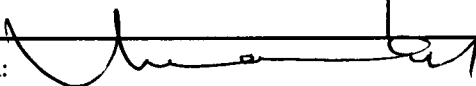


## SOFTWARE RELEASE NOTICE

01. SRN Number: RDCO-SRN-109		
02. Project Title: ISOSHL D - Isotope Shielding		Project No. 20-5702-622
03. SRN Title: ISOSHL D		
04. Originator/Requestor: Budhi Sagar		Date: 01/22/96
05. Summary of Actions <ul style="list-style-type: none"> <li><input type="checkbox"/> Release of new software</li> <li><input type="checkbox"/> Release of modified software:             <ul style="list-style-type: none"> <li><input type="checkbox"/> Enhancements made</li> <li><input type="checkbox"/> Corrections made</li> </ul> </li> <li><input type="checkbox"/> Change of access software</li> <li><input checked="" type="checkbox"/> Software Retirement</li> </ul>		
06. Persons Authorized Access		
Name	RO/RW	A/C/D
N/A		
07. Element Manager Approval: 		Date: 1/30/96
08. Remarks:  <p style="text-align: center;">Not considered important to regulatory reviews in revised FY96 OPS Plans.</p>		

# SOFTWARE SUMMARY FORM

01. Summary Date: 06/29/94		02. Summary prepared by(Name and Phone) T.J. Ratchford 522-3083		03. Summary Action:  New	
04. Software Date: 8/29/94		05. Short Title: ISOSHL D			
06. Software Title: ISOSHL D - Isotope Shielding.				07. Internal Software ID:  NONE	
08. Software Type:  <input type="checkbox"/> Automated Data System  <input checked="" type="checkbox"/> Computer Program  <input type="checkbox"/> Subroutine/Module		09. Processing Mode:  <input type="checkbox"/> Interactive  <input type="checkbox"/> Batch  <input checked="" type="checkbox"/> Combination		10. APPLICATION AREA A. General: <input type="checkbox"/> Scientific/Engineering <input type="checkbox"/> Auxiliary Analyses <input type="checkbox"/> Total System PA <input type="checkbox"/> Subsystem PA <input checked="" type="checkbox"/> Other  b. Specific:	
11. Submitting Organization and Address:  CNWRA, SwRI, San Antonio, Texas			12. Technical Contact(s) and Phone:  H. Karimi, (210) 522-5253		
13. Narrative:  ISOSHL D - A program for calculating solutions of radioisotopes shielding problems.					
14. Computer Platform  CRAY/XMP		15. Computer Operating System:  UNIX		16. Programming Language(s):  FORTRAN	
17. Number of Source Program Statements: 12,085 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: <input checked="" type="checkbox"/> Available <input type="checkbox"/> Limited <input type="checkbox"/> In-House ONLY			24. Documentation Availability: <input checked="" type="checkbox"/> Available <input type="checkbox"/> Inadequate <input type="checkbox"/> In-House ONLY		
25. Submission Package Status:  Acceptance Criteria: Met <input checked="" type="checkbox"/> Not Met <input type="checkbox"/> Software QA Assessment: Successful <input checked="" type="checkbox"/> Unsuccessful <input type="checkbox"/>  Code Custodian: <u>T.J. Ratchford</u> Date: <u>6/29/94</u>					

**CENTER FOR NUCLEAR REGULATORY WASTE ANALYSES**

**CHECKLIST FOR CNWRA CMS CODE CUSTODIAN UNDER TOP-018**

CODE: ISOSHLD Ver.1.1

RESPONSIBLE: H. Karimi

TARGET SUBMISSION DATE: Apr. 94

- ☐ Software Licensing Agreement (if not needed, N/A ☒ )
- ☒ Software Summary Form [6.1.1.1, 8.2.2, Appendix A]
- ☐ Software Requirements Document [7.3.1, 8.2.1]  
(if not needed, N/A ☒ )
- ☒ Code User's Manual [6.1.1.2, 6.2.2.1, 8.2.3.1]  
☐ preliminary ( final date \_\_\_\_\_ )      OR      ☒ final
- ☒ Code Technical Description [6.2.2.2, 8.2.2]  
(if not needed, N/A ☐ )
- ☒ Two Copies of Code in Electronic Format [6.1.1.3, 7.4.3]
- ☒ Evidence of Input Data Tractability to Output [6.1.1.4, 6.2.4]
- ☒ Evidence of Verification Reliability [6.1.1.5]
- ☒ Evidence of Benchmark Test Reliability [6.2.5]  
(if not possible or necessary, N/A ☐ )
- ☒ CRAFT Analyses ( N/A ☐ )
- ☒ FORWARN Analyses ( N/A ☐ )
- ☒ PC-Metric ( N/A ☐ )

*IF ALL OF THE ABOVE ARE COMPLETE, THEN THE CODE PACKAGE IS READY FOR SUBMISSION.*

- ☒ Date Code Package Submitted: 06/29/94

  
\_\_\_\_\_  
CNWRA CMS Custodian

[...] - refer to applicable sections of TOP-18

## ISOSHL D CRAY DIRECTORY LISTING

-rwxrwx---	1	tjr1	tjr1	1494	Jun	29	07:34	Makefile*
-rwxrwx---	1	tjr1	tjr1	2540	Jun	29	07:35	adjust.F*
-rwxrwx---	1	tjr1	tjr1	18821	Jun	29	07:35	ancyl.F*
-rwxrwx---	1	tjr1	tjr1	57513	Jun	29	07:35	bd.F*
-rwxrwx---	1	tjr1	tjr1	2354	Jun	29	07:35	beta.F*
-rwxrwx---	1	tjr1	tjr1	9020	Jun	29	07:35	bfunc.F*
-rwxrwx---	1	tjr1	tjr1	4822	Jun	29	07:35	blibe.F*
-rwxrwx---	1	tjr1	tjr1	20109	Jun	29	07:35	byield.F*
-rwxrwx---	1	tjr1	tjr1	47923	Jun	29	07:35	contrl.F*
-rwxrwx---	1	tjr1	tjr1	13305	Jun	29	07:35	cyl.F*
-rwxrwx---	1	tjr1	tjr1	2889	Jun	29	07:35	disc.F*
-rwxrwx---	1	tjr1	tjr1	227	Jun	29	07:35	drum-co*
-rwxrwx---	1	tjr1	tjr1	1137	Jun	29	07:35	drum-cs*
-rwxrwx---	1	tjr1	tjr1	1820	Jun	29	07:35	dscsrc.F*
-rwxrwx---	1	tjr1	tjr1	2208	Jun	29	07:35	e1.F*
-rwxrwx---	1	tjr1	tjr1	1345	Jun	29	07:35	e2.F*
-rwxrwx---	1	tjr1	tjr1	7358	Jun	29	07:35	endcyl.F*
-rwxrwx---	1	tjr1	tjr1	1978	Jun	29	07:35	f1.F*
-rwxrwx---	1	tjr1	tjr1	269	Jun	29	07:35	iacs*
-rwxrwx---	1	tjr1	tjr1	464	Jun	29	07:35	isa15*
-rwxrwx---	1	tjr1	tjr1	526	Jun	29	07:35	isa15m3*
-rwxrwx---	1	tjr1	tjr1	250	Jun	29	07:35	iscs*
-rwxrwx---	1	tjr1	tjr1	125195	Jun	29	07:35	iso-pc.lib*
-rwxrwx---	1	tjr1	tjr1	460	Jun	29	07:35	iso.in*
-rwxrwx---	1	tjr1	tjr1	233	Jun	29	07:35	isops*
-rwxrwx---	1	tjr1	tjr1	279	Jun	29	07:35	isopsm3*
-rwxrwx---	1	tjr1	tjr1	504	Jun	29	07:35	isoro2*
-rwxrwx---	1	tjr1	tjr1	468	Jun	29	07:35	isoro2b*
-rwxrwx---	1	tjr1	tjr1	389	Jun	29	07:35	isoru-rh*
-rwxrwx---	1	tjr1	tjr1	2359	Jun	29	07:35	isoshld.F*
-rwxrwx---	1	tjr1	tjr1	228297	Jun	29	07:35	isoshld.SRC*
-rw-r-----	1	tjr1	tjr1	0	Jun	29	11:52	isoshld.dir
-rwxrwx---	1	tjr1	tjr1	228297	Jun	29	07:35	isoshld.src*
-rwxrwx---	1	tjr1	tjr1	3586	Jun	29	07:35	line.F*
-rwxrwx---	1	tjr1	tjr1	2538	Jun	29	07:35	linsrc.F*
-rwxrwx---	1	tjr1	tjr1	2359	Jun	29	07:35	point.F*
-rwxrwx---	1	tjr1	tjr1	6513	Jun	29	07:35	rect.F*
-rwxrwx---	1	tjr1	tjr1	2635	Jun	29	07:35	simps.F*
-rwxrwx---	1	tjr1	tjr1	3259	Jun	29	07:35	sphere.F*
-rwxrwx---	1	tjr1	tjr1	3901	Jun	29	07:35	sphsrc.F*
-rwxrwx---	1	tjr1	tjr1	460	Jun	29	07:35	ssi-in*
-rwxrwx---	1	tjr1	tjr1	6067	Jun	29	07:35	tcone.F*
-rwxrwx---	1	tjr1	tjr1	463	Jun	29	07:35	terp.F*
-rwxrwx---	1	tjr1	tjr1	398	Jun	29	07:35	terpb.F*
-rwxrwx---	1	tjr1	tjr1	214	Jun	29	07:35	xpn.F*

4/12 6/29/94

# ISOSHLD Fortran Program Static and Dynamic Analysis

**DRAFT**

June 7, 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 used 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 ISOSHLD program aborts when executing the sample problem with a core preset of indefinite. It was therefore re-loaded with a core preset of zeros for the 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 ISOSHLD program contains 27 Fortran routines. There is 1 block data routine.

Some externals are declared but never used:

External	Declared in
-----	-----
dscsrc	disc
linsrc	line
sphsrc	sphere

## 4. Common Block Irregularities

There are 2 common blocks in the ISOSHLD program.

Common block variable exceptions are noted as follows:

Block	Variable	Exception
-----	-----	-----
//	ncd	Used but undefined
//	mue	Undefined and unused
//	data2	Undefined and unused
//	total	Undefined and unused
//	fmwd	Undefined and unused
//	tet	Undefined and unused
//	tlh	Undefined and unused
//	tlh	Undefined and unused
//	modsav	Used but undefined

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

Block name	Modules not using
-----	-----
/blok1/	adjust, ancyl, beta, bfunc, blibe, contrl, cyl, disc, dscsrc, endcyl, line, linsrc, point, rect, sphere, sphsrc, tcone

Some common block variables are altered by function subprograms:

Block name	Modifying functions
-----	-----
//	linsrc, sphsrc

Some common blocks have inconsistent layouts:

Block name	Different in	At variable
-----	-----	-----
//	beta	mu
//	byield	mu

The variables "mu" and "mue" in common block // are of inconsistent data types from one routine to another.

## 5. Interface Irregularities

Exceptions are noted as follows:

Module	Exception
-----	-----
contrl	argument #1 to "beta" has the wrong type

## 6. Local Variable Irregularities

Local variable exceptions are noted as follows:

Module	Variable	Exception
-----	-----	-----
cyl	tanpsi	Defined, Unused
isoshld	ch	Undefined, Unused

## 7. Fortran Extensions

Module "contrl" uses namelist I/O.

Module "isoshld" contains lowercase characters in its active Fortran.

Modules "contrl", "terp", and "terpb" contain INTEGER\*n declarations.

Modules "isoshld" and "contrl" contain entity names which are longer than 6 characters.

Modules "blibe" and "contrl" have format statements which contain fields 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.

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 more typical (possibly longer) runs.

The performance data show that a high percentage of the overall execution time (99.780%) is spent in the first 5 routines listed. This is due primarily to the following (applies to some or all of the 5 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 ISOSHL D

ROUTINE NAME	Time	%ExTime	%AccumT	%Vflops	IFact	MC/MR	IBFR
XPN	0.701	35.714	35.714	0.00000	452.21	2.622	0.775
BFUNC	0.578	29.466	65.180	0.00000	22.98	1.531	1.369
ANCYL	0.369	18.788	83.968	0.00000	0.00	0.893	1.334
CONTRL	0.198	10.110	94.078	0.52557	0.00	0.280	1.138
BLIBE	0.112	5.702	99.780	0.00000	0.00	0.315	1.221
ISOSHL D	0.004	0.213	99.994	0.00000	0.00	0.597	0.703
TERPB	0.000	0.004	99.998	72.72728	0.06	0.435	0.918

TERP	0.000	0.002	100.000	62.50000	0.03	0.458	0.933
ADJUST	0.000	0.000	100.000	0.00000	0.00	0.579	0.849
=====							
Totals (All Traced Routines)	1.962	100.000	100.000	0.02214	234.50	1.268	1.117

## 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 29% segment coverage of ISOSHL D. 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 17 routines have 0% coverage. These routines are not tested with the supplied sample problem.

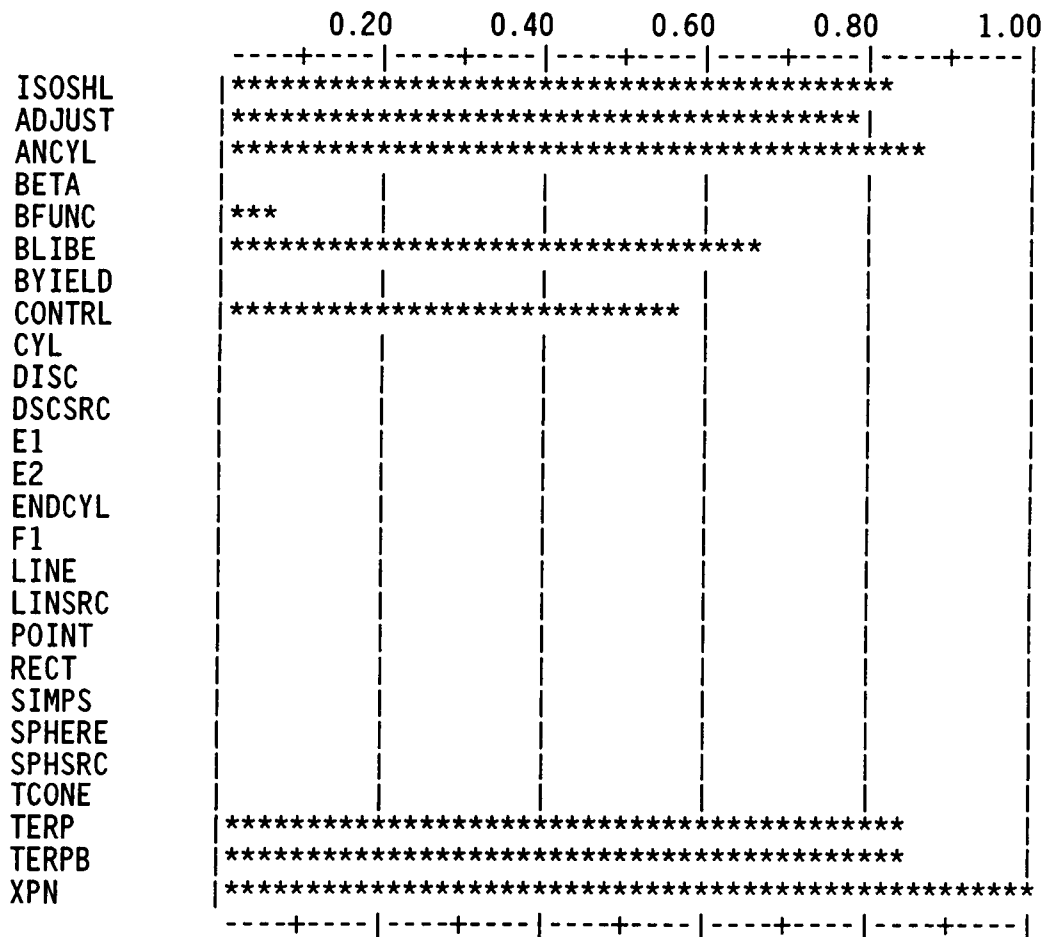
One routine achieves 1%-19% coverage, 1 routine achieves 40%-59% coverage, 2 routines achieve 60%-79% coverage, 4 routines achieve 80%-99% coverage, and 1 routine achieves 100% coverage.

Module Name	Number of Segments in module	Number of Segments Executed	Percent Segment Coverage
ISOSHL	11	9	81.8
ADJUST	9	7	77.8
ANCYL	88	76	86.4
BETA	5	0	0.0
BFUNC	75	5	6.7
BLIBE	30	20	66.7
BYIELD	180	0	0.0
CONTRL	370	207	55.9
CYL	113	0	0.0
DISC	12	0	0.0
DSCSRC	4	0	0.0
E1	12	0	0.0
E2	5	0	0.0
ENDCYL	63	0	0.0
F1	21	0	0.0
LINE	16	0	0.0
LINSRC	7	0	0.0
POINT	8	0	0.0
RECT	42	0	0.0



SIMPS	17	0	0.0
SPHERE	10	0	0.0
SPHSRC	14	0	0.0
TCONE	38	0	0.0
TERP	6	5	83.3
TERPB	6	5	83.3
XPN	6	6	100.0

Totals	1168	340	29.1
--------	------	-----	------



coverage = 0.

BETA	BYIELD	CYL	DISC	DSCSRC
E1	E2	ENDCYL	F1	LINE
LINSRC	POINT	RECT	SIMPS	SPHERE
SPHSRC	TCONE			

0.01 &lt;= coverage &lt; 0.20

BFUNC

0.40 &lt;= coverage &lt; 0.60

CONTRL

0.60 <= coverage < 0.80	ADJUST	BLIBE	
0.80 <= coverage < 0.85	ISOSHL	TERP	TERPB
0.85 <= coverage < 0.90	ANCYL		
coverage = 1.00	XPN		

Program coverage for this run =0.29

## 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 (ugoto/sloc, lif/sloc) indicate high values for several routines. This code may benefit from a restructuring effort aimed at reducing the number of unconditional GO TO and logical IF statements in such routines.

Several routines show a poor ratio of non-blank comments to source code. This code may benefit from more internal documentation.

M 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 ISOSHL D

Name	loc	sloc	cmnt	ncomt	ncomt /sloc	vg2 /sloc	cgoto	cgoto /sloc	ugoto	ugoto /sloc	bIF	bif /sloc	lIF	lif /sloc	Bhat
ISOSHL D	64	38	18	15	39.5	7.9	0	0.0	3	7.9	1	2.6	1	2.6	0
ADJUST	34	16	7	7	43.8	25.0	0	0.0	2	12.5	0	0.0	2	12.5	0
ANCYL	240	176	78	78	44.3	22.7	1	0.6	24	13.6	0	0.0	28	15.9	2
BETA	33	14	7	7	50.0	21.4	0	0.0	0	0.0	0	0.0	0	0.0	0
BFUNC	123	87	23	23	26.4	23.0	0	0.0	16	18.4	0	0.0	8	9.2	1
BLIBE	64	45	8	8	17.8	31.1	0	0.0	10	22.2	0	0.0	8	17.8	1
BYIELD	258	191	46	46	24.1	32.5	0	0.0	31	16.2	0	0.0	6	3.1	3
CONTRL	779	528	165	155	29.4	26.7	3	0.6	78	14.8	11	2.1	62	11.7	11
CYL	173	169	14	14	8.3	27.2	0	0.0	28	16.6	0	0.0	29	17.2	4
DISC	39	14	10	10	71.4	28.6	0	0.0	2	14.3	0	0.0	1	7.1	0
DSCSRC	25	9	4	4	44.4	22.2	0	0.0	1	11.1	0	0.0	1	11.1	0
E1	29	22	6	6	27.3	22.7	0	0.0	5	22.7	0	0.0	3	13.6	0
E2	25	11	13	13	118.2	27.3	0	0.0	0	0.0	2	18.2	0	0.0	0
ENDCYL	94	90	5	5	5.6	30.0	0	0.0	14	15.6	0	0.0	17	18.9	1
F1	26	21	6	6	28.6	61.9	0	0.0	7	33.3	0	0.0	4	19.0	0
LINE	48	20	17	17	85.0	30.0	0	0.0	2	10.0	0	0.0	3	15.0	0
LINSRC	34	16	5	5	31.3	12.5	0	0.0	1	6.3	0	0.0	0	0.0	0
POINT	32	14	6	6	42.9	28.6	0	0.0	1	7.1	0	0.0	2	14.3	0
RECT	87	80	6	6	7.5	26.3	0	0.0	8	10.0	0	0.0	12	15.0	1
SIMPS	34	22	11	11	50.0	27.3	0	0.0	1	4.5	0	0.0	1	4.5	0
SPHERE	43	15	15	15	100.0	26.7	0	0.0	1	6.7	0	0.0	1	6.7	0
SPHSRC	51	29	6	6	20.7	20.7	0	0.0	1	3.4	0	0.0	0	0.0	0
TCONE	79	53	9	9	17.0	26.4	0	0.0	9	17.0	0	0.0	4	7.5	1
TERP	14	9	3	3	33.3	33.3	0	0.0	1	11.1	0	0.0	1	11.1	0
TERPB	13	9	2	2	22.2	33.3	0	0.0	1	11.1	0	0.0	1	11.1	0
XP N	10	9	0	0	0.0	22.2	0	0.0	0	0.0	1	11.1	0	0.0	0

## Legend of Metrics in Report

loc -- lines of code

sloc -- number of executable statements

cmnt -- total number of comments

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  
 $100 * \text{cgoto} / \text{sloc}$  -- percent, computed GOTO's to number of executable statements  
ugoto -- number of UNCONDITIONAL GO TO statements  
 $100 * \text{ugoto} / \text{sloc}$  -- percent, unconditional GOTO's to number of executable statements  
bIF -- number of BLOCK IF statements  
 $100 * \text{bif} / \text{sloc}$  -- percent, Block IF statements to number of executable statements  
lIF -- number of LOGICAL IF statements  
 $100 * \text{lif} / \text{sloc}$  -- percent, logical IF statements to number of executable statements  
Bhat -- Halstead's predicted number of errors in writing code