Radiance Simulator v3 Test Plan

James Hocking, Met Office, UK

This documentation was developed within the context of the EUMETSAT Satellite Application Facility on Numerical Weather Prediction (NWP SAF), under the Cooperation Agreement dated 7 December 2016, between EUMETSAT and the Met Office, UK, by one or more partners within the NWP SAF. The partners in the NWP SAF are the Met Office, ECMWF, DWD and Météo France.

Change record			
Version	Date	Author / changed by	Remarks
0.1	09/03/2021	J. Hocking	Initial version
0.2	31/03/2021	J. Hocking	Updates after internal review
0.3	26/04/2021	J. Hocking	Updates after DRR
1.0	18/01/2022	J. Hocking	Updates for v3.1

Table Of Contents

1 INTRODUCTION

This document defines the Test Plan for the NWP SAF Radiance Simulator in accordance with the guidelines set out in [RD-1]. The aim of the Test Plan is to verify that the requirements of the product specification in [RD-2] are met.

1.1 Reference documents

[RD-1]	NWPSAF-MO-SW-002: Development Procedures for Software Deliverables
[RD-2]	NWPSAF-MO-DS-041: Radiance Simulator Product Specification
[RD-3]	NWPSAF-MO-TV-051: Radiance Simulator User Guide

2 TESTED ITEMS

The following items are covered as defined in [RD-1]:

1. Coding
2. Module and Integration tests
3. Validation tests
4. Portability tests
5. Beta testing
6. User documentation

2.1 Coding

The majority of the code is written in Fortran-90 with the addition of some features from Fortran-2003. These additional features are supported by most modern compilers and have been approved for use in Met Office operational systems. The code is checked for conformance with the Fortran-2003 standard using compiler flags.

2.2 Module and Integration tests

The Radiance Simulator consists of a set of Fortran subroutines and modules which are compiled and linked to a single program unit. Independent module testing is not applicable in this case; all testing is performed as an integrated package.

A simple test case is included in the release package which runs RadSim for a single sensor in a basic configuration and the output is compared to reference data. This is used to ensure the compilation was successful and that RadSim runs successfully. This is performed for all compilers/platforms tested (see section on Portability below).

The primary integration testing is carried out using the first script described in the next section (Validation testing) which runs RadSim in a wide variety of configurations. This script is run for all compilers/platforms tested and ensures that the majority of RadSim options are tested across multiple platforms.

2.3 Validation tests

The purpose of the validation tests is to check that the main functions of the Radiance Simulator are performing correctly. As described in the Product Specification [RD-2], these are

- Ingest of NWP model fields (all formats)
- Ingest of observation data files
- Interpolation of model fields to observation positions and times
- Footprint simulation capability
- Setup and running of the radiative transfer model (RTTOV) for different sensors and simulation types (e.g. clear-sky, cloudy, Principal Components models, etc).
- Writing of results and associated data to an output file

The following test scenarios are performed (numbers given are test scenario IDs). Note that the various scripts referred to below are not included in the package distributed to users.

1. General validation test

   A shell script (*radsim_test.sh*) has been written which runs RadSim in multiple configurations in order to test the majority of the RadSim functionality. The script tests the following:
   - All input model types:
     - Met Office fieldsfiles and PPfiles
     - ECMWF GRIB and netCDF files
     - ECMWF CAMS GRIB file including aerosol fields
     - DWD ICON GRIB files
     - HARMONIE GRIB files
     - JMA GRIB files
     - all NWP SAF profile datasets
   - MW sensor, VIS/IR radiometer, hyperspectral IR sounder
   - Clear-sky simulations, cloudy simulations, CAMS aerosol simulations, PC-RTTOV, HTFRTC
   - Simulations with solar radiation with fixed and RadSim-calculated solar angles
   - Simulations with RadSim-calculated geostationary satellite angles
   - Use of modified CO_2 background profile
   - Use of RTTOV land surface emissivity and BRDF atlases
   - Simulations on the model grid
   - Simulations using observation data files
   - Temporal interpolation
   - Footprint simulations
   - RTTOV direct and Jacobian model calls
     - includes all Jacobians: T, q, O3, Tskin, wind u/v, surface emissivity.
   - CADS height assignments
   - Geometric height output

A full description of all tests performed by the script is given in the RadSim v3.0 and v3.1 Test Logs (a separate Annex to this document). This script is run for all compilers/platforms being tested. Results are cross-compared to ensure results are consistent allowing for small compiler-related differences. This comparison is performed by a Python script (*radsim_compare_output.py*) which compares all datasets of interest and allows for absolute and relative tolerances to be specified such that only larger differences are reported. The default absolute tolerance is 1E-16 and the default relative tolerance is 1E-4%. A small number of compilers report maximum differences which exceed the default tolerance (1E-4%) for a few tests: the differences are still very small (order 1E-3% at most) and are not considered problematic.

In addition, a Python script has been created (*radsim_rttov_compare.py*) which accepts a RadSim configuration namelist file and the corresponding RadSim output netCDF file containing the profile data as output by RadSim and the calculated radiances. The script runs the corresponding simulations separately in RTTOV (using the RTTOV Python wrapper - see below) and compares the outputs with those from RadSim to ensure differences are within expected tolerances (differences are primarily due to the lower precision used in certain places in RadSim). This ensures that RadSim is running the RTTOV simulations as intended. This script has proven invaluable for uncovering errors during RadSim development. The *radsim_test.sh* shell script described above automatically runs this Python validation script for all applicable tests.

Unfortunately, the RTTOV Python wrapper does not currently support PC-RTTOV or HTFRTC simulations. Instead, these are validated by comparing the output brightness temperatures to those from equivalent standard RTTOV simulations to ensure the differences are within expected tolerances using the *radsim_compare_output.py* script.

The footprint simulations also cannot be validated by this Python script because the necessary profile data are not output by RadSim for these simulations (since there are multiple profiles per observation). The footprint simulation outputs are validated in test scenario 3 described below.

Finally, the internal RadSim geostationary satellite angle calculation runs cannot be validated by this Python script either, and these are instead validated in test scenario 4 described below.

2. OpenMP test

   This is tested as part of test scenario 1 above. The general validation test is run with multiple threads using OpenMP builds of RadSim and RTTOV, and with a single thread using an optimised non-OpenMP build, and the results are compared to ensure they are identical (to within expected tolerances). This test is primarily intended to ensure that the OpenMP multi-threading in RadSim (for footprint simulations) is working correctly, but it also checks that multi-threaded RTTOV simulations are running correctly through RadSim.

3. Footprints test

   A separate test script (*radsim_test_footprints.sh*) has been created to validate the footprint calculation. This uses netCDF ECMWF data at a lat/lon resolution of 5 degrees (this is very coarse but keeps the test run-times short and memory requirements low while fully testing the functionality). Three obs data files are used:
   a) all model grid points with small footprints (1km/0.75km semi-major/minor axes)
   b) all model grid points with large footprints (500km/375km)
   c) one obs per grid box, lying off the grid points, with small footprints (1km/0.75km)
   d) mixture of grid and off-grid points with mixture of footprint sizes
   Each observation is assigned a random azimuth angle which defines the orientation of the footprint ellipse.

   The shell script runs RadSim in 9 configurations for RTTOV cloudy simulations for MSG-4 SEVIRI IR channels:
   1. On the model grid.
   2. Using obs file a) without footprints enabled.
   3. Using obs file a) with footprints enabled.
   4. Using obs file c) without footprints enabled.
   5. Using obs file c) with footprints enabled.
   6. Using obs file b) with footprints enabled and writing out radiances (rather than BTs).
   7. On the model grid writing out radiances.
   8. Using obs file d) with footprints enabled, writing out the footprint data file.
   9. Using obs file d) with footprints enabled, reading the footprint data file from 8.

   The script confirms that runs 1-3 give identical results (to within expected tolerances) and runs 4-5 give identical results (to within expected tolerances): these comparisons are made using the netCDF *nccmp* utility. A separate Python script (*radsim_footprint_validation.py*) is used to compute footprint-averaged radiances from run 7 and ensures they are identical (to within expected tolerances) to the RadSim results from run 6. Finally, outputs from runs 8 and 9 are compared using *nccmp* to ensure they are strictly identical. The last two runs are for RadSim v3.1+ only.

4. Satellite and solar angle calculation validation

   RadSim calculates satellite zenith and azimuth angles in the *radsim_geo_obs.py* script which generates obs datafiles for GEO sensors, and it calculates solar zenith and azimuth angles internally using the location and date/time of observations.

   These calculations are validated by comparing the computed angles to those calculated by the Met Office operational Satellite Processing System (SPS). This comparison is performed using the test script *radsim_test_angle_calculations.sh*. It requires NWP model fields and a “slotstore” file from SPS containing the latitudes, longitudes and satellite and solar angles on the UM model grid. The test script uses *radsim_geo_obs.py* to generate an obs datafile for the grid points, runs RadSim using this, and compares the angles from RadSim with those from SPS.

The solar angle calculation in RadSim follows that in SPS very closely and as such the differences in solar zenith angle should be very small (e.g. less than 0.2°). The differences are due primarily to the fact that RadSim operates at a resolution of minutes, while SPS performs calculation at a resolution of seconds. Differences in solar azimuth angles should generally be small, but at the subsolar point the angles are very sensitive to small differences in the calculation inputs, and so points near the sub-solar point will exhibit larger differences.

The satellite zenith angle calculation in RadSim differs slightly in certain assumptions to that in SPS and as such the differences are slightly larger than for the solar zenith angle but should still be well below 1°. Similar considerations apply to the azimuth angle, with differences generally being small, but larger differences again being observed close to the sub-satellite point.

The aim of this test is to ensure that the angle calculations are producing sensible values. This test is run for one compiler with OpenMP.

RadSim v3.1 also offers the option to compute geostationary satellite zenith and azimuth angles internally. These calculations are validated by running RadSim again for the same obs data file as above but omitting the satellite angles, and with the internal satellite angle calculations turned on. The output satellite zenith and azimuth angles are compared to those from the previous RadSim run. They should be very close, but differences in precision of the calculations mean the outputs are not strictly identical.

5. Miscellaneous options test

This test scenario is intended to check miscellaneous capabilities of RadSim. These are tested via a script *radsim_test_misc_options.sh*. Currently it runs two tests:

- test *use_all_atlas_months* option: simulations are run for an NWP SAF profile dataset with a MW sensor with this option set to true and false. The test is successful if the outputs differ.
- test specification of SSU PMC CO2 cell pressure specification: RadSim is run for non-PMC-shift SSU coefficients, and then for PMC-shift SSU coefficients with no cell pressures specified, with the default (nominal) cell pressures for the given sensor specified, and finally with modified cell pressure specified. The outputs of the first three of these simulations should be identical and the results of the final test should differ.

The script runs the simulations, compares the outputs, and reports success or failure as appropriate.

6. Comparisons to previous version

   Comparing RadSim v2.2 to RadSim v3.0:

   The intention behind this test scenario is to ensure RadSim v3.0 behaves in the same way as v2.2 in respect of capabilities implemented in both versions. RTTOV underwent a number of changes between v12.3 and v13.0 which must be taken into account in these comparisons: the values of some fundamental constants were updated, the default values of some run-time options were changed and some new options were added, and some ancillary input files were changed.

   However, RTTOV v13.0 can easily be compiled with the v12 values of the fundamental constants, and it is simple to modify RadSim v3.0 to specify relevant options to reproduce RTTOV v12.3 behaviour. This should yield near-identical results for most types of clear-sky simulations when using the same RTTOV optical depth coefficients. Scattering simulations give different results because optical properties were updated between RTTOV v12.3 and v13.0, and HTFRTC simulation outputs differ because the coefficients changed.

   A modified version of the *radsim_test.sh* script (test scenario 1) is used to run clear-sky simulations (excluding HTFRTC) in RadSim v2.2 and v3.0 for all NWP model inputs supported by v2.2 (i.e. HARMONIE and JMA GRIB data were excluded). The outputs are then compared using the *radsim_compare_output.py* script to ensure any differences are within expected tolerances.

   Comparing RadSim v3.0 to v3.1:

   This is much more straightforward. The *radsim_test.sh* script (test scenario 1) from RadSim v3.0 is run for both v3.0 and v3.1. All results should be identical with the exception of MFASIS simulations for which RTTOV v13.1 gives slightly different outputs for ice cloud (see the RTTOV documentation).

7. Batching test

   A script (*radsim_test_batching.sh*) has been created which performs a simple run for clear-sky ATMS simulations with 150000 random observations using model fields on a 1 degree lat/lon grid. All profiles are simulated in a single profile batch and all outputs are written in one output batch. The test is repeated with *max_profs* (maximum number of input profiles per batch) set to 70000 yielding three profile batches. The *max_array_size* (maximum size of data per output batch) is set to 1MB yielding 14 output batches. The two output files are compared using *nccmp* to ensure they are identical.

   Note that the batching features are not intended to be triggered by such small numbers of profiles, the default values are much higher, but this test is run to verify that batching does not in any way affect the results by doing comparisons with single batch runs.

8. Heavy load test

Two tests are run to see how the Radiance Simulator performs under extreme conditions. These tests are run with one compiler with OpenMP enabled: completion of the RadSim runs without errors implies success.

The first test is to run clear-sky SEVIRI simulations (all 12 channels) for a very large number of data points (of order 10000000, corresponding to a full disc scan as generated by the new geostationary observation Python script). This would normally result in an excessive memory requirement to hold the interpolated model data, but the Radiance Simulator batches the input profiles to reduce this.

The second test is to run clear-sky IASI simulations (all channels) on a global model grid at a lat/lon resolution of 0.5 degrees (roughly 2.6×10^5 grid points) with emissivity output switched on. This would normally result in an excessive memory requirement to hold the calculated output data (over 16GB), but the Radiance Simulator batches the processing to reduce this. Note that this is separate and in addition to the batching of input profiles, each profile batch may have multiple processing batches depending on output data requirements.

9. Valgrind/memcheck test

The purpose of this test is to check RadSim for memory leaks and other memory errors. The test is run for one compiler (typically gfortran) *without* OpenMP for both RadSim and RTTOV. RadSim is compiled with the *-g* flag to include debugging symbols so that Valgrind can report the source (line number) of memory leaks and other issues.

The *bin/radsim_run* script is edited to run valgrind when the RadSim executable is called. For example:

```
valgrind --leak-check=full --show-reachable=yes \
/full/path/to/radsim/bin/radsim.exe $RADSIM_NL
```

The general validation test script (test scenario 1) is then run (without calls to the RadSim/RTTOV validation script) and the resulting Valgrind/memcheck output is examined to ensure no memory leaks are reported. Note that calls to the ecCodes library result in a large number of allocations classed as “still reachable” which affect all tests involving ingest of GRIB data: these are not related to RadSim and do not necessarily indicate a problem (see the Valgrind documentation).

10. Timing tests

In most cases, the majority of the RadSim run-time is dominated by the calls to RTTOV, and since each version of RadSim is based on a different version of RTTOV, a comparison of RadSim run-times amounts to a comparison of RTTOV run-times. Each RTTOV release is compared to the previous one in terms of performance and the reports are available from the RTTOV web pages. However, some basic configurations are tested for consecutive versions (RadSim v2.2 vs v3.0, and v3.0 vs v3.1) and the total run-times are recorded. For v3.0, an indicative timing for a footprint simulation is given as these are significantly more expensive. For v3.1, the impact of the footprint writing/reading feature is demonstrated. Note that timings are given for single-threaded runs: in practice use of multiple threads via OpenMP is recommended and reduces run-time significantly.

The timing simulations use ECMWF GRIB data for the region bounded by latitudes 45N to 65N and longitudes 15W to 5E (covering the UK and Ireland) at a resolution of 0.04 degrees. The grid has 251001 points. The observation data file is generated using the new GEO obs data file tool and represents MSG-4 SEVIRI pixels over this same area (though the nadir footprint radius is doubled to 3km to ensure the footprints are not empty at this model grid resolution): this comprises 126178 observations. Tests are run for SNPP ATMS and MSG-4 SEVIRI.

Timing tests performed:
- clear sky grid, ATMS
- cloudy grid, ATMS
- cloudy grid, SEVIRI IR channels
- cloudy obs with temporal interpolation, SEVIRI IR channels
- clear sky obs with temp. interp., SEVIRI VIS/IR channels
- clear sky obs with temp. interp. and footprints, SEVIRI VIS/IR channels (v3.0+ only, with writing of footprint data file v3.1+ only)
- clear sky obs with temp. interp. and footprints, SEVIRI VIS/IR channels, reading footprint data file (v3.1+ only)

All tests are run for the RadSim versions under test except where indicated. The same RTTOV optical depth coefficients are used (v7/8/9 predictors for v2.2/v3.0 comparisons, v13 predictors for v3.0+ comparisons), and RadSim/RTTOV options are selected to ensure the comparisons are fair between each pair of versions.

This test is intended to give an idea of run-times for RadSim and whether they are expected to change between versions. Since the majority of the run-time comes from RTTOV, there are no criteria for success/failure.

2.4 Portability tests

The general validation test (test scenario 1) is repeated for each of the following compilers and platforms:

- *ifort* v*17.0.1* on x86_64 virtual desktop running Red Hat Enterprise Linux version 7
- *pgfortran v18.7.0* on x86_64 virtual desktop running RHEL7
- *gfortran v8.1.0 and v11.2.0* on x86_64 virtual desktop running RHEL7
- *ifort v15.0.0.090* on Cray XC40 running Cray Linux

Results for all tests are cross-checked for consistency. The other validation tests are performed for a single compiler except where noted otherwise above. The RadSim v3.0 and v3.1 Test Logs provides full details of all platforms and compiler flags tested.

2.5 Beta testing

RadSim v2 users were emailed a request for volunteer beta testers. Four users responded to this call. The aim is for beta testers to trial RadSim v3.0 in their existing application and assess how it compares to RadSim v2. They are also requested to test any new features in RadSim v3.0 which are relevant to their application. The aim is to assess RadSim v3.0 in terms of ease of use of the code and to verify that all main components of the code run successfully. In addition, testers are asked to critically review the User Guide and other supporting documentation. Where significant problems are identified, the tester will be asked to re-test the code after changes have been made to confirm satisfactory operation.

Answers to the following questions will be collected and recorded below.

1. Is the software installation straightforward?
2. What hardware platform, operating system, and compiler did you use to build the executable? What versions of external libraries were used? If you ran any of the Python scripts, which version of Python are you using?
3. Did you encounter any problems building the package on your system?
4. How easy was it to set up and run the code? Did you encounter any problems?
5. Do you think the results are acceptable? Where relevant, how do they compare to RadSim v2?
6. Is the User Guide [RD-3] clear and comprehensive?
7. Assess whether the overall package meets the requirements of the Product Specification [RD-2]
8. Do you have any suggestions for improvements?

The beta testers were also asked to provide an overview of what features of RadSim they used such as which type(s) of NWP model fields, what types of RTTOV simulations, etc.

2.6 User Documentation

The Radiance Simulator documentation will be reviewed in-house by one or more people not on the development team.

3 REQUIREMENTS TRACEABILITY MATRIX

This section demonstrates how the requirements listed in section 8 of the Product Specification relate to the test plan.

Table 1: Requirements traceability

Requirement	**Testing method**	**Test plan reference**	**Test Scenarios**
RQ-4.1 Ingest of model fields	Test run	2.3	1, 9
RQ-4.2 Ingest of obs data files	Test run	2.3	1, 9
RQ-4.3 Spatial and temporal interpolation	Test run	2.3	1, 9
RQ-4.4 Footprint simulations	Test run	2.3	1, 3, 9
RQ-4.5 Calculate solar angles	Test run	2.3	4
RQ-4.6 Prepare/run RTTOV	Test run	2.3	1, 9
RQ-4.7 Generate outputs	Test run	2.3	1, 5, 6, 9
RQ-4.8 GEO obs data file tool	Test run	2.3	4, 8
RQ-8.1 Functionality	See RQ-4.1 - RQ-4.7 above	2.3	See above
RQ-8.2 Compilation / user instructions	Test install, beta test	2.2, 2.3, 2.4, 2.5	1
RQ-8.3 Test scripts successful	Test install, beta test	2.2, 2.5	N/A
RQ-8.4 Multi-threading via OpenMP	Test run	2.3	2
RQ-8.5 Performance	Test run	2.3	10
RQ-8.6 Portability	Test install, beta test	2.2, 2.3, 2.4, 2.5	1
RQ 8.7 Resources	Test run	2.3	7, 8
RQ-8.8 Language	Inspection, compiler flags	2.1, 2.4	N/A
RQ-8.9 Documentation	Review, beta test	2.5, 2.6	N/A
RQ-8.10 Interface requirements	Test install, beta test	2.2, 2.4, 2.5	N/A

4 TEST RESULTS

4.1 Summary

The RadSim v3.0 and v3.1 Test Logs are provided as a separate Annex to this document. They contain details on the specific tests performed including:

- information on compilers and compiler flags
- versions of external libraries
- portability test results
- basic installation test results
- test results for all test scenarios described above
- timing test results

The testing overall was successful and did not indicate any issues with the package.

4.2 Beta Testing

Volunteer beta testers for RadSim v3.0 were solicited from RadSim v2 users. Four users expressed interest in beta testing RadSim, but in the end only two users provided any feedback after a 6-week beta test period.

Below are the (anonymised) email conversations with the beta testers who provided feedback. NWP SAF responses are in red.

4.2.1 Tester 1

Thanks for the software. We have installed RTTOV 13 and RadSim v3.0. […] is performing some tests, which compare the results of RadSim v3.0 to what we got with RadSim v2.3. […] and I would like to give you already a first feedback.

To your questions:

1. Is the software installation straightforward?

Yes. The installation was as easy as for the previous RadSim versions. We only had to tell radsim to use the lib64 directories instead of lib.

2. What hardware platform, operating system, and compiler did you use to build the executable? What versions of external libraries were used? If you ran any of the Python scripts, which version of Python are you using?

We are working on *Red Hat Enterprise Linux Server* V7.6 and compiled RadSim with gfortran.

- GNU Fortran (GCC) 4.8.5
- netCDF 4.6.2
- HDF5 Version 1.10.5 (I hope I extracted the correct version; this is taken from h5dump)

3. Did you encounter any problems building the package on your system?

No

4. How easy was it to set up and run the code? Did you encounter any problems?

No problems. We have embedded Radsim in a processing suite. The only thing we needed to modify was the namelist in the config file for RadSim. BUT we would like to mention 2 points:

1) we need to update the software to handle the JRA input (as described by an email from […])
2) the usage of the RTTOV coefficient v13 files is not easy. We need to explicitly provide the coefficient file name in the Configuration namelist file and not only the directory as before. This is related to added complexity to the filenames (now with O2_CO2, or IRONLY, etc.).

It would be good, if RadSim could automatically find the required coefficient file. This would make it easier to switch between instruments. One idea comes from […]:

In src/code/main/radsim_setup_rttov.F90:

Change *trim(platform) // '_', satid, '_' // trim(inst) // rttov_coeffs_type* to *trim(platform) // '_', satid, '_' // trim(inst) // rttov_coeffs_options // rttov_coeffs_type*

Then one need to specify the options (gases used etc.) in the Configuration namelist file

5. Do you think the results are acceptable? Where relevant, how do they compare to RadSim v2.0?

We are using RadSim with ERAInterim and ERA 5 profiles (grib files). We perform mostly clear sky simulations. We only provide one time step per grib file. Further, we did not use the python scripts. We currently compare the results of RadSim2 and RadSim3 via nccmp/1.8.6.2.

We compare simulations of SMMR, MHS, MVIRI (IR and WV; NO vis), HIRS.

(a) Using the old predictors, we see differences in brightness temperatures between 0.01K and 0.1K.

(b) The differences for the V13 coefficient files will be evaluated now and we will let you know soon about the results.

6. Is the User Guide clear and comprehensive?

7. Assess whether the overall package meets the requirements of the Product Specification.

8. Do you have any suggestions for improvements to the software or documentation?

I will update this feedback as soon as we have more results.

NWP SAF response:

Many thanks for the early feedback!

Regarding JRA-55 support: can you provide a sample GRIB file? I'm sorry that I didn't manage to get this into the beta version, but I could have a look now at implementing the basic support for this that you require based on […] code.

For the coefficient filenames, your suggestion is a good one and I will add this "rttov_coeffs_options" namelist variable. The idea behind the filenames on the website is to clearly distinguish between multiple files for a given sensor that users might download or that are included in the package. The filenames are not fundamental though, so once you have the files you want you can rename them without the extra information and put them in any folder you want.

For the old predictors, the RTTOV v13 coefficients for IR sensors have been updated with new spectroscopy: this would account for the larger differences (of order 0.1K). There are also some changes to default option values in RTTOV (new values are recommended), which have a small impact (order 0.01K or less).

There was subsequent discussion related to the technical implementation of the ingest of JMA GRIB data: the users provided two example input files that were used for testing. The updated code was subsequently provided to all beta testers who were asked to apply the updates and recompile to ensure their applications were unaffected by the code updates.

4.2.2 Tester 2

1. Is the software installation straightforward?

Yes, I modified user.cfg file (paths) and ran ./radsim_install and got 'Installation complete' at first attempt. Radsim_check_install works as well.

2. What hardware platform, operating system, and compiler did you use to build the
executable? What versions of external libraries were used? If you ran any of the Python scripts, which version of Python are you using?

Hardware: x86_64 AMD EPYC 7502 32-Core Processor
Operating System: Red Hat 7.7
Compiler: GNU Fortran (GCC) 9.1.0
Versions of external libraries:
eccodes/2.14.1-x86-gnu
aec/1.0.3-x86-gnu
hdf5/1.10.5-x86-gnu
netcdf4/4.7.3-x86-gnu
Jasper: libjasper 1.900.1
Python scripts: I ran radsim_plot_example.py (using Python 3.8.5).

3. Did you encounter any problems building the package on your system?

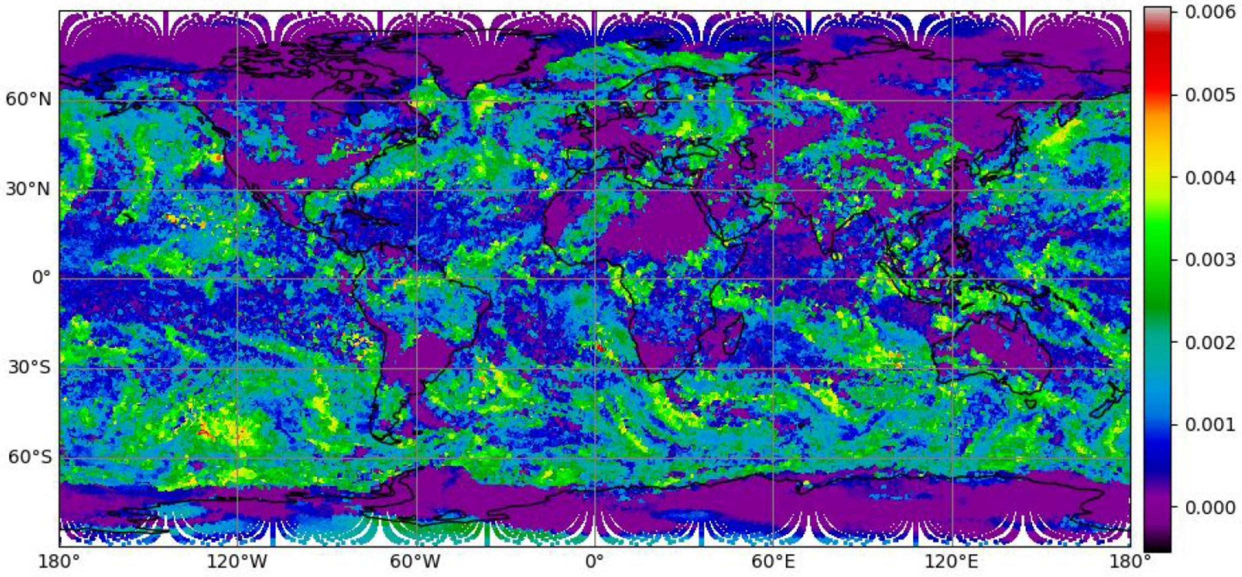
No.

4. How easy was it to set up and run the code? Did you encounter any problems?

Straightforward. No problems.

5. Do you think the results are acceptable? Where relevant, how do they compare to RadSim v2.0? It would be helpful here if you can give a brief overview of how you used RadSim.

I used a global ICON model field as input (but fixed the observation geometry) and ran an MFASIS simulation. I compared the results to DWD's 'datool': The plot shows 'datool minus RadSim'. The errors are comparable (or maybe slightly larger) to a previous RadSim v2-datool comparison, but it is also a different test case. I assume the differences are acceptable?



6. Is the User Guide clear and comprehensive?

Yes, I can confirm that for the part that I read: Sections 1-4 and section 5.8.

7. Assess whether the overall package meets the requirements of the Product Specification.

8. Do you have any suggestions for improvements to the software or documentation?

I used a DWD ICON 3h forecast grib file valid at 2020041512, with analysis time 2020041509. I noticed that the RadSim output filename was radsim-msg_4_seviri-2020041509.nc, i.e., the timestamp corresponds to the analysis time. It should be possible to calculate the valid timestamp of a forecast grib file from 'dateTime' and add the 'forecastTime' if 'significanceOfReferenceTime=1' ('significanceOfReferenceTime=0' is analysis) to make the output file name unique. I have not checked this part of the RadSim code. Note that I have seen 'forecstTime' in seconds and minutes, so units should be checked.

I was interested in running a cloud+aerosol visible simulation using RadSim but realised that aerosol profiles cannot be used in RadSim, but this may be available in a future release. I would be interested in this.

NWP SAF response:

That's great, many thanks for providing feedback!

Regarding whether the differences between RadSim and your datool are acceptable, I think you are probably better able to judge. I assume the differences come from the way the cloud information is provided to RTTOV in the two cases, in particular how the cloud liquid and ice effective diameters are computed. RadSim is limited to the Martin et al parameterisation for liquid cloud: in RadSim v3.0 you can either use the new RTTOV v13.0 internal calculation (if your GRIB file does not contain a density field) or otherwise the same calculation is carried out by RadSim using the model density field. These yield similar but slightly different results since RTTOV's calculation of density is not identical to ICON's. The application of Martin et al has been updated since RadSim v2.2 so will give different results to v2.2. For ice, you have to choose between one of the four Deff parameterisations in RTTOV. I would assume that the datool does something different for ice Deff, and probably also for liquid cloud (as the v13.0 implementation of Martin et al is likely to be unique to RTTOV), so the differences are probably explained mostly by this. In short, if my assumptions above are correct about the datool, then I think the differences are OK.

I will have a look at updating the code to specify the filename according to the data validity time. RadSim already calculates this so it should be easy. But one could theoretically imagine someone running RadSim on a T+9 forecast from 00UTC analysis and a T+3 forecast from a 06UTC analysis and then the filenames would not be unique as the data times are both 09UTC (I admit this is probably a less common scenario though!) The general recommendation is to use the “output_dir” option to keep output files separate, but now I think about it, I will add an “output_filename” option so that users can (optionally) specify a filename explicitly.

I can certainly look at adding aerosols in a future version. It requires aerosol fields to be present in the input model data, and to define a suitable mapping from those fields onto aerosol types in RTTOV. There is now a tool for generating custom aerosol optical property files for use with RTTOV (with the current RTTOV limitations - spherical particles, no explicit Deff dependence). One would mandate the use of specific aerosol property files with aerosol fields from a particular model. If you would like to pursue this we can certainly look at it: I will add a note to the RadSim plans.

4.2.3 Summary

The following is a summary of the outcomes of the RadSim v3.0 beta testing:

- No major issues found, differences to RadSim v2.2 noted by users are within expected tolerances.
- Added *rttov_coeffs_options* namelist variable to make it easier to use RadSim with the new RTTOV v13 optical depth coefficient files that have extra text in the filename.
- Added capability to ingest JMA GRIB data (clear-sky simulations only).
- Added *output_file* namelist variable to enable output filename to be explicitly specified.
- Changed default output filename to use the data validity time (e.g. forecast time) rather than the nominal data time (e.g. corresponding analysis time).
- Noted interest in support for aerosol simulations.