SOFTWARE RELEASE NOTICE

01. SRN Number: RDCO-SRN-115	
02. Project Title: QAD-CGGP - A Point Kernel Code System for Neutron and Gamma-Ray Shielding Calculations Using the GP Buildup Factor.	Project No. 20-5702-611
03. SRN Title: QAD-CGGP	
04. Originator/Requestor: Budhi Sagar	Date: 01/22/96

05. Summary of Actions

- ☐ Release of new software
- ☐ Release of modified software:
  - ☐ Enhancements made
  - ☐ Corrections made
- ☐ Change of access software
- ■ Software Retirement

06. Persons Authorized Access

Name	RO/RW	A/C/D
N/A		

07. Element Manager Approval: [signature] Date: 1/30/96

08. Remarks:

Not considered important to regulatory reviews in revised FY96 OPS Plans.

CNWRA Form TOP-6 (06/95)

SOFTWARE SUMMARY FORM

01.Summary Date: 06/27/94	02. Summary prepared by(Name and Phone) T.J. Ratchford 522-3083	03. Summary Action: New
04. Software Date: 8/27/94	05. Short Title: QAD-CGGP	
06. Software Title: QAD-CGGP - A Point Kernel Code System for Neutron and Gamma-Ray Shielding Calculations Using the GP Buildup Factor.		07. Internal Software ID: NONE

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

QAD-CGGP - QAD-CGGP is a point-kernel code for calculating the fast-neutron and gamma-ray penetration through various shield configurations defined by combinatorial geometry specifications.

14. Computer Platform CRAY/XMP	15. Computer Operating System: UNIX	16. Programming Language(s): FORTRAN	17. Number of Source Program Statements: 25,996 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: ■ Available ☐ Limited ☐ In-House ONLY	24. Documentation Availability: ■ 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: 6/27/94

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

QAD 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.

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 QAD program contains 31 Fortran routines.

There is 1 alternate entry point: "slitet" in module "slite".

Three modules are not used: "geom", "gomflp", and "norml".

4. Common Block Irregularities

There are 13 common blocks in the QAD program.

There are 2 common blocks which are declared only once in the program:

Single Use Block	Module
/kk/	slite
/normal/	norml

Common block variable exceptions are noted as follows:

Block name	Variable	Exception
/gomloc/	naad	Defined but unused
/gomloc/	nadd	Defined but unused
/orgi/	nblz	Defined but unused
/orgi/	blzold	Defined but unused
/parem/	wp	Undefined and unused
/parem/	n0a	Undefined and unused
/qadata/	nbound	Unused
/stdata/	xsecea	Undefined and unused
/stdata/	dconst	Undefined and unused
/tape/	int	Undefined and unused
/tape/	iedt	Undefined and unused
/tape/	int1	Undefined and unused
/tape/	int2	Undefined and unused
/tape/	iagg	Undefined and unused
/tape/	itran	Undefined and unused
/tape/	inter	Undefined and unused

Common blocks which are declared, but not used, in a module are:

Block name	Modules not using
/gomloc/	norml
/orgi/	bdata1
/tape/	norml

Some common blocks have inconsistent declarations:

Block name	Different in	Type of difference
//	jomin	size
//	pr	layout
/dbg/	length	layout
/orgi/	length	size
/orgi/	lookz	size
/qadata/	print	size
/qadata/	source	layout
/tape/	jomin	size

5. Interface Irregularities

Exceptions are noted as follows:

Module	Exception
jomin	Argument #2 to GENI has the wrong type
jomin	Argument #1 to GTVLIN has the wrong type
jomin	Argument #3 to GTVLIN has the wrong type
length	Argument #5 to LOOKZ has the wrong type
length	Argument #3 to G1 has the wrong type

6. Local Variable Irregularities

Local variable exceptions are noted as follows:

Module	Variable	Exception
dtlist	listdt	Defined, Unused
gl	irl	UNUSED
gl	ir2	UNUSED
geni	il	UNUSED
input	epsln	Defined, Unused
input	kerpl	Defined, Unused
input	nlpt	UNUSED
jomin	isave	Defined, Unused
norml	iout	USED but UNDEFINED
qad	ierr3	Defined, Unused
qad	ip	Defined, Unused
qad	nlpt	Possibly undefined

7. Fortran Extensions

The following modules have format statements containing fields which are not separated by commas as required by the standard:

```
input, length, litebf, pr, print, source.
```

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 (91.158%) is spent in the first 6 routines listed. This is due primarily to the following (applies to some or all of the 6 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 rate of instruction buffer fetches (IBFR > 1).

A detailed optimization analysis effort should focus on these 3 areas.

PERFORMANCE DATA FOR QAD

ROUTINE NAME	Time	%ExTime	%AccumT	%Vflops	IFact	MC/MR	IBFR
BUILD	4.220	23.943	23.943	0.00004	214.26	0.000	0.850
KERNEL	2.960	16.795	40.738	69.82759	6.23	0.032	0.945

GG	2.772	15.727	56.465	0.00000	665.49	0.146	0.740
G1	2.705	15.347	71.812	0.00000	61.38	0.136	1.447
LENGTH	2.039	11.572	83.384	80.83730	9.04	0.071	1.404
FUNCF	1.370	7.775	91.158	0.00000	659.87	0.026	0.952
QAD	0.530	3.008	94.166	80.50551	0.00	0.075	1.431
SLITET	0.438	2.485	96.651	0.00000	1052.97	0.001	1.596
LOOKZ	0.435	2.470	99.122	0.01609	42.36	0.570	1.284
READBF	0.064	0.361	99.482	99.02622	0.00	0.087	0.938
INPUT	0.058	0.331	99.813	61.67872	0.00	0.075	0.954
DTLIST	0.012	0.069	99.883	99.80130	0.00	0.210	0.453
GENI	0.005	0.030	99.913	84.70522	0.00	0.178	0.969
SOURCE	0.005	0.028	99.941	60.13465	0.00	0.212	1.048
JOMIN	0.005	0.027	99.969	89.20301	0.00	0.288	0.987
PRINT	0.002	0.010	99.979	72.92554	0.00	0.225	0.890
LITEBF	0.002	0.010	99.989	45.64018	0.00	0.201	1.138
TITLE	0.001	0.003	99.992	86.43479	0.00	0.532	0.738
GTVLIN	0.000	0.003	99.995	88.76092	0.00	0.202	0.883
SLITE	0.000	0.002	99.997	0.00000	1.16	0.014	0.587
XSECQD	0.000	0.002	99.999	98.76508	0.00	0.088	0.058
SETP	0.000	0.001	100.000	57.11372	0.00	0.537	0.442
EXIT	0.000	0.000	100.000	100.00000	0.00	0.289	0.954
SETIO	0.000	0.000	100.000	0.00000	0.01	0.000	0.989
Totals (All Traced Routines)	17.623	100.000	100.000	30.61699	1282.65	0.085	1.060

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)
```

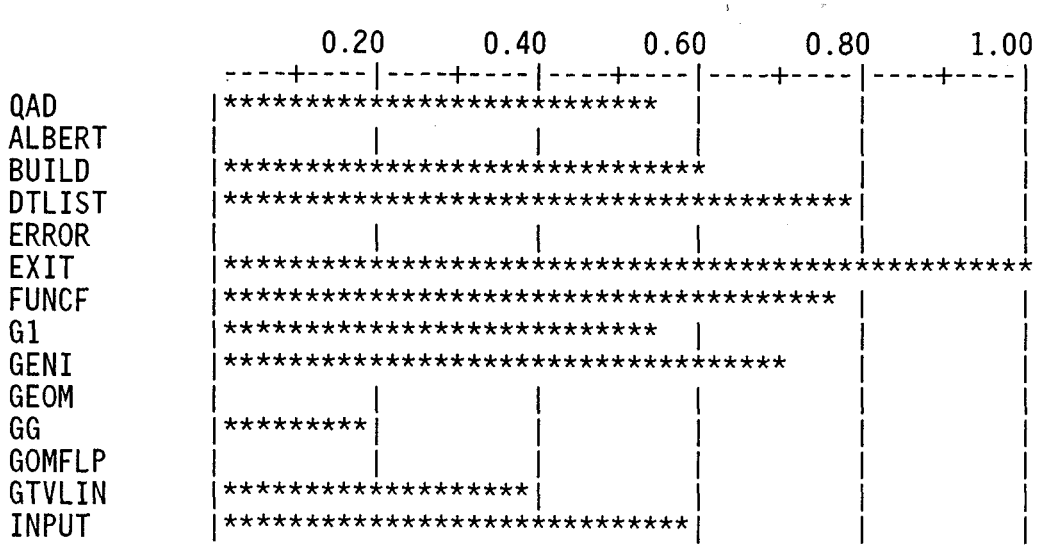
9. Coverage Analysis

A coverage analysis shows that the sample problem yielded a 40% segment coverage of QAD. 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 6 routines have 0% coverage. These routines are not tested with the supplied sample problem.

Two routines achieve 1%-19% coverage, 3 routines achieve 20%-39% coverage, 7 routines achieve 40%-59% coverage, 8 routines achieve 60%-79% coverage, 1 routine achieves 95%-99% coverage, and 2 routines achieve 100% coverage.

Module Name	Number of Segments in module	Number of Segments Executed	Percent Segment Coverage
QAD	183	98	53.6
ALBERT	47	0	0.0
BUILD	25	15	60.0
DTLIST	9	7	77.8
ERROR	1	0	0.0
EXIT	1	1	100.0
FUNCF	4	3	75.0
G1	119	64	53.8
GENI	57	40	70.2
GEOM	1	0	0.0
GG	304	55	18.1
GOMFLP	1	0	0.0
GTVLIN	27	10	37.0
INPUT	101	58	57.4
JOMIN	59	42	71.2
KERNEL	41	14	34.1
LENGTH	22	21	95.5
LITEBF	6	6	100.0
LOOKZ	50	25	50.0
NORML	97	0	0.0
PR	3	0	0.0
PRINT	51	8	15.7
READBF	20	15	75.0
SETIO	3	2	66.7
SETP	30	18	60.0
SLITE	15	12	80.0
SOURCE	111	34	30.6
TITLE	14	9	64.3
XSECQD	44	19	43.2
Totals	1446	576	39.8



```
JOMIN      |***********************************   |         |
KERNEL     |*****************  |         |         |         |
LENGTH     |************************************************ |
LITEBF     |*************************************************
LOOKZ      |************************   |         |         |
NORML      |         |         |         |         |         |
PR         |         |         |         |         |         |
PRINT      |******** |         |         |         |         |
READBF     |************************************  |         |
SETIO      |********************************     |         |
SETP       |******************************       |         |
SLITE      |****************************************         |
SOURCE     |***************   |         |         |         |
TITLE      |*******************************      |         |
XSECQD     |**********************      |         |         |
            ----+----|----+----|----+----|----+----|----+----|
```

coverage = 0.	ALBERT PR	ERROR	GEOM	GOMFLP	NORML
0.01 <= coverage < 0.20	GG	PRINT			
0.20 <= coverage < 0.40	GTVLIN	KERNEL	SOURCE		
0.40 <= coverage < 0.60	QAD SETP	BUILD XSECQD	G1	INPUT	LOOKZ
0.60 <= coverage < 0.80	DTLIST SETIO	FUNCF SLITE	GENI TITLE	JOMIN	READBF
0.95 <= coverage < 1.00	LENGTH				
coverage = 1.00	EXIT	LITEBF			

Program coverage for this run =0.40

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 moderate to 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 these routines.

The table shows low values for the ratio of non-blank comments to source code for many routines. This program could probably use more internal documentation.

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 QAD

Name	loc	sloc	cmnt	ncomt	ncomt /sloc	vg2 /sloc	cgoto	cgoto /sloc	ugoto	ugoto /sloc	bIF	bif /sloc	lIF	lif /sloc	Bhat
QAD	285	215	32	32	14.9	25.6	6	2.8	12	5.6	0	0.0	1	0.5	6
ALBERT	82	75	9	3	4.0	30.7	0	0.0	6	8.0	0	0.0	8	10.7	2
BUILD	53	42	13	6	14.3	28.6	0	0.0	9	21.4	2	4.8	7	16.7	1
DTLIST	30	16	10	7	43.8	18.8	0	0.0	3	18.8	0	0.0	2	12.5	0
ERROR	7	3	3	2	66.7	33.3	0	0.0	0	0.0	0	0.0	0	0.0	0
EXIT	9	4	3	3	75.0	25.0	0	0.0	0	0.0	0	0.0	0	0.0	0
FUNCF	10	8	1	0	0.0	25.0	0	0.0	0	0.0	1	12.5	0	0.0	0
G1	189	168	20	15	8.9	26.8	0	0.0	32	19.0	0	0.0	27	16.1	1
GENI	161	123	14	9	7.3	17.9	0	0.0	16	13.0	0	0.0	14	11.4	2
GEOM	4	3	1	1	33.3	33.3	0	0.0	0	0.0	0	0.0	0	0.0	0
GG	543	528	75	30	5.7	20.5	3	0.6	104	19.7	0	0.0	73	13.8	2
GOMFLP	4	2	2	2	100.0	50.0	0	0.0	0	0.0	0	0.0	0	0.0	0
GTVLIN	61	43	12	12	27.9	23.3	1	2.3	6	14.0	0	0.0	5	11.6	0
INPUT	304	172	36	29	16.9	19.2	1	0.6	5	2.9	0	0.0	6	3.5	4
JOMIN	155	120	15	11	9.2	17.5	0	0.0	14	11.7	0	0.0	12	10.0	2
KERNEL	76	45	8	5	11.1	26.7	0	0.0	4	8.9	1	2.2	1	2.2	1
LENGTH	89	59	3	3	5.1	18.6	2	3.4	2	3.4	0	0.0	2	3.4	1
LITEBF	18	10	2	1	10.0	30.0	0	0.0	0	0.0	0	0.0	0	0.0	0
LOOKZ	93	80	7	4	5.0	35.0	0	0.0	11	13.8	0	0.0	11	13.8	1
NORML	173	138	25	16	11.6	27.5	3	2.2	16	11.6	0	0.0	6	4.3	3
PR	43	15	11	8	53.3	20.0	0	0.0	0	0.0	0	0.0	1	6.7	1
PRINT	119	45	1	1	2.2	33.3	0	0.0	1	2.2	0	0.0	0	0.0	1
READBF	59	40	10	4	10.0	20.0	0	0.0	7	17.5	0	0.0	6	15.0	0
SETIO	25	5	21	18	360.0	40.0	0	0.0	0	0.0	0	0.0	1	20.0	0
SETP	81	59	18	9	15.3	22.0	0	0.0	10	16.9	0	0.0	6	10.2	1
SLITE	19	22	2	2	9.1	31.8	0	0.0	1	4.5	0	0.0	6	27.3	0
SOURCE	212	136	25	25	18.4	22.8	4	2.9	18	13.2	0	0.0	0	0.0	2
TITLE	24	11	1	1	9.1	45.5	2	18.2	2	18.2	0	0.0	0	0.0	0
XSECQD	83	51	19	19	37.3	33.3	0	0.0	5	9.8	0	0.0	6	11.8	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

Cray QAD-CGGP Directory Listing

-r--r-----	1	tjr1	tjr1	1500	Jun	24	07:34	Makefile
-r--r-----	1	tjr1	tjr1	2249	Jun	24	07:34	albert.F
-r--r-----	1	tjr1	tjr1	6048	Jun	24	07:34	bdata1.F
-r--r-----	1	tjr1	tjr1	7289	Jun	24	07:34	bdata2.F
-r--r-----	1	tjr1	tjr1	1619	Jun	24	07:34	build.F
-r--r-----	1	tjr1	tjr1	639	Jun	24	07:34	dtlist.F
-r--r-----	1	tjr1	tjr1	243	Jun	24	07:34	error.F
-r--r-----	1	tjr1	tjr1	280	Jun	24	07:34	exit.F
-r--r-----	1	tjr1	tjr1	27386	Jun	24	07:34	fort.10
-r--r-----	1	tjr1	tjr1	97560	Jun	24	07:34	fort.11
-r--r-----	1	tjr1	tjr1	257	Jun	24	07:34	funcf.F
-r--r-----	1	tjr1	tjr1	5251	Jun	24	07:34	g1.F
-r--r-----	1	tjr1	tjr1	4543	Jun	24	07:34	geni.F
-r--r-----	1	tjr1	tjr1	65	Jun	24	07:34	geom.F
-r--r-----	1	tjr1	tjr1	12280	Jun	24	07:34	gg.F
-r--r-----	1	tjr1	tjr1	95	Jun	24	07:34	gomflp.F
-r--r-----	1	tjr1	tjr1	1796	Jun	24	07:34	gtvlin.F
-r--r-----	1	tjr1	tjr1	11527	Jun	24	07:34	input.F
-r--r-----	1	tjr1	tjr1	4418	Jun	24	07:34	jomin.F
-r--r-----	1	tjr1	tjr1	2818	Jun	24	07:34	kernel.F
-r--r-----	1	tjr1	tjr1	3021	Jun	24	07:34	length.F
-r--r-----	1	tjr1	tjr1	579	Jun	24	07:34	litebf.F
-r--r-----	1	tjr1	tjr1	2723	Jun	24	07:34	lookz.F
-r--r-----	1	tjr1	tjr1	4369	Jun	24	07:34	norml.F
-r--r-----	1	tjr1	tjr1	1779	Jun	24	07:34	pr.F
-r--r-----	1	tjr1	tjr1	5455	Jun	24	07:34	print.F
-r--r-----	1	tjr1	tjr1	9794	Jun	24	07:34	qad.F
-r--r-----	1	tjr1	tjr1	108366	Jun	24	07:34	qad.SRC
-r--r-----	1	tjr1	tjr1	105820	Jun	24	07:34	qad.src
-rw-r-----	1	tjr1	tjr1	0	Jun	27	08:26	qad_dir.tjr
-rw-r-----	1	tjr1	tjr1	524288	Jun	24	08:50	qad_tar.tjr
-r--r-----	1	tjr1	tjr1	2512	Jun	24	07:34	qadgp.inp
-r--r-----	1	tjr1	tjr1	40029	Jun	24	07:34	qadgp.out
-r--r-----	1	tjr1	tjr1	1743	Jun	24	07:34	readbf.F
-r--r-----	1	tjr1	tjr1	984	Jun	24	07:34	setio.F
-r--r-----	1	tjr1	tjr1	2216	Jun	24	07:34	setp.F
-r--r-----	1	tjr1	tjr1	418	Jun	24	07:34	slite.F
-r--r-----	1	tjr1	tjr1	7135	Jun	24	07:34	source.F
-r--r-----	1	tjr1	tjr1	1003	Jun	24	07:34	title.F
-r--r-----	1	tjr1	tjr1	205	Jun	24	07:34	x.qad.covr
-r--r-----	1	tjr1	tjr1	286	Jun	24	07:34	x.qad.test
-r--r-----	1	tjr1	tjr1	3184	Jun	24	07:34	xsecqd.F

TJR 6/27/94