RadSim v4.0 Test Log - Appendix to RadSim v4 Test Plan (NWPSAF-MO-TV-052)

This document summarises the results of testing performed on the RadSim v4.0 code. The test scenarios are described fully in the RadSim v4 Test Plan.

0.a Portability testing:
RadSim is compiled with ifort (on Intel and Cray), gfortran, pgfortran, and nagfor. This includes OpenMP-enabled (optimised), non-OpenMP (optimised), and debugging compiler flags. It is expected that users will compile with optimised compiler flags, and usually with OpenMP support. All builds are against RTTOV v14.0. Note that the nagfor builds omit ecCodes as there have been technical issues compiling ecCodes within this compiler (this compiler was not tested for previous releases as ecCodes was previously a mandatory dependency). These builds therefore test the new optional dependence on ecCodes and for these builds all tests involving GRIB model data files are omitted. The ifort builds are compiled against MKL for LAPACK routines to test the optional external LAPACK dependency.

0.b Run *radsim_check_install*:
Users should run this script after compilation to test their installation. It runs RadSim for a single instrument and compares the output to a reference file included in the package. This is run for all compilers tested.

The following tests correspond to the test scenarios listed in the Test Plan.

1. General validation test

The tests run as part of this test scenario are described in the Test Plan. They cover most of the core functionalities in RadSim. All outputs are validated against external RTTOV simulations run through the RTTOV *pyrttov* wrapper (where this is possible). Small differences are expected: these result primarily from RadSim using single precision to store certain types of data. The Python wrapper does not support PC-RTTOV, so these outputs are compared against equivalent "classical" RTTOV runs to ensure the differences are within expected tolerances, though they are certainly not expected to be identical in this case.

This test is performed for all compilers. This includes a single-threaded run. Other runs use multiple threads for RTTOV (via OpenMP) and for non-debug flags, multiple threads for RadSim (only relevant for footprint simulations). The outputs from all compilers are cross compared to ensure any differences are within expected tolerances. This therefore covers test scenario 2 below (OpenMP test).

2. OpenMP test

This falls under test scenario 1 above: the general validation test script is run for multiple compilers for both single-threaded and OpenMP cases, and the results are compared as part of the validation of test scenario 1.

3. Footprints test

This test scenario involves running RadSim in various configurations and comparing the outputs (as described in the Test Plan) to validate the footprint simulation capability. Success is indicated by any reported differences being within expected tolerances.

4. Satellite and solar angles validation

As described in the Test Plan this scenario compares the RadSim calculations for satellite and solar zenith and azimuth angles with those from the Met Office operational Satellite

Processing System. Success is indicated by reported/plotted differences being within expected tolerances.

5. Misc options test

This tests other RadSim capabilities not tested elsewhere. Currently two features are tested:
- the *use_all_atlas_months* option
- specification of SSU PMC CO2 cell pressures

The test script reports success or failure for each feature tested.

6. Comparison to RadSim v3.2

Test scenario 1 is run for the previous version and the current version. The simulations are configured to be as consistent as possible. Differences are compared to ensure they are within expected tolerances. As RTTOV v14.0 does not replicate outputs from RTTOV v13.2, we expect larger differences than usual for this test. They are compared with known differences between the two RTTOV versions to ensure they are consistent with what is expected.

7. Batching test

This test scenario compares output from a run with no batching to that from the same run with artificially small batch thresholds to ensure that outputs are identical.

8. Heavy load test

This test scenario involves running RadSim for two cases which involve large amounts of data to see whether RadSim copes. The first is for clear-sky SEVIRI simulations (all channels) with around 10^7 observations (corresponding to a full disc scan): this is a large number of input observations. The second is for clear-sky IASI simulations (all channels) on a global model grid at 0.5 degree lat/lon resolution (around 2.6×10^5 grid points) with emissivity output enabled: this test involves a very large amount of output data. Success is indicated by the RadSim runs completing successfully.

9. Valgrind/memcheck test

In this test scenario the general validation test (test scenario 1) is run for a non-OpenMP build with the Valgrind memcheck tool (ICON, HARMONIE, and JMA model data tests are excluded as GRIB input is covered by the ECMWF tests). Output is examined to ensure there are no memory leaks or other issues. NB Valgrind reports a large number of allocations that are “still reachable” for calls to the ecCodes library for tests involving ingest of GRIB data: these are not related to RadSim and are not necessarily considered problematic (see the Valgrind documentation).

10. Timing tests

These tests are run using an optimised build with a single thread for the RTTOV simulations. The model grid here comprises ~250000 profiles, and the observation data files define ~126000 profiles. These results give an indication of the run-times one can expect from RadSim. The differences in run-times between versions is dominated by those in the corresponding RTTOV versions (see the RTTOV Performance Test Log). Regarding cases 7 and 8, the footprint simulation code in RadSim has not changed so the longer run-times for v4.0 (especially case 8) are most likely due to noise on the testing system.

Test	Date	Platform	Compiler	Compiler flags	Pass/fail	Notes
0.a Compilation	28/03/2025	x86_64 GNU/Linux	ifort v17.0.1	ifort (with OpenMP)	Pass	NetCDF v4.1.1 ecCodes v2.24.0 with MKL for LAPACK
	28/03/2025	x86_64 GNU/Linux	ifort v17.0.1	ifort (debugging flags)	Pass	NetCDF v4.1.1 ecCodes v2.24.0 with MKL for LAPACK
	28/03/2025	x86_64 GNU/Linux	gfortran v11.2.0	gfortran (with OpenMP)	Pass	NetCDF v4.1.1 ecCodes v2.24.0
	28/03/2025	x86_64 GNU/Linux	gfortran v11.2.0	gfortran (debugging flags)	Pass	NetCDF v4.1.1 ecCodes v2.24.0
	28/03/2025	x86_64 GNU/Linux	gfortran v8.1.0	gfortran (with OpenMP)	Pass	NetCDF v4.1.1 ecCodes v2.24.0
	28/03/2025	x86_64 GNU/Linux	gfortran v8.1.0	gfortran (optimised flags without OpenMP)	Pass	NetCDF v4.1.1 ecCodes v2.24.0
	28/03/2025	x86_64 GNU/Linux	pgf90 v18.7-0	pgf90 (with OpenMP)	Pass	NetCDF v4.1.1 ecCodes v2.24.0
	28/03/2025	x86_64 GNU/Linux	nagfor v6.1	nagfor (with OpenMP)	Pass	NetCDF v4.1.1 No ecCodes
	28/03/2025	x86_64 GNU/Linux	nagfor v6.1	nagfor (debugging flags)	Pass	NetCDF v4.1.1 No ecCodes
	01/04/2025	Cray	cray-ifort v18.0.5	cray-ifort	Pass	NetCDF v4.3.2 ecCodes v2.6.0
0.b radsim_check_install	28/03/2025	x86_64 GNU/Linux	ifort v17.0.1	ifort (with OpenMP)	Pass	
	28/03/2025	x86_64 GNU/Linux	ifort v17.0.1	ifort (debugging flags)	Pass	
	28/03/2025	x86_64 GNU/Linux	gfortran v11.2.0	gfortran (with OpenMP)	Pass	
	28/03/2025	x86_64 GNU/Linux	gfortran v11.2.0	gfortran (debugging flags)	Pass	
	28/03/2025	x86_64 GNU/Linux	gfortran v8.1.0	gfortran (with OpenMP)	Pass	
	28/03/2025	x86_64 GNU/Linux	gfortran v8.1.0	gfortran (optimised flags without OpenMP)	Pass	
	28/03/2025	x86_64 GNU/Linux	pgf90 v18.7-0	pgf90 (with OpenMP)	Pass	
	28/03/2025	x86_64 GNU/Linux	nagfor v6.1	nagfor (with OpenMP)	Pass	

	28/03/2025	x86_64 GNU/Linux	nagfor v6.1	nagfor (debugging flags)	Pass	
	01/04/2025	Cray	cray-ifort v18.0.5	cray-ifort	Pass	

Test scenario	**Date**	**Platform**	**Compiler**	**Compiler flags**	**Pass/fail**	**Notes**
1. General validation test	31/03/2025	x86_64 GNU/Linux	ifort v17.0.1	ifort (with OpenMP)	Pass	
	31/03/2025	x86_64 GNU/Linux	ifort v17.0.1	ifort (debugging flags)	Pass	
	31/03/2025	x86_64 GNU/Linux	gfortran v11.2.0	gfortran (with OpenMP)	Pass	
	31/03/2025	x86_64 GNU/Linux	gfortran v11.2.0	gfortran (debugging flags)	Pass	
	31/03/2025	x86_64 GNU/Linux	gfortran v8.1.0	gfortran (with OpenMP)	Pass	
	31/03/2025	x86_64 GNU/Linux	gfortran v8.1.0	gfortran (optimised flags without OpenMP)	Pass	
	31/03/2025	x86_64 GNU/Linux	pgf90 v18.7-0	pgf90 (with OpenMP)	Pass	
	31/03/2025	x86_64 GNU/Linux	nagfor v6.1	nagfor (with OpenMP)	Pass	NB No tests involving model data in GRIB format are run for the nagfor builds.
	31/03/2025	x86_64 GNU/Linux	nagfor v6.1	nagfor (debugging flags)	Pass	NB No tests involving model data in GRIB format are run for the nagfor builds. The UM PP file ingest code referred to a dangling pointer which has not been a problem for other compilers in this or previous releases but caused a failure for this build. In addition, for UM fieldsfiles on a non-rotated grid, the pole lat/lon values were uninitialised and the output gave them as NaN (not a problem as they are unused in this case). The code was updated to avoid the reference to dangling pointer and to initialise the pole lat/lon in all cases. After this all tests pass.

	01/04/2025	Cray	cray-ifort v18.0.5	cray-ifort	Fail	The ICON tests with cloud liquid and ice Deff fields and the ICON-ART tests failed as the ecCodes library is too old. Since ifort builds on Linux are OK there is no reason to suspect they would not work on Cray with a newer ecCodes version. All other tests passed.
2. OpenMP test	31/03/2025	x86_64 GNU/Linux	gfortran v8.1.0	gfortran (optimised flags with/without OpenMP)	Pass	
3. Footprints test	01/04/2025	x86_64 GNU/Linux	gfortran v11.2.0	gfortran (with OpenMP)	Pass	
4. Satellite and solar angles	01/04/2025	x86_64 GNU/Linux	gfortran v11.2.0	gfortran (with OpenMP)	Pass	
5. Misc options test	01/04/2025	x86_64 GNU/Linux	gfortran v11.2.0	gfortran (with OpenMP)	Pass	
6. Comparison to RadSim v3.2	04/04/2025	x86_64 GNU/Linux	gfortran v11.2.0	gfortran (with OpenMP)	Pass	
7. Batching test	01/04/2025	x86_64 GNU/Linux	gfortran v11.2.0	gfortran (with OpenMP)	Pass	
8. Heavy load test	01/04/2025	x86_64 GNU/Linux	gfortran v11.2.0	gfortran (with OpenMP)	Pass	
9. Valgrind/memcheck test	31/03/205	x86_64 GNU/Linux	gfortran v8.1.0	gfortran (optimised flags without OpenMP)	Pass	

Test scenario 10 - Timing test

Test	RadSim v3.2 time (s)	RadSim v4.0 time (s)
1. ATMS clear-sky grid	173	110
2. ATMS cloudy grid	546	543
3. SEVIRI IR channels cloudy grid	812	830
4. SEVIRI IR channels cloudy obs	450	454
5. SEVIRI VIS/NIR channels cloudy obs (MFASIS-NN)	160	142
6. SEVIRI VIS/IR channels clear-sky obs	44	44
7. SEVIRI VIS/IR channels clear-sky obs with footprints (writing footprint data file)	1183	1248
8. SEVIRI VIS/IR channels clear-sky obs with footprints reading footprint data file	653	750