SOFTWARE SUMMARY FORM

01.Summary Date: 04/01/94	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: SLIM	

06. Software Title: SLIM	07. Internal Software ID: NONE

08. Software Type:	09.Processing Mode:	10. APPLICATION AREA
☐ Automated Data System ■ Computer Program ☐ Subroutine/Module	☐ Interactive ☐ Batch ■ Combination	A. General: ■ Scientific/Engineering ☐ Auxiliary Analyses ☐ 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. Bagtzoglou, (210) 522-5182

13. Narrative:

SLIM - SLIM is a three dimensionl particle tracking code that simulates the migration of a single, neutrally buoyant, aqueous chemical component within a saturated ground water flow domain.

14. Computer Platform	15. Computer Operating System:	16. Programming Language(s):	17. Number of Source Program Statements:
CRAY/XMP	UNIX	FORTRAN	8,699 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: 4/1/94

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

```
gemstone.1 ~ => cd ~/SLIM/VCS
gemstone.2 ~/SLIM/VCS => ls -l
total 8011
-rwx------   1 tjr1     tjr1           0 Apr  1 08:31 plerst.dat*
-rwx------   1 tjr1     tjr1     20980895 Apr  1 08:34 s.V2*
-rwx------   1 tjr1     tjr1        1552 Apr  1 08:34 s.common.h*
-rwx------   1 tjr1     tjr1         505 Apr  1 08:34 s.open.h*
-rwx------   1 tjr1     tjr1       83860 Apr  1 08:42 s.plstat.dat*
-rwx------   1 tjr1     tjr1         666 Apr  1 08:39 s.rxn.1.dat*
-rwx------   1 tjr1     tjr1         173 Apr  1 08:39 s.rxn.2.dat*
-rwx------   1 tjr1     tjr1    10739949 Apr  1 08:40 s.rxn.log*
-rwx------   1 tjr1     tjr1       80602 Apr  1 08:37 s.slim.f*
-rwx------   1 tjr1     tjr1        1129 Apr  1 08:37 s.slim.time*
-rwx------   1 tjr1     tjr1      894958 Apr  1 08:38 s.slim.x*
```

MJR 4/1/94

134.20.1.1 09:59:29

SLIM Fortran Program
Static and Dynamic Analysis

March 10, 1994

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.

The sample problem was found to abort when SLIM is loaded with a core preset of indefinite. The program was loaded with a core preset of zero for the dynamic analysis runs.

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 SLIM program contains 29 Fortran routines.

There are no alternate entry points.

There are 3 extraneous subroutines: "place2", "place3", "remove".

4. Common Block Irregularities

There are 9 common blocks in the SLIM program.

Blank common differs in size in module "input2" from the initial declaration in module "slim".

Variable exceptions are noted as follows:

Block name	Variable	Exception
/parm1/	istat	undefined, unused
/parm1/	netsum	undefined, unused
/parm5/	s	undefined, unused
/parm6/	pintrp	defined, unused

There are several instances of a common block not being used by a module in which it is declared:

Block name	Modules not using
/parm0/	analy, displa2, displa3, mclear, plotb, plotcp, plotm, plotp, remove
/parm1/	erstat, plerst, plotc, plstat, statb, statf
/parm2/	erstat, input2, plotcp, plotp, remove, slim, statb, statf
/parm3/	analy, book, chem, conc, displa3, erstat, input2, input3, mclear, place2, place3, plerst, plotb, plotc, plotcp, plotm, plotp, plstat, remove, statb, statf, update
/parm4/	analy, book, chem, conc, displa1, displa2, displa3, driver, erstat, input1, input2, input3, mclear, place, place2, place3, plerst, plotb, plotc, plotcp, plotp, plstat, remove, slim, statb, statf, update
/parm5/	analy, book, chem, conc, displa2, displa3, driver, erstat, input2, input3, mclear, place, place2, place3, plerst, plotb,plotc, plotcp, plotm, plotp, plstat, print, remove, slim, statb, statf, update
/parm6/	analy, book, displa1, displa2, displa3, erstat, input3, mclear, plerst, plotb,plotc, plotcp, plotm, plotp, plstat, print, remove, slim, statb, statf, update
/units/	analy, book, conc, displa1, displa2, displa3, erstat, mclear, remove, statb, statf, update

5. Interface Irregularities

Exceptions are noted as follows:

Module	Line#	Exception
slim	397	argument #5 to DRIVER has the wrong type
slim	397	argument #8 to DRIVER has the wrong type
slim	397	argument #9 to DRIVER has the wrong type
slim	397	argument #10 to DRIVER has the wrong type
chem	80	CONC is called with too few arguments
chem	234	CONC is called with too few arguments
input2	77	argument #6 to PLACE has the wrong type
update	68	PLACE is called with too many arguments
update	79	PLACE is called with too many arguments

6. Local Variable Irregularities

Parameter usage is consistent.

Local variable exceptions are noted as follows:

Module	Variable	Exception
analy	kode	UNUSED
analy	ran	Undefined, Unused
analy	rm1	Undefined, Unused
analy	rssq1	Undefined, Unused
analy	zz1	Undefined, Unused
analy	zz2	Undefined, Unused
analy	zz3	Undefined, Unused
book	lold	Defined, Unused
book	ran	Undefined, Unused
book	rm1	Undefined, Unused
book	rssq1	Undefined, Unused
book	zz1	Undefined, Unused
book	zz2	Undefined, Unused
book	zz3	Undefined, Unused
chem	itype	Defined, Unused
chem	kode	UNUSED
chem	ran	Undefined, Unused
chem	rm1	Undefined, Unused
chem	rssq1	Undefined, Unused
chem	zz1	Undefined, Unused
chem	zz2	Undefined, Unused
chem	zz3	Undefined, Unused
conc	ran	Undefined, Unused
conc	rm1	Undefined, Unused
conc	rssq1	Undefined, Unused
conc	zz1	Undefined, Unused
conc	zz2	Undefined, Unused
conc	zz3	Undefined, Unused
displa1	ran	Undefined, Unused
displa1	rm1	Undefined, Unused
displa1	rssq1	Undefined, Unused
displa2	adx	Defined, Unused
displa2	ady	Defined, Unused
displa2	adz	Defined, Unused
displa2	ran	Undefined, Unused
displa2	rm1	Undefined, Unused
displa2	rssq1	Undefined, Unused
displa3	ran	Undefined, Unused
displa3	rm1	Undefined, Unused
displa3	rssq1	Undefined, Unused
displa3	v4	Defined, Unused
displa3	zz1	Undefined, Unused
displa3	zz2	Undefined, Unused
displa3	zz3	Undefined, Unused
driver	ran	Undefined, Unused
driver	rm1	Undefined, Unused
driver	rssq1	Undefined, Unused
driver	zz1	Undefined, Unused
driver	zz2	Undefined, Unused

driver	zz3	Undefined, Unused
erstat	ran	Undefined, Unused
erstat	rml	Undefined, Unused
erstat	rssql	Undefined, Unused
erstat	zz1	Undefined, Unused
erstat	zz2	Undefined, Unused
erstat	zz3	Undefined, Unused
input1	ran	Undefined, Unused
input1	rml	Undefined, Unused
input1	rssql	Undefined, Unused
input1	zz1	Undefined, Unused
input1	zz2	Undefined, Unused
input1	zz3	Undefined, Unused
input2	nbre	USED but UNDEFINED
input2	ran	Undefined, Unused
input2	rml	Undefined, Unused
input2	rssql	Undefined, Unused
input2	zz1	Undefined, Unused
input2	zz2	Undefined, Unused
input2	zz3	Undefined, Unused
input3	dummy1	Undefined, Unused
input3	dummy2	Undefined, Unused
input3	dummy3	Undefined, Unused
input3	ran	Undefined, Unused
input3	rml	Undefined, Unused
input3	rssql	Undefined, Unused
input3	th	UNUSED
input3	zz1	Undefined, Unused
input3	zz2	Undefined, Unused
input3	zz3	Undefined, Unused
mclear	ran	Undefined, Unused
mclear	rml	Undefined, Unused
mclear	rssql	Undefined, Unused
mclear	zz1	Undefined, Unused
mclear	zz2	Undefined, Unused
mclear	zz3	Undefined, Unused
place	ran	Undefined, Unused
place	rml	Undefined, Unused
place	rssql	Undefined, Unused
place2	ran	Undefined, Unused
place2	rml	Undefined, Unused
place2	rssql	Undefined, Unused
place2	zz1	Undefined, Unused
place2	zz2	Undefined, Unused
place2	zz3	Undefined, Unused
place3	ran	Undefined, Unused
place3	rml	Undefined, Unused
place3	rssql	Undefined, Unused
place3	zz1	Undefined, Unused
place3	zz2	Undefined, Unused
place3	zz3	Undefined, Unused
plerst	ran	Undefined, Unused
plerst	rml	Undefined, Unused
plerst	rssql	Undefined, Unused

plerst	zz1	Undefined, Unused
plerst	zz2	Undefined, Unused
plerst	zz3	Undefined, Unused
plotb	ran	Undefined, Unused
plotb	rml	Undefined, Unused
plotb	rssql	Undefined, Unused
plotb	zz1	Undefined, Unused
plotb	zz2	Undefined, Unused
plotb	zz3	Undefined, Unused
plotc	latrib	UNUSED
plotc	ran	Undefined, Unused
plotc	rml	Undefined, Unused
plotc	rssql	Undefined, Unused
plotc	zz1	Undefined, Unused
plotc	zz2	Undefined, Unused
plotc	zz3	Undefined, Unused
plotcp	ran	Undefined, Unused
plotcp	rml	Undefined, Unused
plotcp	rssql	Undefined, Unused
plotcp	zz1	Undefined, Unused
plotcp	zz2	Undefined, Unused
plotcp	zz3	Undefined, Unused
plotm	latrib	UNUSED
plotm	ran	Undefined, Unused
plotm	rml	Undefined, Unused
plotm	rssql	Undefined, Unused
plotm	zz1	Undefined, Unused
plotm	zz2	Undefined, Unused
plotm	zz3	Undefined, Unused
plotp	ran	Undefined, Unused
plotp	rml	Undefined, Unused
plotp	rssql	Undefined, Unused
plotp	zz1	Undefined, Unused
plotp	zz2	Undefined, Unused
plotp	zz3	Undefined, Unused
plstat	ran	Undefined, Unused
plstat	rml	Undefined, Unused
plstat	rssql	Undefined, Unused
plstat	zz1	Undefined, Unused
plstat	zz2	Undefined, Unused
plstat	zz3	Undefined, Unused
print	zz1	Undefined, Unused
print	zz2	Undefined, Unused
print	zz3	Undefined, Unused
remove	itype	UNUSED
remove	kode	UNUSED
remove	ran	Undefined, Unused
remove	rml	Undefined, Unused
remove	rssql	Undefined, Unused
remove	zz1	Undefined, Unused
remove	zz2	Undefined, Unused
remove	zz3	Undefined, Unused
slim	erav	Possibly undefined
slim	ersd	Possibly undefined

```
slim      erss      Possibly undefined
slim      ran       Undefined, Unused
slim      rml       Undefined, Unused
slim      rssql     Undefined, Unused
slim      zz1       Undefined, Unused
slim      zz2       Undefined, Unused
slim      zz3       Undefined, Unused
statb     ran       Undefined, Unused
statb     rml       Undefined, Unused
statb     rssql     Undefined, Unused
statb     zz1       Undefined, Unused
statb     zz2       Undefined, Unused
statb     zz3       Undefined, Unused
statf     ran       Undefined, Unused
statf     rml       Undefined, Unused
statf     rssql     Undefined, Unused
statf     zz1       Undefined, Unused
statf     zz2       Undefined, Unused
statf     zz3       Undefined, Unused
update    grid      UNUSED
update    ltyp      USED but UNDEFINED
update    ran       Undefined, Unused
update    rml       Undefined, Unused
update    rssql     Undefined, Unused
update    zz1       Undefined, Unused
update    zz2       Undefined, Unused
update    zz3       Undefined, Unused
```

7. Fortran Extensions

Each program module contains some lower case letters in the active Fortran.

Each program module contains entity names which are longer than 6 characters.

Modules "displa1" and "displa2" contain references to the non-standard intrinsic function "dfloat".

Format statement 30 in module "remove" has 2 consecutive strings which are not separated by a comma.

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 (90.959%) is spent in the first 3 routines listed. This is due primarily to a low percentage of floating point operations which are performed in vector mode (%Vflops is small). A detailed optimization analysis effort should focus on this area.

PERFORMANCE DATA FOR SLIM

ROUTINE NAME	Time	%ExTime	%AccumT	%Vflops	IFact	MC/MR	IBFR
DISPLA1	26.355	47.508	47.508	0.00000	0.00	0.501	0.311
CONC	18.883	34.039	81.547	60.69497	0.00	0.681	0.318
INPUT3	5.222	9.412	90.959	5.55432	0.00	0.633	0.813
PLSTAT	3.269	5.893	96.853	0.00000	0.00	0.927	0.991
BOOK	1.440	2.596	99.449	0.00000	0.00	0.968	0.004
PRINT	0.251	0.453	99.903	0.00000	0.00	0.961	0.390
PLACE	0.026	0.046	99.949	0.00000	0.00	1.092	0.467
DRIVER	0.011	0.020	99.969	0.00000	0.00	0.535	0.880
STATF	0.008	0.014	99.983	99.99967	0.00	0.289	0.001
SLIM	0.005	0.009	99.992	0.00000	0.00	0.832	0.612
UPDATE	0.001	0.002	99.994	0.00000	0.47	1.915	1.004
INPUT2	0.001	0.002	99.997	0.00000	0.00	0.925	0.648
INPUT1	0.001	0.002	99.999	0.00000	0.00	0.476	1.116
ANALY	0.001	0.001	100.000	0.00000	0.98	4.467	1.266
MCLEAR	0.000	0.000	100.000	0.00000	0.00	7.182	1.349
Totals (All Traced Routines)	55.475	100.000	100.000	33.99437	0.00	0.669	0.393

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)

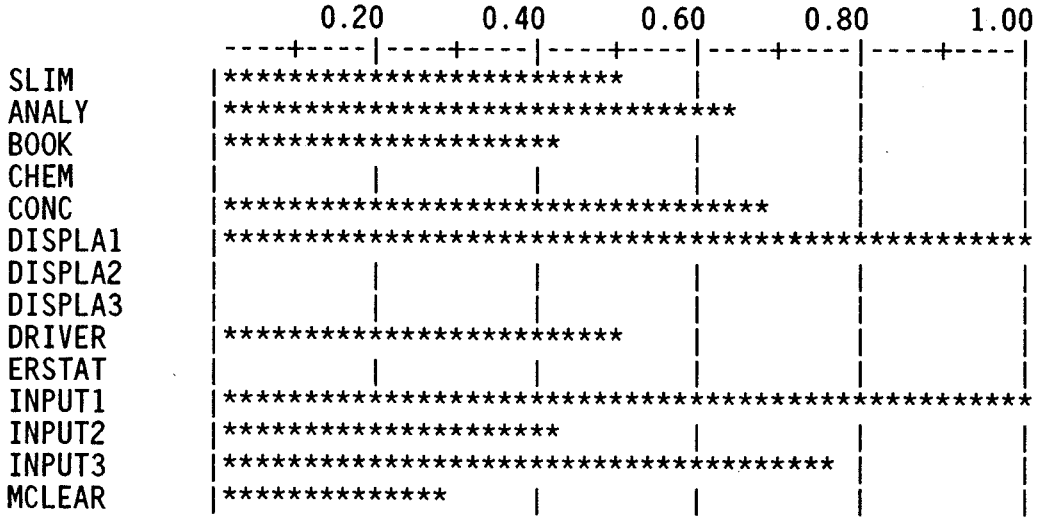
9. Coverage Analysis

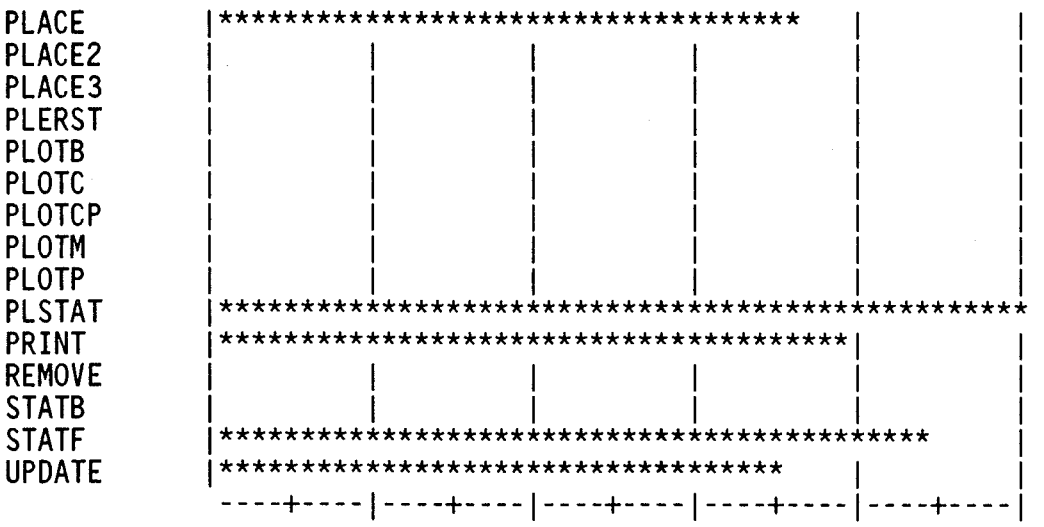
A coverage analysis shows that the sample problem yielded a 44% segment coverage of SLIM. Sample problems provided with simulation programs typically achieve only 35% to 50% coverage. A statement of software quality cannot be made for routines that have low coverage, i.e., large portions of the code are untested.

Note that 14 routines have 0% coverage. These routines are not tested with the supplied sample problem.

One routine achieves 21%-39% coverage, 4 routines achieve 40%-59% coverage, 6 routines achieve 60%-79% coverage, 1 routine achieves 85%-99% coverage, and 3 routines achieve 100% coverage.

Module Name	Number of Segments in module	Number of Segments Executed	Percent Segment Coverage
SLIM	4	2	50.0
ANALY	19	12	63.2
BOOK	60	25	41.7
CHEM	27	0	0.0
CONC	64	43	67.2
DISPLA1	3	3	100.0
DISPLA2	3	0	0.0
DISPLA3	3	0	0.0
DRIVER	73	37	50.7
ERSTAT	8	0	0.0
INPUT1	3	3	100.0
INPUT2	17	7	41.2
INPUT3	8	6	75.0
MCLEAR	28	8	28.6
PLACE	7	5	71.4
PLACE2	5	0	0.0
PLACE3	5	0	0.0
PLERST	3	0	0.0
PLOTB	1	0	0.0
PLOTC	3	0	0.0
PLOTCP	1	0	0.0
PLOTM	5	0	0.0
PLOTP	3	0	0.0
PLSTAT	3	3	100.0
PRINT	18	14	77.8
REMOVE	15	0	0.0
STATB	8	0	0.0
STATF	8	7	87.5
UPDATE	10	7	70.0
Totals	415	182	43.9





coverage = 0.	CHEM	DISPLA2	DISPLA3	ERSTAT	PLACE2
	PLACE3	PLERST	PLOTB	PLOTC	PLOTCP
	PLOTM	PLOTP	REMOVE	STATB	
0.20 <= coverage < 0.40	MCLEAR				
0.40 <= coverage < 0.60	SLIM	BOOK	DRIVER	INPUT2	
0.60 <= coverage < 0.80	ANALY	CONC	INPUT3	PLACE	PRINT
	UPDATE				
0.85 <= coverage < 0.90	STATF				
coverage = 1.00	DISPLA1	INPUT1	PLSTAT		

Program coverage for this run =0.44

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 fairly well structured.

Most routines have a good ratio of non-blank comments to source code.

McCabe's extended cyclomatic complexity (vg2), normalized per 100 lines of source code, indicates 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 SLIM

Name	loc	sloc	cmnt	ncomt	ncomt /sloc	vg2 /sloc	cgoto	cgoto /sloc	ugoto	ugoto /sloc	bIF	bif /sloc	lIF	lif /sloc	Bhat
slim	406	53	350	301	567.9	3.8	0	0.0	1	1.9	0	0.0	1	1.9	1
analy	74	20	43	26	130.0	50.0	0	0.0	1	5.0	1	5.0	8	40.0	0
book	186	113	78	46	40.7	28.3	0	0.0	2	1.8	6	5.3	7	6.2	2
chem	256	91	148	101	111.0	16.5	0	0.0	1	1.1	5	5.5	1	1.1	2
conc	284	174	92	47	27.0	20.1	0	0.0	0	0.0	5	2.9	2	1.1	3
displa1	162	87	59	32	36.8	2.3	0	0.0	0	0.0	0	0.0	0	0.0	2
displa2	116	50	52	32	64.0	4.0	0	0.0	0	0.0	0	0.0	0	0.0	1
displa3	73	21	34	23	109.5	9.5	0	0.0	0	0.0	0	0.0	0	0.0	0
driver	397	193	182	102	52.8	18.7	0	0.0	6	3.1	6	3.1	19	9.8	2
erstat	54	12	30	23	191.7	33.3	0	0.0	0	0.0	0	0.0	1	8.3	0
input1	87	29	46	28	96.6	6.9	0	0.0	0	0.0	0	0.0	0	0.0	0
input2	134	34	75	51	150.0	26.5	0	0.0	5	14.7	2	5.9	3	8.8	0
input3	106	21	67	53	252.4	23.8	0	0.0	0	0.0	1	4.8	0	0.0	0
mclear	112	67	28	21	31.3	19.4	0	0.0	4	6.0	3	4.5	0	0.0	1
place	107	41	42	26	63.4	9.8	0	0.0	0	0.0	1	2.4	1	2.4	1
place2	81	27	31	21	77.8	11.1	0	0.0	0	0.0	1	3.7	0	0.0	0
place3	84	30	31	21	70.0	10.0	0	0.0	0	0.0	1	3.3	0	0.0	0
plerst	52	7	28	21	300.0	28.6	0	0.0	0	0.0	0	0.0	0	0.0	0
plotb	50	3	30	22	733.3	33.3	0	0.0	0	0.0	0	0.0	0	0.0	0
plotc	66	8	40	30	375.0	25.0	0	0.0	0	0.0	0	0.0	0	0.0	0
plotcp	47	3	27	21	700.0	33.3	0	0.0	0	0.0	0	0.0	0	0.0	0
plotm	93	38	37	25	65.8	7.9	0	0.0	0	0.0	0	0.0	1	2.6	0
plotp	54	7	30	22	314.3	28.6	0	0.0	0	0.0	0	0.0	0	0.0	0
plstat	52	7	38	29	414.3	28.6	0	0.0	0	0.0	0	0.0	0	0.0	0
print	177	34	49	29	85.3	29.4	0	0.0	0	0.0	7	20.6	0	0.0	1
remove	90	26	42	31	119.2	30.8	0	0.0	2	7.7	2	7.7	0	0.0	0
statb	54	12	26	21	175.0	33.3	0	0.0	0	0.0	0	0.0	1	8.3	0
statf	54	12	26	21	175.0	33.3	0	0.0	0	0.0	0	0.0	1	8.3	0
update	80	26	30	20	76.9	23.1	0	0.0	0	0.0	1	3.8	1	3.8	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