "." after the whole number exponent in each such instance.

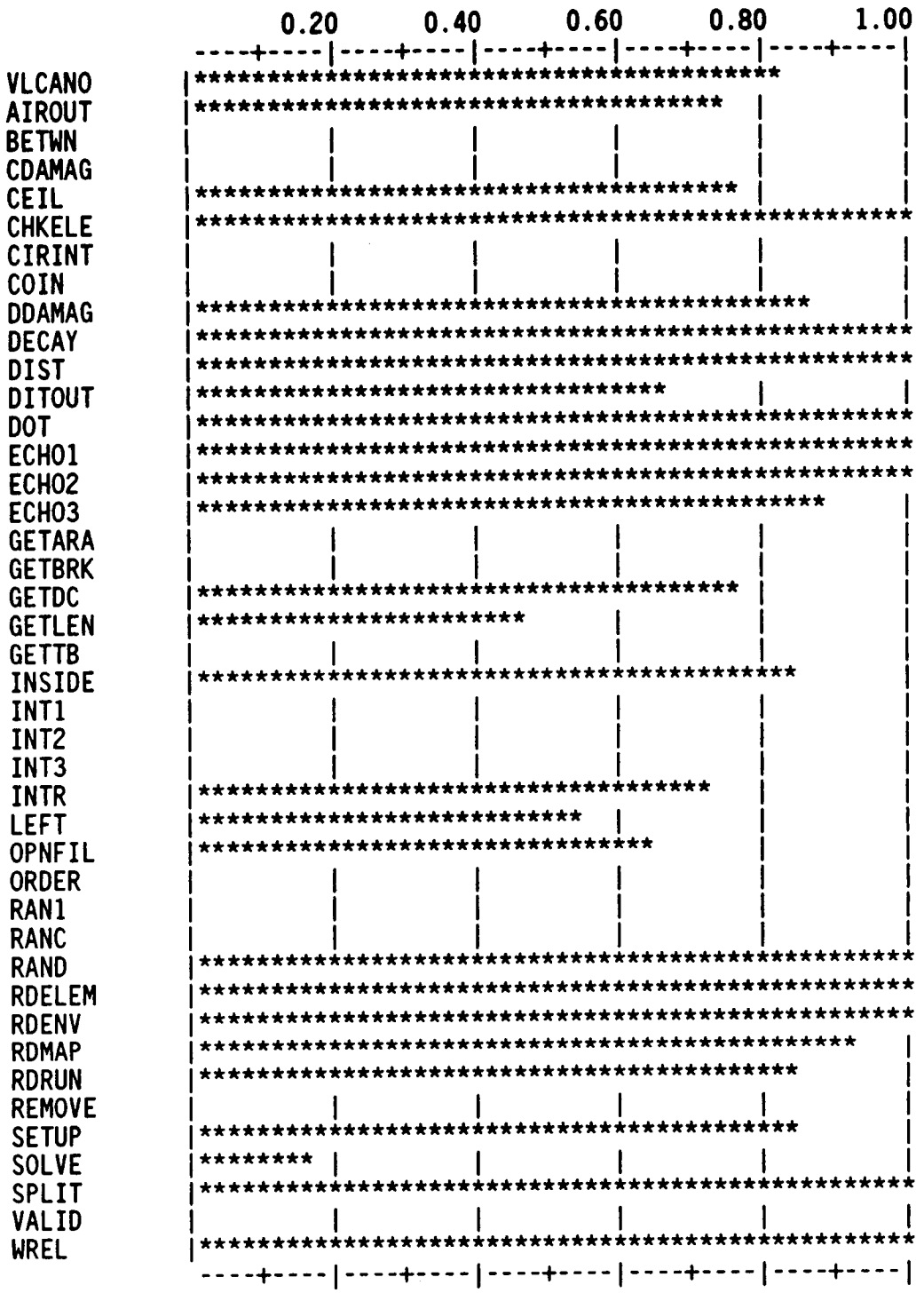
A coverage analysis shows that the supplied sample problem yielded a 56% segment coverage of VOLCANO. Sample problems provided with simulation programs typically achieve 35% to 50% coverage. A statement of software quality cannot be made for routines that have low coverage, i.e., large portions of the code are untested.

Note that 15 routines have 0% coverage. These routines are not tested with the supplied sample problem.

One routine achieves 1%-19% coverage, 2 routines achieve 40%-59% coverage, 6 routines achieve 60%-79% coverage, 7 routines achieve 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
VLCANO	11	9	81.8
AIROUT	19	14	73.7
BETWN	3	0	0.0
CDAMAG	13	0	0.0
CEIL	4	3	75.0
CHKELE	10	10	100.0
CIRINT	13	0	0.0
COIN	21	0	0.0
DDAMAG	7	6	85.7
DECAY	20	20	100.0
DIST	1	1	100.0
DITOUT	6	4	66.7
DOT	1	1	100.0
ECHO1	5	5	100.0
ECHO2	3	3	100.0
ECHO3	18	16	88.9
GETARA	17	0	0.0
GETBRK	15	0	0.0
GETDC	4	3	75.0
GETLEN	22	10	45.5
GETTB	11	0	0.0
INSIDE	6	5	83.3
INT1	1	0	0.0
INT2	4	0	0.0
INT3	1	0	0.0
INTR	7	5	71.4
LEFT	13	7	53.8
OPNFIL	28	18	64.3
ORDER	7	0	0.0
RAN1	7	0	0.0
RANC	4	0	0.0
RAND	1	1	100.0
RDELEM	14	14	100.0
RDENV	7	7	100.0
RDMAP	28	26	92.9
RDRUN	6	5	83.3
REMOVE	3	0	0.0
SETUP	30	25	83.3
SOLVE	18	3	16.7
SPLIT	3	3	100.0
VALID	3	0	0.0
WREL	22	22	100.0
Totals	437	246	56.3


0.20
0.40
0.60
0.80
1.00
VLCANO
AIROUT
BETWN
CDAMAG
CEIL
CHKELE
CIRINT
COIN
DDAMAG
DECAY
DIST
DITOUT
DOT
ECHO1
ECHO2
ECHO3
GETARA
GETBRK
GETDC
GETLEN
GETTB
INSIDE
INT1
INT2
INT3
INTR
LEFT
OPNFIL
ORDER
RAN1
RANC
RAND
RDELEM
RDENV
RDMAP
RDRUN
REMOVE
SETUP
SOLVE
SPLIT
VALID
WREL

coverage = 0.	BETWN	CDAMAG	CIRINT	COIN	GETARA
	GETBRK	GETTB	INT1	INT2	INT3
	ORDER	RAN1	RANC	REMOVE	VALID
0.01 <= coverage < 0.20	SOLVE				
0.40 <= coverage < 0.60	GETLEN	LEFT			
0.60 <= coverage < 0.80	AIROUT	CEIL	DITOUT	GETDC	INTR
	OPNFIL				
0.80 <= coverage < 0.85	VLCANO	INSIDE	RDRUN	SETUP	
0.85 <= coverage < 0.90	DDAMAG	ECHO3			
0.90 <= coverage < 0.95	RDMAP				
coverage = 1.00	CHKELE	DECAY	DIST	DOT	ECHO1
	ECHO2	RAND	RDELEM	RDENV	SPLIT
	WREL				

Program coverage for this run =0.56

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 very few unconditional GO TO statements and logical IFs for most program modules. This code appears to be well structured.

Many routines have a good ratio of non-blank comments to source code. Some have poor ratios, e.g., "ceil", "chkele", "coin".

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 VOLCANO

Name	loc	sloc	cmnt	ncomt	ncomt /sloc	vg2 /sloc	cgoto	cgoto /sloc	ugoto	ugoto /sloc	bif bIF	bif /sloc	lif lIF	lif /sloc	Bhat
vlcano	143	28	50	31	110.7	21.4	0	0.0	0	0.0	3	10.7	0	0.0	0
airOut	205	35	108	88	251.4	28.6	0	0.0	0	0.0	2	5.7	0	0.0	1
betwn	18	5	10	5	100.0	100.0	0	0.0	0	0.0	0	0.0	1	20.0	0
Cdamag	103	28	39	27	96.4	17.9	0	0.0	0	0.0	2	7.1	0	0.0	0
ceil	11	7	3	1	14.3	28.6	0	0.0	0	0.0	1	14.3	0	0.0	0
chkele	29	18	6	2	11.1	22.2	0	0.0	1	5.6	2	11.1	0	0.0	0
cirInt	43	25	15	9	36.0	24.0	0	0.0	0	0.0	3	12.0	2	8.0	0
coin	43	29	10	5	17.2	31.0	0	0.0	0	0.0	5	17.2	1	3.4	0
Ddamag	88	14	38	27	192.9	21.4	0	0.0	0	0.0	1	7.1	0	0.0	0
decay	151	29	78	62	213.8	34.5	0	0.0	0	0.0	1	3.4	1	3.4	0
dist	4	3	1	1	33.3	33.3	0	0.0	0	0.0	0	0.0	0	0.0	0
ditOut	115	18	59	47	261.1	27.8	0	0.0	0	0.0	1	5.6	0	0.0	0
dot	11	3	7	3	100.0	33.3	0	0.0	0	0.0	0	0.0	0	0.0	0
echo1	124	26	31	19	73.1	11.5	0	0.0	0	0.0	1	3.8	0	0.0	0
echo2	96	14	42	30	214.3	14.3	0	0.0	0	0.0	0	0.0	0	0.0	0
echo3	217	43	100	85	197.7	18.6	0	0.0	0	0.0	2	4.7	1	2.3	1
getAra	55	30	20	10	33.3	26.7	0	0.0	0	0.0	5	16.7	0	0.0	1
getBrk	110	36	37	21	58.3	19.4	0	0.0	0	0.0	2	5.6	1	2.8	0
getDC	76	9	25	17	188.9	22.2	0	0.0	0	0.0	1	11.1	0	0.0	0
getLen	111	40	35	22	55.0	27.5	0	0.0	0	0.0	3	7.5	3	7.5	1
getTB	100	31	32	20	64.5	22.6	0	0.0	0	0.0	3	9.7	0	0.0	0
inside	85	15	32	19	126.7	20.0	0	0.0	0	0.0	1	6.7	0	0.0	0
int1	26	14	10	4	28.6	7.1	0	0.0	0	0.0	0	0.0	0	0.0	0
int2	20	9	7	2	22.2	22.2	0	0.0	0	0.0	1	11.1	0	0.0	0
int3	15	5	7	2	40.0	20.0	0	0.0	0	0.0	0	0.0	0	0.0	0
intr	36	14	15	8	57.1	42.9	0	0.0	0	0.0	2	14.3	0	0.0	0
left	22	18	6	2	11.1	38.9	0	0.0	0	0.0	2	11.1	4	22.2	0
opnfil	208	59	132	119	201.7	27.1	0	0.0	0	0.0	8	13.6	0	0.0	0
order	22	10	8	3	30.0	40.0	0	0.0	0	0.0	1	10.0	0	0.0	0
RAN1	29	24	1	1	4.2	25.0	0	0.0	0	0.0	1	4.2	1	4.2	0
ranC	106	17	47	31	182.4	11.8	0	0.0	0	0.0	1	5.9	0	0.0	0
ranD	96	10	44	28	280.0	10.0	0	0.0	0	0.0	0	0.0	0	0.0	0
rdelem	162	28	85	73	260.7	25.0	0	0.0	0	0.0	2	7.1	0	0.0	0
rdEnv	112	31	37	23	74.2	12.9	0	0.0	0	0.0	0	0.0	0	0.0	1
rdmap	103	42	31	18	42.9	31.0	0	0.0	1	2.4	2	4.8	9	21.4	1
rdRun	184	35	79	60	171.4	8.6	0	0.0	0	0.0	1	2.9	0	0.0	0

Name	loc	sloc	cmnt	ncomt	ncomt /sloc	vg2 /sloc	cgoto	cgoto /sloc	ugoto	ugoto /sloc	bIF	bif /sloc	lIF	lif /sloc	Bhat
remove	14	4	7	2	50.0	50.0	0	0.0	0	0.0	0	0.0	0	0.0	0
setup	162	64	45	28	43.8	23.4	0	0.0	0	0.0	6	9.4	4	6.3	1
solve	48	28	17	9	32.1	21.4	0	0.0	0	0.0	5	17.9	0	0.0	0
split	11	6	3	1	16.7	33.3	0	0.0	0	0.0	1	16.7	0	0.0	0
valid	17	8	8	4	50.0	37.5	0	0.0	0	0.0	1	12.5	0	0.0	0
Wrel	221	55	98	85	154.5	21.8	0	0.0	0	0.0	1	1.8	0	0.0	1

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