RTTOV v13 Quick Start Guide

James Hocking

Met Office

Change record			
Version	Date	Author / changed by	Remarks
0.1	30/03/20	JH	First draft for v13 beta.
1.0	18/09/20	JH	Updates for release.
1.1	19/10/20	JH	Updates after DRR.
1.2	07/10/21	JH	Updates for v13.1

Table of Contents

1. *Introduction*

This guide provides a summary of how to get started with RTTOV for new users. It assumes a basic knowledge of Fortran 90. Sections 2 and 3 cover compilation of the package and testing your installation. Section 4 gives an overview of basic RTTOV concepts, and these are then demonstrated in section 5 which guides you through a simple example of a clear-sky simulation. You should refer to the user guide for full details of all RTTOV capabilities. If you have problems or questions when compiling or running RTTOV please contact the helpdesk:

https://nwp-saf.eumetsat.int/site/help-desk/

2. *Compilation*

You can compile RTTOV by running the interactive script *build/rttov_compile.sh* from within the *src/* directory.

The script asks which compiler flags you wish to use: it prints out a list of the available options. The default is to use gfortran. If there is an “-openmp” option for your compiler (e.g. gfortran-openmp) this is recommended to enable multi-threaded simulations. The “-debug” options are not recommended.

The script then asks for an installation directory: leave this blank to use the default. (Note: if you do specify a directory, it is a path relative to your top-level RTTOV directory rather than an absolute path).

The script will test whether you have f2py installed and if so will ask whether you want to compile the Python wrapper and the GUI: choose either “y” or “n” (it does not matter which for this guide).

The script asks if you want to specify additional flags: leave this blank.

Finally the script prints a summary of your selected options: enter “y” and return to compile RTTOV.

After compilation the RTTOV executables and libraries can be found in the *bin/* and *lib/* directories in the top-level RTTOV directory respectively (or, if you specified one, within your chosen installation directory).

Several features require RTTOV to be compiled against the HDF5 library: this includes reading of HDF5 coefficient files (the hyperspectral IR sounder coefficients are distributed as HDF5 files for efficiency), the land surface emissivity/BRDF atlases, and the RTTOV GUI (graphical user interface). It is recommended to compile against the HDF5 library, but it is not mandatory and it is not required in order to follow this quick start guide. To compile with HDF5 you must first edit the file *build/Makefile.local* **before** running the *rttov_compile.sh* script: specify the path of your HDF5 installation in the *HDF5_PREFIX* variable, and then uncomment one *FFLAGS_HDF5* line and one *LDFLAGS_HDF5* line.

3. Verifying your installation

Change to the *rttov_test/* directory. You can then verify your installation by running the following script:

```
$ ./test_rttov13.sh ARCH=myarch
```

You should replace *myarch* with whatever compilation flags you chose in answer to the first question from the compilation script (e.g. *gfortran* or *gfortran-openmp*).

This script runs RTTOV simulations for various sensors and compares the output against supplied reference data. It prints out any differences: you may see a few very small differences in Jacobians (the output will mention "*profiles_k*"), but these are due to small compiler-related numerical differences and are normal.

4. RTTOV basics

Coefficient files

The RTTOV optical depth calculation is a parameterisation which requires pre-computed regression coefficients. These coefficients are specific to each instrument and are stored in coefficient files beginning with "*rtcoef_*". The coefficient files can be found in the sub-directories within the *rtcoef_rttov13/* directory.

The RTTOV package contains optical depth coefficient files for a selection of common sensors. RTTOV supports a large number of sensors and all RTTOV coefficients including the large hyperspectral sounder coefficient files are available from the web site:

https://nwp-saf.eumetsat.int/site/software/rttov/download/coefficients/coefficient-download/

If coefficients are not available for the sensor you wish to simulate then you can request a new coefficient file via the helpdesk (see section 1).

There are various types of optical depth coefficient file: the user guide gives details (section 3). In this guide we will focus on the coefficient files contained in the *rtcoef_rttov13/rttov13pred54L/* directory.

The "54L" refers to the fact that RTTOV calculates the optical depths on a fixed set of layers defined by 54 pressure levels (see below). Coefficients are available on other sets of levels (in particular 101 levels) for certain sensors. The "13pred" refers the optical depth parameterisation: the files in this directory are based on the new parameterisation implemented in RTTOV v13.

Options

All aspects of an RTTOV simulation can be configured at run-time by setting the various options defined in the *rttov_options* derived type. As with all RTTOV derived types, this is defined in the *rttov_types* module. The options are grouped in sub-types according to function. The full list of options is given in Annex O of the user guide. Some examples which we will encounter in this guide are:

opts%interpolation%addinterp	activate the RTTOV internal interpolation
opts%rt_ir%addsolar	activate solar radiation in the simulations (UV/visible/IR sensors only)
opts%rt_all%ozone_data	activate ozone as an input trace gas

Input atmospheric/surface variables

RTTOV is a one-dimensional radiative transfer model: it simulates satellite-seen radiances given a vertical profile of atmospheric temperature and water vapour specified on an arbitrary set of pressure levels. The input atmospheric and surface variables are provided to RTTOV in the *rttov_profile* derived type. The pressure, temperature and water vapour on each level are mandatory inputs. Other mandatory inputs include the surface pressure and skin temperature. The full *rttov_profile* structure is given in Annex O of the user guide.

You must declare an array of type *rttov_profile*, say *profiles(:)*. It is possible to pass one or more profiles into RTTOV: RTTOV will simulate radiances for the sensor channels you choose for each profile in *profiles(:)*.

You can specify the input profiles on any pressure levels: the only limitation is that the *number* of pressure levels must be the same for all profiles passed into RTTOV in a single call. In general it is best to use profile data on its native vertical resolution. In this case the RTTOV interpolator must be used by setting the *opts%interpolation%addinterp* option to true (see section 5) so that RTTOV interpolates the profile internally in order to carry out the optical depth regression. The optical depth profiles are then interpolated back onto the input pressure levels on which the radiative transfer equation is solved. For accurate simulations it is essential that the pressure levels encompass the weighting functions of the channels being simulated (i.e. that they cover the full range of atmosphere to which the channels are sensitive).

For UV/visible/IR and some MW simulations RTTOV allows a number of other optional trace gas profiles to be specified such as ozone. The variable gases available depend on the coefficient file (see section 3 of the user guide). If you wish to supply ozone profiles to RTTOV, for example, you must set the *opts%rt_all%ozone_data* option to true. If you leave the option as false RTTOV will use a fixed climatological ozone profile.

You must specify the surface pressure. This is independent of the pressure levels. It is possible to specify a surface pressure below the bottom level of the input pressure profile, but this is not recommended.

Other variables include satellite and solar zenith and azimuth angles, the latitude of the profile (this is used when calculating the effects of Earth's curvature on the atmospheric path) and the surface elevation.

For clear-sky simulations the following variables must always be specified for each profile:

profiles(:)%p(1:nlevels)	pressure levels (hPa)
profiles(:)%t(1:nlevels)	temperature on each level (K)
profiles(:)%q(1:nlevels)	water vapour concentration on each level (units: see below)
profiles(:)%skin%t	surface skin temperature (K)
profiles(:)%skin%surftype	surface type: 0=land, 1=sea, 2=sea-ice
profiles(:)%s2m%p	surface (2m) pressure (hPa)
profiles(:)%s2m%t	surface (2m) temperature (K)
profiles(:)%zenangle	local zenith angle at surface (degrees)
profiles(:)%elevation	surface elevation (km)
profiles(:)%latitude	latitude (degrees)

You can choose the units for all input gas profiles (water vapour, ozone, etc) by setting the *profiles(:)%gas_units* variable. There are three options:

2 => ppmv over moist air

1 => kg/kg over moist air (the default)

0 => ppmv over dry air

Other important profile variables include:

profiles(:)%o3(1:nlevels)	ozone profile (same units as *q*)
profiles(:)%skin%watertype	for water surfaces: 0=fresh water, 1=ocean
profiles(:)%skin%salinity	for water surfaces (PSU), *typical value for sea water: 35*
profiles(:)%s2m%q	surface (2m) water vapour (same units as *q*)
profiles(:)%s2m%u and %v	10m u and v wind components (m/s)
profiles(:)%s2m%wfetc	wind fetch (m), *typical value: 100000m*
profiles(:)%azangle	local azimuth angle at surface (degrees) (see user guide section 8.2)
profiles(:)%sunzenangle	local solar zenith angle at surface (degrees)
profiles(:)%sunazangle	local solar azimuth angle at surface (degrees)

The user guide details the types of simulations in which each profile variable is used. When carrying out your own simulations you should check the description of the *rttov_profile* structure in Annex O carefully and specify values for all relevant profile variables.

Channel selection

RTTOV allows you to simulate an arbitrary subset of sensor channels for each profile you pass. Usually you will want to simulate the same set of the channels for all profiles. The profiles and channels to simulate are specified in the *rttov_chanprof* structure. You must declare an array of this type with size equal to the total number of channels you are simulating over all profiles. If you are simulating *nchannels* channels for each of *nprofiles* profiles, then the size of the *chanprof(:)* array is *nchanprof = nchannels * nprofiles*.

The members of this structure are *chanprof(:)%prof* and *chanprof(:)%chan* which specify each profile/channel combination to be simulated. Suppose you are simulating channels 1-3 of a sensor for 2 profiles. Then you would populate *chanprof(:)* as follows:

chanprof(1:6)%prof = (/ 1, 1, 1, 2, 2, 2 /)

chanprof(1:6)%chan = (/ 1, 2, 3, 1, 2, 3 /)

The profiles must be specified in ascending order with all channels to simulate for profile 1 specified first, followed by all channels for profile 2, and so on. The profile numbers are simply the indices into the *profiles(:)* array that you declare to input your profile data to RTTOV.

The channel numbers are defined by the optical depth coefficient file. Channel numbering in RTTOV *always* starts at 1. You should check the headers of the coefficient file to determine the ordering of the channels. RTTOV generally adheres to each instrument's convention. However note that for an instrument like SEVIRI on the MSG platform for which channels 1, 2, 3 and 12 are in the visible/near-IR, the IR-only coefficient files (for example, with "*_ironly*" in the filename) only contain coefficients for channels 4-11 which are numbered 1-8 in RTTOV. For the new v13 predictor coefficients, the default is for coefficient files to include all channels unless otherwise indicated so that, for example, SEVIRI coefficient files contain coefficients for all 12 channels and in this case they have their standard numbering.

Surface emissivity/reflectance

RTTOV can provide surface emissivities (for IR and MW channels) and BRDFs (for solar-affected channels). The user guide provides details of the models and assumptions RTTOV makes in this case. Alternatively you can input surface emissivity and/or BRDF values for use in the simulation.

You specify whether RTTOV should supply the emissivity or BRDF for each channel individually in the logical *calcemis(:)* and *calcrefl(:)* arrays which are of size *nchanprof*. These correspond to the channels/profiles specified in the *chanprof(:)* structure. For those channels where *calcemis(:)* is true RTTOV will provide an emissivity value, and where *calcemis(:)* is false you should supply an emissivity value. Likewise for *calcrefl(:)* and BRDFs for simulations with solar radiation.

You must also declare arrays *emissivity(:)* of type *rttov_emissivity* and *reflectance(:)* of type *rttov_reflectance*, again both of length *nchanprof*. For those elements of *calcemis(:)* which are false you must specify the surface emissivity in *emissivity(:)%emis_in*, and likewise for *calcrefl(:)* and *reflectance(:)%refl_in*. RTTOV provides a number of emissivity and BRDF atlases which can be used to provide land surface emissivity and BRDF values. These are not described here: see the user guide.

Outputs

The TOA (top-of-atmosphere) radiance outputs are stored in the *rttov_radiance* structure. RTTOV computes radiances for all channels, brightness temperatures (BTs) for channels with significant thermal component, (wavelengths above 3µm) and reflectances for solar-affected channels (wavelengths below 5µm). These can be found in the following members of the *radiance* structure:

radiance%clear(:)	TOA clear-sky radiances (mW/m^2/sr/cm^{-1})
radiance%total(:)	TOA radiances including clouds (mW/m^2/sr/cm^{-1})
radiance%bt_clear(:)	TOA clear-sky BTs (K)
radiance%bt(:)	TOA BTs including clouds (K)
radiance%refl_clear(:)	TOA clear-sky reflectances (no unit)
radiance%refl(:)	TOA reflectances including clouds (no unit)

Each array has size *nchanprof*: there is one element per simulated radiance corresponding to the channels/profiles specified in the *chanprof(:)* structure.

This guide is focusing on clear-sky simulations, so the cloudy outputs will not be mentioned further. If clouds are not activated in the simulations then the clear-sky and “cloudy” outputs are identical.

RTTOV outputs additional radiance and transmittance quantities: these are not described further here, see the user guide Annex O for more information.

RTTOV outputs the surface emissivities used in the simulation in *emissivity(:)%emis_out*. Similarly, for solar simulations the BRDFs used are output in *reflectance(:)%refl_out*.

5. *The example code*

Example Fortran code for running the RTTOV forward model is provided in *src/test/example_fwd.F90*. This is a basic program intended to demonstrate how RTTOV is run. It reads data for one or more profiles from an ASCII file, calls RTTOV once, and then writes various outputs to an ASCII file. The intention is that this code can form the basis of your own application. The code contains many comments to help you understand what it does. The program is compiled to an executable in the *bin/* directory when you build RTTOV.

All file paths in this section are relative to the *rttov_test/* directory. You can run the example from within the *rttov_test/* directory using the *run_example_fwd.sh* shell script:

```
$ ./run_example_fwd.sh ARCH=myarch
```

As before you should set “myarch” to the compiler flags you specified when building RTTOV. The input profile data are specified in the ASCII file *test_example.1/prof.dat*: this file also contains comments to describe the contents. There is a section at the top of the *run_example_fwd.sh* shell script which allows you to configure aspects of the simulation, in particular: the coefficient file of the sensor to simulate, the channel list to simulate, and whether to include solar radiation. You must also specify the number of profiles defined in *prof.dat* and the number of pressure levels. By default the script compares the output to some reference data: if you modify the example code or profile data you can turn off this checking by setting *CHECK_REF=0* in the shell script.

The following paragraphs highlight some important features of the *example_fwd.F90* program:

Variable declarations

RTTOV defines Fortran kinds for real, integer and logical variables in the module *parkind1:* you should import the variables *jprb* (real kind), *jpim* (integer kind) and *jplm* (logical kind) from this module. All variables input to RTTOV should use these kinds where relevant.

The program imports the various derived types mentioned above (plus a few others not mentioned) from the *rttov_types* module and it declares variables of each type. In particular note the following:

Type	***Variable***	***Description***
rttov_options	*opts*	options to configure the simulations
rttov_coefs	*coefs*	structure to hold data from coefficient file
rttov_chanprof	*chanprof(:)*	specify channels/profiles to simulate
logical(kind=jplm)	*calcemis(:)*	specifies if RTTOV should provide emissivities
rttov_emissivity	*emissivity(:)*	input/output surface emissivities
logical(kind=jplm)	*calcrefl(:)*	specifies if RTTOV should provide BRDFs
rttov_reflectance	*reflectance(:)*	input/output surface BRDFs
rttov_profile	*profiles(:)*	input atmospheric and surface data
rttov_radiance	*radiance*	output radiances
rttov_transmission	*transmission*	output transmittances

Specify options

It is important to specify the RTTOV options first as the values of some options will define what members are available in RTTOV structures which are subsequently allocated.

RTTOV will fail with a fatal error if you try to specify mutually incompatible options or if the options are not compatible with the loaded coefficient file (see below). This would happen, for example, if you set the *addsolar* option to true with a coefficient file that does not support solar simulations.

In general you will want to enable the RTTOV interpolator (*opts%interpolation%addinterp*): you can then input profiles on arbitrary pressure levels.

In this example the *ozone_data* option is set to false. If you wish to supply ozone profiles you would need to set this to true, modify the section of code which reads profile data to ingest ozone data (see below) and finally provide ozone data in the input data file.

Read the coefficient file

The coefficients are read by calling the *rttov_read_coefs* subroutine. The example code demonstrates a simple call to this subroutine which reads all channels from the file specified by it's full path. Annex C of the user guide gives the full interface to this subroutine. In particular it is possible read only a subset of channels from the file, but this will not be described here.

Allocate input/output arrays and structures

RTTOV provides the *rttov_alloc_direct* subroutine which can be used to allocate (and later deallocate) any/all input/output arrays and structures for the RTTOV direct model.

Populate *chanprof(:)*

The code then populates the *chanprof(:)* as described in section 4 above. In this case the sensor channels specified in the *run_example_fwd.sh* shell script are simulated for each profile passed in.

Read profile data

The next section reads the profile data from *prof.dat* into the *profiles(:)* structure. This is intended for demonstration purposes: this ASCII format is not recommended for a real-world application. It is recommended to replace this section with code to read the profile data relevant to your simulations directly from whatever file format they are provided in such as NetCDF, for example.

Specify surface emissivity/BRDF

In this example we get RTTOV to provide values for the surface emissivity (and BRDF if solar simulations have been activated) by setting *calcemis(:)* and *calcrefl(:)* to true for all channels. In this case it does not matter what we specify in the *emissivity(:)%emis_in* and *reflectance(:)%refl_in* arrays.

Call the RTTOV direct model

This is achieved with a single call to *rttov_direct*. If you compiled RTTOV with “-openmp” compiler flags and you have a CPU with multiple cores you can take advantage of multi-threaded simulations simply by calling the *rttov_parallel_direct* subroutine instead. This has exactly the same interface as *rttov_direct*, but with a final optional argument, *nthreads*, which allows you to specify the number of threads to use. It is recommended to make use of this capability if you are running many simulations.

Once *rttov_direct* returns you should check the *errorstatus*: if it is zero the simulation was successful, but any non-zero value indicates there was an error. This applies to all RTTOV subroutines which return an error code.

Store the results

In this example the output is quite verbose: it prints out the options structure and, for each profile, it prints out the contents of the profile structure, and a selection of radiance, emissivity, BRDF and transmittance outputs. In a real-world application you would want to replace this section with code to store the outputs you need to some suitable file.

Deallocate data

In order to prevent memory leaks you should ensure all allocated data is deallocated. This is easily achieved for the RTTOV arrays and structures using the *rttov_alloc_direct* subroutine again.

Running RTTOV for your application

You have seen how to run the RTTOV direct model for clear-sky simulations. The *src/test/* directory contains similar *example_*.F90* code for various simulation types: these are all structured in a similar way and hopefully you can now understand these with reference to the user guide. You can base your own application on the relevant *example_*.F90*. However you would almost certainly want to make some modifications:

- Profile ingest: the ASCII format used here is used to make the example clear. In practice you would want to read the profile data directly from whatever file format they are provided in.
- Output: the output from the example code is very verbose and is in ASCII format. In practice you would want to remove much of the output code and write the simulated data to some suitable binary format, especially if simulating many profiles.
- Simulating many profiles: this example is set up to call RTTOV once with multiple profiles because this is the simplest way to demonstrate running RTTOV. In practice if you have many thousands of profiles to simulate or if your simulations require a lot of memory (e.g. many channels of a hyperspectral sounder, scattering simulations) you may prefer to modify the code so that the profiles are simulated in smaller batches. In this case you would allocate your structures with some small value of *nprofiles* (say 10 or 50) and then loop over the input data, reading in *nprofiles*-worth of data and calling RTTOV on them in each loop. You might want to set up large arrays into which you store the outputs you require as they are calculated so that you can write these outputs to a file in one go once all profiles have been simulated.

6. *Other capabilities*

Having run a clear-sky simulation, you may be interested in exploring other capabilities of RTTOV. Some examples are given below (the list is not exhaustive): please consult the user guide for full details of how to run RTTOV (section 7) and for details of specific types of simulations (section 8).

Computing Jacobians

RTTOV comprises tangent linear (TL), adjoint (AD) and full Jacobian (K) models. The K model can be used to compute the sensitivity of the top-of-atmosphere radiances to each input variable. This is used, for example, in 1DVar retrievals of atmospheric profiles.

Scattering simulations with aerosols, clouds and/or hydrometeors:

RTTOV can simulate aerosol- and/or cloud-affected UV/visible/IR radiances. RTTOV provides various options for aerosol, liquid cloud and ice cloud optical properties. Alternatively it is possible to pass profiles of optical properties per channel explicitly. The RTTOV-SCATT model can simulate cloud- and hydrometeor-affected MW radiances: this is a separate interface to RTTOV, but it uses RTTOV for the gas optical depth calculations.

Principal Components simulations for hyperspectral IR sounders:

PC-RTTOV is a Principal Components (PC) based model for hyperspectral sensors: by carrying out normal RTTOV simulations for a few hundred channels PC-RTTOV can compute PC scores for the whole spectrum and then, optionally, reconstruct radiances for any set of channels of the sensor. This is an efficient way of simulating full spectra of hyperspectral sounders.

RTTOV also provides an interface to the HTFRTC PC-based radiative transfer model.

Calling RTTOV from Python or C++:

A wrapper has been created for RTTOV which allows you to call much RTTOV capability directly from Python or C++ (or C). Note that RTTOV only supports Python3; Python2 is not supported. This could be useful if you are more familiar with one of these languages than Fortran: the wrapper is fully documented in *docs/rttov-wrapper.pdf*, but you must also refer to the user guide to understand how to run RTTOV correctly.

RTTOV GUI:

A graphical user interface (GUI) has been created for RTTOV which allows most types of simulation to be run. Input profiles and output radiances and Jacobians are visualised and it provides a clear way to understand the impact of changing RTTOV options or profile variables. The GUI is described in *docs/rttov_gui_v13.pdf*.