DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof. Reference herein to any social initiative (including but not limited to Diversity, Equity, and Inclusion (DEI); Community Benefits Plans (CBP); Justice 40; etc.) is made by the Author independent of any current requirement by the United States Government and does not constitute or imply endorsement, recommendation, or support by the United States Government or any agency thereof.

ANL/NSE-25/41

Code Coverage Status of the ARC Code RCT

Nuclear Science and Engineering Division

ANL/NSE-25/41

Code Coverage Status of the ARC Code RCT

prepared by

Micheal A. Smith
Nuclear Science and Engineering Division, Argonne National Laboratory

August 15, 2025

ABSTRACT

The Argonne Reactor Code (ARC) software system supports users in their fast reactor design goals by providing neutronic, thermal-hydraulic, and structural analysis capabilities. REBUS plays a pivotal role in the ARC system as the primary fuel cycle analysis capability for fast reactor problems. Over its 60 year history, ARC software usage with REBUS has been applied to numerous fast and thermal spectrum reactor analysis projects with good to excellent comparison against experiments. The RCT code is a later addition and uses the REBUS restart files to define its input. The RCT code was built to provide pin depletion details on EBR-II models and thus many features of RCT were specifically tailored to the needs of EBR-II models. Additional approximations were invoked which are likely only valid for the EBR-II reactor and the particular fuel management that was done for it.

The purpose of the present work is to identify a set of test problems for RCT and assess the code coverage for those test problems. The goal is to document what parts of the existing RCT code are touched by the set of test problems and which are not. Because no detailed verification work has been done on RCT, the existing regression testing suite was chosen for the code coverage assessment. The code coverage analysis of RCT was performed with the Code Coverage Tool of the Intel Fortran compiler which requires modifications to the compilation of RCT. The detailed coverage tables are given for each part of RCT. As will be discussed and shown, some parts of the RCT capability that are known to be used by the EBR-II analysis work are not tested by the regression testing suite. These aspects should be resolved before major source code changes are taken for the RCT software. Because REBUS and DIF3D are not subroutines of RCT, the coverage changes in both of those codes is not altered by RCT. The same is true for all of the modules of DIF3D that are used by RCT such as SYSLIB and SEGLIB.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. Introduction

The Argonne Reactor Code (ARC) software package consists of 13 primary and many secondary pieces of software that are connected through interface files. The ARC software suite supports users in their fast reactor design goals by providing neutronic, thermal-hydraulic, and structural analysis capabilities. Over the past decades, the ARC software suite has been applied to numerous fast and thermal spectrum reactor analysis projects. For those cases with experimental measurements, ARC performed exceptionally well, yielding results of consistent accuracy to those produced by the Monte Carlo method at a fraction of the cost. At present, the ARC software suite is primarily used for the analysis of advanced reactors. The typical design process workflow for the fuel cycle analysis is shown in Figure 1.1, with the cited codes named in orange boxes, and as can be seen, REBUS [1]-[3] and DIF3D [1][5]-[10] are central to the entire ARC system.

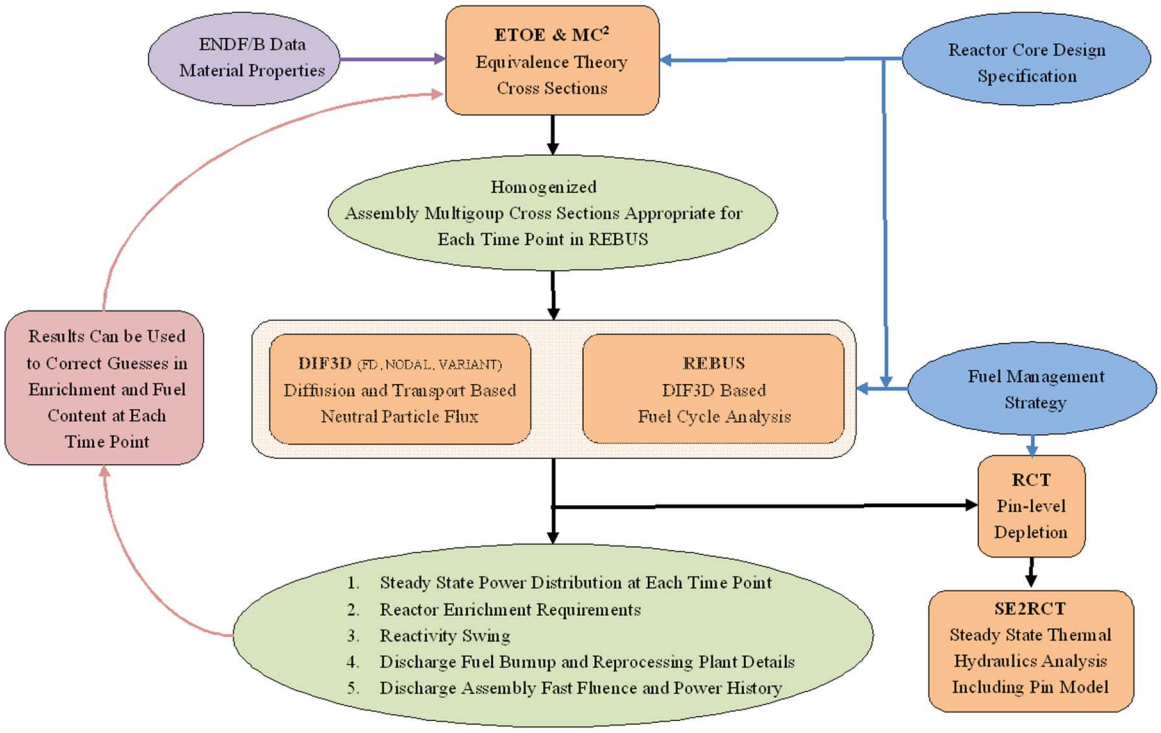

Figure 1.1 ARC Software Connections for Fast Spectrum Fuel Cycle Analysis

The ARC Software Quality Assurance (SQA) Program [18] is aimed to provide the controls and processes necessary to enable software improvement while meeting user and program sponsor requirements. The Software QA Plan (SQAP) delineates the SQA Program framework for the ARC software by describing the Program activities, organization, and documentation, and by clearly defining the interconnection of all program items. The analysis presented in this report is part of the SQAP for each piece of the ARC software.

Much of the ARC software has its origins in the 1960s with many components added or upgraded over the following 60 years of continuous development. RCT [4] was built around REBUS in the 1990s and directly reads the restart files generated by REBUS as part of its input. REBUS is built around the DIF3D code and thus is geometrically limited to the models that can be constructed with DIF3D. RCT is further restricted to the hexagonal 3D geometries of DIF3D-Nodal and DIF3D-VARIANT. Today, the diffusion and transport capabilities of DIF3D-VARIANT are primarily used in the reactor design process with some scattered usage of DIF3D-FD and DIF3D-Nodal and thus this limitation is not really relevant.

RCT uses just a few subroutines from REBUS and only links to a few modules that are common to DIF3D and REBUS (e.g. SYSLIB, SEGLIB and GNIP4C). As a consequence, this code coverage assessment will primarily focus on the RCT specific source code coverage but give details on the other connected components as secondary importance.

The purpose of the present work is to identify a set of test problems for RCT and document the code coverage of RCT for those test problems. Because RCT is distributed with a regression test suite and there is at present no verification work done on it, the set of test problems for the code coverage assessment will be the existing test suite.

In the following sections, the parts of the existing RCT coding which are or are not touched by the set of test problems are documented. The coverage of the connected components is included in a follow up section. The code coverage analysis was performed using the Code Coverage Tool of the Intel Fortran compiler (gprof and codecov). To use this capability, the compilation of RCT must be modified to create a special RCT executable which is different from the production executable. The existing regression test suite will be used to verify that the new executable is correct, in addition to serving as the code coverage assessment test cases. A detailed code coverage report is generated by the Intel Code Coverage Tool, which shows the coverage of subroutines/functions, blocks within each subroutine/function, and the frequency of execution of each subroutine/function. With this code coverage report, it is easy to figure out which subroutines/functions are fully covered, fully uncovered, or partially covered in RCT. A review of the RCT source files that are not fully covered was performed and some indication is reported here for the lack of coverage and its importance for the RCT code usage. One unique behavior of the version of gprof used is that it does not provide call information for all subroutines within a file and thus some details on the call history are not provided in all of the tables here.

2. RCT Overview

RCT was designed around the REBUS code for the purpose of doing EBR-II pin depletion analysis. RCT is not independent of REBUS in that it requires the user to use REBUS in a non-equilibrium calculation mode and the restart files generated by REBUS to define the RCT input. RCT uses the exact same binary interface files and variable naming scheme as in DIF3D. Over its extensive history, REBUS has been applied to numerous fast and thermal spectrum reactor analysis projects, giving users the confidence needed to select it for use with their reactor analysis project. RCT on the other hand was only constructed for the specific need to do pin level depletion analysis for ongoing EBR-II PIE measurements and thus has only ever been involved in those tasks. Using RCT to carry out pin depletion analysis of all assemblies of an existing core model is not advisable today as it is not an integral part of the search procedure used in REBUS. For more details, the user is directed to the RCT manual [4].

3. RCT Coverage Assessment

Because of the limited computational capabilities, the RCT code was developed to be a series of executables that are called in sequence which is termed the modular system. The executables passed information via interface files and had specific roles as part of the RCT execution (e.g., input processing, depletion calculation, and output processing). Today the structure of the RCT modular form is maintained but it is not used, as the driver routines of each non-driver module were converted to subroutines of the driver module. While the approach still uses interface files to transfer information between modules of RCT, the maintenance and use of the software is much easier. All of the modules used in RCT are listed in Table 3.1.

Table 3.1 Modules used by RCT

Modules	Directory locations
RCT (driver)	source
REBUS	rebus/source
Fortran 90 modules	rebus/dif3d/arcmodules/source
SYSLIB	rebus/dif3d/syslib/source
SEGLIB	rebus/dif3d/seglib/source
G4CH4C	rebus/dif3d/gnip4c_hmg4c/source

When RCT is compiled, only a few source files in each module (directory) are compiled and linked as part of the executable which is not the case for DIF3D and REBUS. This is consistent with the purpose of RCT as a simple manipulation of the flux solution coming from DIF3D at each time point of a REBUS depletion calculation. The additional complexity is that it also carries out a pin depletion calculation for an identified mesh in the existing REBUS model.

Unlike DIF3D and REBUS which have detailed software verification reports [11],[13], RCT does not have one. RCT is distributed with 7 test cases and most have 4 independent RCT calculations. Because RCT was primarily developed for and used on EBR-II, one would expect that one of the test cases would be derived from EBR-II. This is not the case and instead all of the test cases are for a single 60 degree periodic geometry model. The first 5 test cases are focused on DIF3D-Nodal and the last two were added when support was included for DIF3D-VARIANT in RCT. None of these test cases can be considered verification of what the software does and thus the code coverage assessment will highlight those aspects of the RCT capabilities that are not handled because of the limited testing suite. The previous work done on code coverage for DIF3D and REBUS can be found in references [15] and [16].

3.1 Primary subroutines and driver of the RCT code

The primary subroutines of the RCT code are stored in the directory *source* in addition to the driver routines. The code coverage of all of the files is shown in Table 3.1.1 and Table 3.1.2. Only a single subroutine used for debugging was uncovered as detailed in Table 3.1.2. For the uncovered or partially-covered subroutines, a description of the uncovered parts is also given in each table. For

the covered source files, the number of calls by the test package are also listed, showing the usage frequency of each source file for the 7 test cases (22 uses of RCT executable).

Starting with the code coverage of the driver and module subroutines in Table 3.1.1, one can see that several of them are partially covered and only the driver routine is fully covered. As detailed in the uncovered portion description, all of the uncovered sections deal with fatal errors and branching for alternative output edits. One item of interest is the older version of REBUS. It is not clear why two versions of REBUS were supported, but this aspect only has to do with the size of the storage arrays for files which has been periodically increased over the years.

For the partially-covered subroutines listed in Table 3.1.2, one can see that there are again many parts dealing with fatal errors that are not covered and thus can be ignored. Much of the "blocking" related aspects deal with when the memory limits on the machine were very small and thus these branches are likely to never be reached today (ISOTXS file would have to be generated in that state in order for REBUS/RCT to even deal with it). Similar to this are the various chi treatments. The chi cross section data is not actually used by RCT in its calculations and thus the only consequence of not covering those branches is that it might segfault or obtain incorrect data. A review of the source code does not indicate there would be any such problem and given no anomalous results were generated by recent usage of RCT on EBR-II with an updated ISOTXS file and model that uses the isotopic chi treatment, one can ignore these issues at this time.

The cited branch for NRASS=1 in subroutine rdgeo.f pertains to a model being constructed in REBUS that does not use coarse-mesh mapping to regions. It is very difficult for a user to naturally create such a problem through the DIF3D/REBUS interface and while it should be tested, it should not be a great concern as the only difference in source coding is how the MR array in GEODST is handled.

The cited LABELS treatment issues in subroutine rdstak.f are interesting but again unlikely to be of concern in RCT. From inspection of the RCT source code, there is no treatment available for the polynomial dependent cross sections. So while the REBUS calculation might use interpolated cross sections (not done for EBR-II), the RCT is incapable of doing so. The various untreated aspects of LABELS are simply not going to occur for the EBR-II models except for maybe the moving control rods and since those aspects are not relevant to the actual RCT calculation not reading them in correctly is of little importance. It is important to note that the GEODST files at each time point that RCT relies upon will have the correct control rod positioning details.

One code coverage detail in Table 3.1.2 that is important to resolve is the lack of coverage due to 120 periodic and full core models. Clearly those models need to be added to the testing as all EBR-II problems are full core. Reviewing the source code, there is virtually nothing to worry about as most of the branching just changes the number of sectors that are used when searching for matching region names.

A second code coverage aspect that needs to be resolved is the dependence of the testing on the number of active isotopes (NACI). For whatever reason, the developers of RCT choose to unroll

the various loops in the code to allow for vectorization. This is an artifact of older compiler methodologies and is considered very unwise today. To explain, instead or programming a loop to go from 1:10 over a set of instructions (e.g. I=1,10 for operation VecA(I)=10.0*VecB(I)), the developers wrote the loop to go 1:10 stepping by 4. This forces the developer to write the loop twice. In the first loop, the instruction would be duplicated four times (e.g. VecA(I)=,VecA(I+1)=,VecA(I+2)=,VecA(I+3)=). In the second loop the parts that the first loop does not cover must be handled (in the example it would be positions 9 and 10). That second loop of course is identical to the one needed for the case when no stepping is used (or stepping by 1). In the uncovered sections of various subroutines, those follow-on loops are not touched because the number of active isotopes in the problem being executed is a perfect multiple of 2 and 4. Further code coverage is wise although it would be preferable to simply eliminate the unrolling of the loops to simplify the coding.

A final code coverage aspect in Table 3.1.2 that should be resolved is the fixed format reading branches for the RCT input are not covered. That is because all of the examples use free format inputs. It should be a minor effort to modify some of the existing test cases to use a fixed format input for A.RCT instead of a free format.

From the preceding code coverage assessment, one can conclude that most of the critical parts of the RCT code are tested. While some additional coverage would be wise, those aspects should be of no importance with regard to the actual results generated by RCT.

Table 3.1.1 Covered or Partially Covered Driver RCT Files in the Directory *source*

File Name	Subroutine	# of calls by the test package	Coverage Status	Uncovered portion description	Further Coverage needed?
0gotows.F	GOTOWS	198	Partially	Fatal error and branch for alternative output file	No
0rct1.F	RCT1	22	Partially	Fatal error and branch for older version of REBUS	No
0rct2.F	RCT2	22	Partially	Fatal error and branch for alternative output file	No
0rct3.F	RCT3	22	Partially	Fatal error and branch for alternative output file	No
0stp034.F		22	Fully		No

Table 3.1.2 Covered or Partially Covered Files in the Directory *source*

File Name	Subroutine	# of calls by the test package	Coverage Status	Uncovered portion description	Further Coverage needed?
arct06.f	ARCT06	22	Partially	Fatal errors, fixed format reading, bad input corrections	No
arct07.f	ARCT07	2	Partially	Fatal errors and fixed format reading	No
arct08.f	ARCT08	2	Partially	Fatal errors and fixed format reading	No
arct09.f	ARCT09	2	Partially	Fatal errors and fixed format reading	No
arct0.f	ARCT0	22	Fully	Fatal errors and fixed format reading	No
assden.f	ASSDEN	328	Fully		No
assflu.f	ASSFLU	328	Fully		No
bocden.f	BOCDEN	66	Partially	Branching for inputs that likely cannot be created	No
bunnum.f	BUNNUM	38,640	Fully		No
chkbcd.f	CHKBCD	22	Partially	Fatal errors and card type 7,8,9 setup of the special pin input option	Yes
comp4d.f	COMP4D	2,134	Partially	Total scattering data provided on ISOTXS instead of elastic, inelastic, n2n.	No
comp7d.f	COMP7D	2,134	Partially	Branch for blocked scattering data in COMPXS	No
cpyf.f	CPYF	3,952	Partially	Branch for blocked data on STACK	No
dacon.f	DACON	1,312	Fully		No
dcymtx.f	DCYMTX	81	Fully		No

dendcy.f	DENDCY	2,316	Fully		No
eocden.f	EOCDEN	66	Fully		No
getik.f	GETIK	132	Partially	Branch for 120 periodic and full core models	Yes
hexreg.f	HEXREG	44	Partially	Fatal errors and branch for 120 periodic and full core models	Yes
isoiso.f	ISOISO	22	Partially	File chi and isotopic chi data reading of ISOTXS	No
mesh.f	MESH	66	Fully		No
poly13.f	POLY13	66	Fully		No
poly19.f	POLY19	66	Fully		No
pwpeak.f	PWPEAK	656	Fully		No
rct20.f	RCT20	14	Fully		No
	RCT20_PRINT	0	Uncovered	Debug print routine called from rct20	No
rct30.f	RCT30	22	Fully		No
rctip6.f	RCTIP6	22	Partially	Fatal Errors	No
rctip7.f	RCTIP7	22	Partially	Fatal Errors and Output pagination control	No
rctip8.f	RCTIP8	22	Partially	Fatal Errors and Special setup of Special Pins	Yes
rctip9.f	RCTIP9	22	Partially	Fatal Errors and special output branch	No
rdcxs.f	RDCXS	28	Partially	Zone chi matrix reading of COMPXS	No
rdgeo.f	RDGEO	44	Partially	Branch for NRASS=1 GEODST setup	No
rdstak.f	RDSTAK	5	Partially	Fatal errors, polynomial XS treatments and other features of LABELS not typically used	No
rdzna.f	RDZNA	44	Partially	Debug print output branch	No
recont.f	RECONT	656	Partially	Debug print output branch and card type 7,8,9 special pin input setups	Yes
rnhfly.f	RNHFLY	28	Partially	Debug print output	No
spinpw.f	SPINPW	656	Partially	Number of active isotopes is not an exact multiple of 2	Yes
spnden.f	SPNDEN	984	Fully		No
spnflu.f	SPNFLU	984	Fully		No
srpeak.f	SRPEAK	15,744	Fully		No
udecay.f	UDECAY	38,721	Partially	Case where number of active isotopes is not an exact multiple of 4	Yes
wrctdn.f	WRCTDN	22	Partially	Case with record types 5 and 6 in RCTDEN	Yes
wrctfx.f	WRCTFX	22	Fully		No

wrctpw.f	WRCTPW	22	Fully		No
wrflux.f	WRFLUX	656	Fully		No
xpower.f	XPOWER	132	Fully		No

4. DIF3D and REBUS Coverage Update

The preceding RCT subroutine coverage is the primary focus on this work. Because some other parts of the code system were used by RCT, primarily those associated with DIF3D, the code coverage is reiterated here.

4.1 Source code in the directory rebus/source

Only the savprg.f subroutine of REBUS is used by RCT and no change in the coverage occurs between it and that seen in the REBUS testing.

4.2 Source code in the directory rebus/dif3d/source

No subroutines in *rebus/dif3d/source* are included in RCT and thus no code coverage data is included here.

4.3 Source code in the directory rebus/dif3d/variantbasis/source

No subroutines in *rebus/dif3d/variantbasis/source* are included in RCT and thus no code coverage data is included here.

4.4 Source code in the directory rebus/dif3d/arcmodules/source

The Fortran coding in the directory *rebus/dif3d/arcmodules/source* of the RCT distribution primarily provides self-contained memory allocation and reading/writing of the binary interface files created in the ARC suite of codes. The RCT code uses the GEODST, NHFLUX, COMPXS, and NE_Kind modules in that directory. Reviewing the source code of RCT, these functions are only used when the DIF3D-VARIANT option of DIF3D is used in the REBUS model. The gprof compilation of the rct code failed to report the number of calls made to these modules, but the code coverage details were provided for all files and are detailed in Table 4.4.1.

Table 4.4.1 Covered or Partially Covered Files in the Directory rebus/*dif3d/arcmodules/source*

File Name	Subroutine	Coverage Details
COMPXS.F90	Multiple	Only uses the define, zero, and void functions and fully covers except for fatal error branches and delay cross section data inclusion
COMPXS_IMPORT.F90		Does not cover delay cross section data inclusion, does not cover zone chi treatment
GEODST.F90	Multiple	Only uses the define and void functions and fully covers except for fatal errors
GEODST_IMPORT.F90		Does not cover the ascii read subroutine, only the binary
GEODST_IMPORT_BINARY.F90		Does not cover 1D or 2D domain options, fatal error branches or NRASS=1 branch
NE_Kind.F90	Multiple	Only uses GetFreeLogicalUnit and UpperCaseString
NHFLUX.F90	Multiple	Only touches the define, void, and intel_ideal_open functions. Fatal error branches are missed.
NHFLUX_IMPORT.F90		Only misses the fatal error and debugging output branches

There are no concerning code coverage aspects of these subroutines that are concerning as PERSENT, GAMSRC, and DASSH will more heavily test out all of these modules and subroutines.

4.5 Source code in the directory rebus/dif3d/gnip4c_hmg4c/source

The directory *rebus/dif3d/gnip4c_hmg4c/source* in the RCT distribution contains the GNIP4C and HMG4C submodules of DIF3D. The GNIP4C program imports the A.NIP3 input data and packages it into the GEODST, NDXSRF, ZNATDN, LABELS, ISOTXS, and FIXSRC interface files. The HMG4C program generates the COMPXS file given the presence of GEODST, NDXSRF, ZNATDN, LABELS and ISOTXS files. The coverage summary of those subroutines identified with partial coverage in DIF3D and REBUS is shown in Table 4.5.1.

Many of the previously partially or fully covered subroutines in Table 4.5.1 are not covered by the RCT test suite. In fact, most of the covered parts of the module only correspond to the HMG4C usage in RCT. Because DIF3D and REBUS cover these subroutines more completely, the code coverage assessment for RCT is of little importance.

Table 4.5.1 Covered or Partially Covered Files in the Directory *dif3d/gnip4c_hmg4c/source*

File Name	Subroutine	# of calls by the DIF3D tests	# of calls by the REBUS tests	# of calls by the RCT tests
ads.f	ADS	4	0	0
ads1.f	ADS1	3	0	0
ads2.f	ADS2	37	0	0
ads3.f	ADS3	1	0	0
anip01.f	ANIP01	87	86	0
anip02.f	ANIP02	87	86	0
anip03.f	ANIP03	75	86	0
anip04.f	ANIP04	75	86	0
anip05.f	ANIP05	29	16	0
anip06.f	ANIP06	34	18	0
anip07.f	ANIP07	23	84	0
anip09.f	ANIP09	57	82	0
anip10.f	ANIP10	11	0	0
anip11.f	ANIP11	11	0	0
anip12.f	ANIP12	3	10	0
anip13.f	ANIP13	4	57	0
anip14.f	ANIP14	147	172	0
anip15.f	ANIP15	75	86	0
anip19.f	ANIP19	12	0	0
anip21.f	ANIP21	3	4	0
anip22.f	ANIP22	3	4	0
anip23.f	ANIP23	2	8	0
anip24.f	ANIP24	1	0	0
anip25.f	ANIP25	1	0	0
anip34.f	ANIP34	8	2	0
anip35.f	ANIP35	4	90	0
anip37.f	ANIP37	4	90	0
anip39.f	ANIP39	2	0	0
anip40.f	ANIP40	7	0	0
anip41.f	ANIP41	7	0	0

anip42.f	ANIP42	7	0	0
anip43.f	ANIP43	59	85	0
anip44.f	ANIP44	1	4	0
aniphx.f	ANIPHX	41	68	0
atdn3.f	ATDN3	77	1,634	44
bcdxst.f	BCDXST	87	86	0
bleep.f	BLEEP	233	19,150	0
chek09.f	CHEK09	98	104	0
cmesh.f	CMESH	106	104	0
copier.f	COPIER	4	90	0
cpylbg.f	CPYLBG	5	4	0
domods.f	DOMODS	4	90	0
edfpws.f	EDFPWS	9	0	0
edsorc.f	EDSORC	12	0	0
edsrch.f	EDSRCH	6	8	0
edtiso.f	EDTISO	1	0	0
edtmat.f	EDTMAT	4	57	0
edtr5.f	EDTR5	2	0	0
edtr6.f	EDTR6	2	0	0
edtr8.f	EDTR8	2	0	0
edtrod.f	EDTROD	3	11	0
edtxs1.f	EDTXS1	9	0	0
edtxs2.f	EDTXS2	60	0	0
egeods.f	EGEODS	59	85	0
fadens.f	FADENS	72	86	0
farset.f	FARSET	39,149	1,559,136	38,412
fgeods.f	FGEODS	75	86	0
fispec.f	FISPEC	77	1,634	44
fmesh.f	FMESH	106	104	0
getbc.f	GETBC	75	86	0
getbuc.f	GETBUC	75	86	0
gethgt.f	GETHGT	3	10	0
getiso.f	GETISO	72	86	0
getmat.f	GETMAT	4	57	0
getmr.f	GETMR	150	172	0
getmsh.f	GETMSH	75	86	0
getreg.f	GETREG	75	86	0
getset.f	GETSET	72	86	0
getszn.f	GETSZN	147	172	0
getzon.f	GETZON	147	172	0
gnip4c.F	GNIP4C	87	86	0
hexpc1.f	HEXPC1	594	1,000	0
hexpc2.f	HEXPC2	548	1,026	0
hexpc3.f	HEXPC3	1,042	1,907	0
hexpc4.f	HEXPC4	614	1,216	0
hexpc6.f	HEXPC6	509	915	0
hexpc7.f	HEXPC7	8	30	0
hexpic.f	HEXPIC	9	45	0
hmg4c.F	HMG4C	92	1,634	44
idlr13.f	IDLR13	77	1,634	44
indexx.f	INDEXX	7	4	0
isor14.f	ISOR14	77	1,634	44
isor58.f	ISOR58	77	1,634	44

lochex.f	LOCHEX	11,361	5,264	0
makmsh.f	MAKMSH	59	85	0
makrod.f	MAKROD	1	4	0
maxbnd.f	MAXBND	77	1,634	44
mpgeod.f	MPGEOD	16	26	0
mshmap.f	MSHMAP	22	26	0
ovl1.f	OVL1	77	1,634	44
ovl2.f	OVL2	77	1,634	44
ovl3.f	OVL3	77	1,634	44
ovl4.f	OVL4	9	0	0
ovl5.f	OVL5	77	1,634	44
prgeod.f	PRGEOD	59	85	0
prntix.f	PRNTIX	239	457	0
ranip1.f	RANIP1	75	86	0
ranip2.f	RANIP2	72	86	0
rcmesh.f	RCMESH	106	58	0
rdatdn.f	RDATDN	77	1,634	44
rdlabl.f	RDLABL	77	1,634	44
rdndx.f	RDNDX	77	1,634	44
redmsh.f	REDMSH	41	68	0
rwisot.f	RWISOT	56	84	0
set2d.f	SET2D	14	8	0
sethex.f	SETHEX	300,823	204,522	0
sordat.f	SORDAT	12	0	0
sordif.f	SORDIF	2	0	0
sorlab.f	SORLAB	7	0	0
sormsh.f	SORMSH	12	0	0
sornzn.f	SORNZN	6	0	0
sorrod.f	SORROD	1	4	0
sort13.f	SORT13	4	57	0
sort14.f	SORT14	72	86	0
srch3d.f	SRCH3D	1	0	0
srch4d.f	SRCH4D	1	4	0
srch6d.f	SRCH6D	1	0	0
srflt.f	SRFLT	5,436	6,891	0
svscat.f	SVSCAT	39,149	1,559,136	38,412
svxs.f	SVXS	166	0	0
trigom.f	TRIGOM	386	402	0
triplt.f	TRIPLT	3	6	0
update.f	UPDATE	638	58,049	4,268
valu2d.f	VALU2D	333	7,235	0
volreg.f	VOLREG	246	230	0
wndxsr.f	WNDXSR	72	86	0
wrec1.f	WREC1	77	1,634	44
wrec2.f	WREC2	77	1,634	44
wrec3.f	WREC3	460	37,078	5,016
wrec4.f	WREC4	460	37,078	5,016
wrrods.f	WRRODS	1	4	0
wrs.f	WRS	12	0	0
wrs0.f	WRS0	140	0	0
wrs1.f	WRS1	68	0	0
wrs2.f	WRS2	75	0	0
wrsorc.f	WRSORC	87	86	0

wrsrch.f	WRSRCH	87	86	0
wrtrod.f	WRTROD	1	4	0

4.6 Source code in the directory rebus/dif3d/syslib/source

The directory *rebus/dif3d/syslib/source* in the RCT distribution is one of the major libraries of DIF3D which contains BPOINTER memory management and file I/O among other character manipulation functions. The full and partially covered files are listed in Table 4.6.1 with just the updated call information for RCT. As can be seen, a majority of the use of RCT is consistent with usage in DIF3D and there is no reason to believe that RCT covers something that DIF3D or REBUS did not cover. The uncovered files in this source directory were also uncovered in RCT and thus they are not included here.

Table 4.6.1 Covered or Partially Covered Files in the Directory *rebus/dif3d/syslib/source*

File Name	Subroutines	# of calls by the DIF3D tests	# of calls by the REBUS tests	# of calls by the RCT tests
cequat.f	CEQUAT	1,104	12,452	0
chrflt.f	CHRFLT	286,758	2,146,552	68,526
	CFCOPY	286,758	0	0
chrset.f	CHRSET	385	2,574	0
cred.f	CRED	60	0	0
dopc.f	DOPC	1,394	20,186	110
	DOPDEL	83	207	0
	DOPFIL	83		0
	DOPLOK	83		0
	DOPSET	449	8,020	88
dred.f	DRED	30,127	359,591	0
	DREAD	30,127		0
	DRED_DOIT	0	0	0
error.F	ERROR	1,945	28,873	693
fequat.f	FEQUAT	0	2,503,256	2,481
fform.f	FFORM	3,772	20,477	982
	FFORM1	3,772	0	0
	FFORM2	15,013	81,217	6,322
fltchr.f	FLTCHR	1,849,042	32,732,597	499,196
	FCCOPY	1,849,042		0
fltset.f	FLTSET	168,442	1,627,263	10,034
iequat.f	IEQUAT	0	2,966,258	8,150
igtlcm.F	IGTLCM	178	6,834	66
	JGT	178		352
	FRELCM	122	6,748	66
	LOCFLC	0	0	0
	LOCFWD	1,294	25,406	308
	LOCF	1,188	28,098	594
intset.f	INTSET	6,695	55,922	154
lines.f	LINES	73,023	768,269	78,268
	PGCNTR	73,023	0	0
linkmd.F	LINKMD	112	0	0
	FINOUT	0	0	0
lunref.f	LUNREF	58,179	2,821,427	122,789
	SEKAPP	0	2,088	0

makddn.f	MAKDDN	4,899	11,645	902
msyncw.F	MSYNCW	1,003	7,020	154
pointr.F	POINTR	396	9,366	132
	PUTPNT	0	0	0
	BULK	396	9,366	132
	ECMREF	0	4,992	66
	BPFREE	396	9,449	132
	WIPOUT	25,078	388,460	6,886
	GETPNT	40,576		0
	IGET	40,576	711,858	5,808
	IPT2	26,117	577,790	5,852
	PUTM	22,181	389,733	1,892
	IPTERR	7,259	138,353	704
	ILAST	3,242	29,018	264
	REDEF	712	867	0
	REDEFM	377	1,900	0
	PURGE	4,204	43,711	220
	STATIS	0	3,912	132
	STATIS_DOIT	0	0	0
	PRTI1	0	0	0
	PRTI2	0	0	0
	PRTR1	0	0	0
	PRTR2	0	0	0
	PRTECM	0	0	0
reed.f	REED	20,260	1,375,940	61,947
	RREAD	20,260		0
seek.f	SEEK	9,147	202,308	3,326
sekphl.f	SEKPHL	3,027	23,923	264
space.f	SPACE	6,676	8,063	44
squeze.f	SQUEZE	2,204	9,507	164
srlab.f	SRLAB	1,596,408	27,247,001	428,269
timer.F	TIMER	12,278	143,300	726
	DSECOND	12,278	0	0
	TLEFT	0	0	0

4.7 Source code in the directory rebus/dif3d/lapack/source and *rebus/dif3d/linpack/source*

The files in directory *rebus/dif3d/lapack/source* and *rebus/dif3d/linpack/source* in the RCT distribution were not created as part of the DIF3D, REBUS, or RCT development. While they contain some useful capabilities for linear algebra manipulations, they are not used by RCT and thus no coverage details are given here.

4.8 Source code in the directory rebus/dif3d/seglib/source

The files in directory *rebus/dif3d/seglib/source* of the RCT distribution were developed as a common location separate from SYSLIB for a variety of ARC codes. It primarily contains the SUMMAR and RODPOS subprograms of ARC. SUMMAR produces isotopic reaction rate summary data for a given DIF3D input while RODPOS is used by REBUS to adjust the control rod positions at each time step. The subroutines with full or partial code coverage are listed in Table 4.8.1 and only the updated call information for the RCT tests is provided. As seen, only a single

function is used from this library which is associated with the HMG4C usage in RCT. The previously uncovered files are not used by RCT and thus those details are not included here.

Table 4.8.1 Covered or Partially Covered Files in the Directory *rebus/dif3d/seglib/source*

File Name	Subroutine	# of calls by the DIF3D tests	# of calls by the REBUS tests	# of calls by the RCT tests
adarea.f	ADAREA	9	766	0
addiso.f	ADDISO	11	769	0
bleep.f	BLEEP	233	19,150	0
calc.f	CALC	11	772	0
codecd.f	CODECD	206,916	569,424	2,679
modrec.f	MODREC	20	2,233	0
preplt.F	PREPLT	11	88	0
prnsum.f	PRNSUM	11	772	0
rdcmin.f	RDCMIN	11	772	0
rdfrpn.f	RDFRPN	11	772	0
rdgeom.f	RDGEOM	11	772	0
rdmixr.f	RDMIXR	11	772	0
rdmr.f	RDMR	23	2,193	0
rdoutm.f	RDOUTM	11	772	0
reacto.f	REACTO	11	772	0
scan.F	SCAN	56	86	22
stmesh.f	STMESH	28	2,260	0
stuff.F	STUFF	199	86	22
	STUFF1	88	86	22
summar.F	SUMMAR	11	772	0
sumre1.f	SUMRE1	11	772	0
udoit1.F	UDOIT1	87	0	0
udoit2.F	UDOIT2	97	88	0
udoit3.F	UDOIT3	87	0	0
udoit4.F	UDOIT4	87	86	0
xsrd.f	XSRD	11	772	0

5. Summary and Conclusions

In the preceding analysis the coverage report for each submodule of RCT was given. The submodule refers to separable libraries (LINPACK, LAPACK, G4CH4C, etc.) that are included either partially or fully as part of RCT (and DIF3D or REBUS) even though the development of the source coding was not specific to RCT. For the RCT specific source code, most of the subroutines were fully or partially covered to the point where additional testing is not necessary.

Three primary cases in RCT were identified that need additional testing. The first is that the existing test suite only uses a 60 degree periodic geometry model and thus branching for 120 periodic and full core models are not tested. There are also some branches that are not tested because of loop unrolling scheme taken by the authors of RCT. Finally, the fixed format reading of the RCT input is not tested and new test cases should be added for this specific purpose.

No meaningfully important code coverage details were found for all of the connected modules (SYSLIB, SEGLIB, etc…) of RCT beyond that which was identified for DIF3D and REBUS.

REFERENCES

[1] R. P. Hosteny, "The ARC System Fuel Cycle Analysis Capability, REBUS-2," ANL-7721, Argonne National Laboratory (1978).

[2] B. J. Toppel, "A User's Guide to the REBUS-3 Fuel Cycle Analysis Capability," ANL-83-2, Argonne National Laboratory (1983).

[3] W. S. Yang, M. A. Smith, "Theory Manual for the Fuel Cycle Analysis Code REBUS," ANL/NE-19/21, Argonne National Laboratory (2020).

[4] W. S. Yang, M. A. Smith, "RCT: REBUS Based Pin Power Reconstruction Using the DIF3D-Nodal and DIF3D-VARIANT Options," ANL/NE-14/15, Argonne National Laboratory (2014).

[5] K. L. Derstine, "DIF3D: A Code to Solve One-, Two-, and Three-Dimensional Finite-Difference Diffusion Theory Problems," ANL-82-64, Argonne National Laboratory (1984).

[6] R. D. Lawrence, "The DIF3D Nodal Neutronics Option for Two- and Three-Dimensional Diffusion Theory Calculations in Hexagonal Geometry," ANL-83-1, Argonne National Laboratory, March (1983).

[7] R. D. Lawrence, "Progress in Nodal Methods for the Solution of the Neutron Diffusion and Transport Equations," *Prog. Nucl. Energy*, **17**, 271 (1986).

[8] M. R. Wagner, "Three-Dimensional Nodal Diffusion and Transport Theory for Hexagonal-Z Geometry," *Nucl. Sci. Eng*., **103**, 377-391 (1989).

[9] G. Palmiotti, E. E. Lewis, and C. B. Carrico, "VARIANT: VARIational Anisotropic Nodal Transport for Multidimensional Cartesian and Hexagonal Geometry Calculation," ANL-95/40, Argonne National Laboratory (1995).

[10] M. A. Smith, E. E. Lewis, and E. R. Shemon, "DIF3D-VARIANT 12.0, A Decade of Updates," ANL/NE-14/1 Rev. 1, Argonne National Laboratory (2025).

[11] M. A. Smith and A. Nelson, "Verification of the REBUS Software," ANL-VTR-50 Rev. 2, Argonne National Laboratory (2025).

[12] M. A. Smith and A. Nelson, "Requirements Description of the REBUS Software for the Versatile Test Reactor," ANL-VTR-38, Argonne National Laboratory (2021).

[13] A. Nelson and M. A. Smith, "Verification of the DIF3D Software to Support Fast Reactor Analysis," ANL/NSE-20/3, Argonne National Laboratory (2020).

[14] M. A. Smith, A. Nelson, and F. Heidet, "Requirements Description of the DIF3D Software for the Versatile Test Reactor," ANL-VTR-19, Argonne National Laboratory (2019).

[15] Z. Zhong, M. A. Smith, C. H. Lee, "Code Coverage Status of the ARC Code DIF3D," ANL/NSE-23/65 Rev. 1, Argonne National Laboratory (2025).

[16] Z. Zhong, M. A. Smith, C. H. Lee, "Code Coverage Status of the ARC Code DIF3D," ANL/NSE-23/65 Rev. 1, Argonne National Laboratory (2025).

[17] Micheal. A. Smith and Zhaopeng Zhong, "Code Coverage Status of the ARC Code REBUS," ANL/NSE-25/36, Argonne National Laboratory (2025).

[18] K. Kiesling and M. A. Smith, “Software Quality Assurance Plan,” ANL/NSE-23/13, Argonne National Laboratory (2023).

Nuclear Science and Engineering Division
Argonne National Laboratory
9700 South Cass Avenue, Bldg. 208
Argonne, IL 60439-4842

www.anl.gov