NWP SAF

RTTOV Version 14 Top Level Design

Version 1.1

2nd July 2024

RTTOV Version 14 Top Level Design

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 September 2021, 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**	**Approved**	**Remarks**
1.0	19/07/2023	JH		Initial version
1.1	02/07/2024	JH		Updated to reflect changes during development

Table of Contents

1. INTRODUCTION

This document defines the top-level design for Version 14 of RTTOV, in accordance with the requirements of the NWP SAF.

1.1 Reference documents

[RD-1] NWPSAF-MO-DS-048 RTTOV v14 Product Specification

1.2 Design overview

Managing computer resources

The major requirement for RTTOV v14 is to allow the user to be able to run the code without limits on the number of channels, profiles, and instruments in the same run. This is achieved by using arrays of derived types and dynamic memory allocation. Derived types reduce the number of arguments to be passed to the subroutines and facilitates the memory allocation for the arrays. The remaining limits are given by:

- The external RTTOV coefficient file which contains the number of fixed pressure levels on which the optical depth computation is performed.
- The size of the computer physical memory and sufficient disk space to store the files.

The user can allocate all the derived types which are arguments of the subroutines (coefficients, profiles, radiances, channels, PCscores, etc). In this way the user can run inside the same main program calls for different instruments for which the user input profiles can be on different pressure levels to the coefficients.

Configuring simulations

All aspects of RTTOV simulations are configured at run-time by setting variables in the *rttov_options* derived type. Many options are available, for example, whether solar radiation is enabled, and whether hydrometeors and/or aerosols are included. Users should set the broadest range of options required for their simulations before reading coefficients or allocating data structures, to ensure that all relevant data are read in, and all necessary arrays are allocated. This is because only the necessary data are read in and allocated to minimise memory usage for each type of simulation. Options can subsequently be turned off when running RTTOV if desired.

Managing coefficient files

The user manages the RTTOV coefficients within their program and is responsible for allocating the maximum number of instruments for the run. The user calls the RTTOV coefficient file ingest subroutine *rttov_read_coefs* with the IDs of the satellite and instrument, or with a filename, and optionally with a list of selected channels to read (otherwise all channels are read). *rttov_read_coefs* reads the data and allocates memory for the storage. The *rttov_dealloc_coefs* subroutine deallocates memory for the coefficients. For hydrometeor/aerosol simulations (including the MFASIS-NN fast visible cloud parameterisation) and/or PC-RTTOV simulations, additional optical property or coefficient files are required, and these are ingested in the same call to *rttov_read_coefs* as the RTTOV gas optical depth coefficient file. The *rttov_conv_coef* executable enables the user to convert all these file types between ASCII, Fortran unformatted ("binary"), and netCDF formats, and allows the extraction of a subset of channels. This enables greater efficiency when using the large coefficient files required by hyperspectral sounders.

RTTOV v14 remains compatible with all existing RTTOV v13 optical depth (prefix *rtcoef*) coefficient files. However, the merging of the old RTTOV-SCATT model capabilities within RTTOV has resulted in updates to the hydrometeor and aerosol optical property files, and the changes in simulated radiances mean that new MFASIS-NN coefficients and PC-RTTOV coefficients must be generated for RTTOV v14. As a result, the hydrometeor and aerosol optical property files, and the MFASIS-NN and PC-RTTOV coefficient files are all new for v14 and there is no inter-compatibility with RTTOV v13 for these.

The *rttov_user_check_options* subroutine is provided to check compatibility between the selected options and coefficients that have been read in. This can be useful for debugging simulations as it warns not only of strictly inconsistent option settings, but optionally also questionable settings that will not cause errors but might indicate that the user is not simulating what they expect.

Management of atmospheric profiles

The user provides RTTOV with atmospheric profiles and surface variables via the *rttov_profile* derived type. Pressure levels are not fixed and need to be provided by the user along with the profile: the only limitation is that all profiles passed to RTTOV in a single call have the same number of pressure levels, though the pressures themselves may vary between profiles. The *rttov_user_check_profile* subroutine allows users to check the validity of an input profile before calling RTTOV. Alternatively, RTTOV can check the profiles internally. This checking includes comparisons of the profile to the limits derived from the minimum/maximum value envelopes of the profile set with which the RTTOV coefficients are trained. The min/max profile envelopes are contained in the coefficient files.

The *rttov_opt_param* derived type is used to provide explicit profiles of optical properties for scattering simulations. The *rttov_user_check_opt_param* subroutine allows users to validate these optical property inputs before calling RTTOV.

Management of instrument channels

The user provides the model with a list of channel numbers for each profile for which radiances are to be simulated. The definition of the channel numbers is provided in the coefficient file for each instrument. An error is thrown if an invalid channel number is provided.

Surface emissivities and reflectances

The *rttov_emis_refl* derived type is used to provide surface emissivity and reflectance information to RTTOV and to obtain emissivity and reflectance information used in the simulations.

RTTOV can compute IR (over ocean) and MW (over ocean and land/sea-ice) emissivities according to some input surface parameters. The user populates a logical array indicating for which channels and profiles the model should calculate emissivities. The ocean surface emissivity models available in RTTOV v14 are ISEM or IREMIS for infrared radiances, and SURFEM-Ocean, and FASTEM versions 5 and 6 for microwave radiances. For infrared radiances RTTOV provides a fixed value of emissivity over land and sea-ice.

For solar simulations, a BRDF (bi-directional reflectance function) value is required for the direct solar beam that is reflected by the Earth's surface towards the sensor. RTTOV can compute sea surface BRDFs due to sun glint. For land and sea-ice, RTTOV provides crude defaults for channels below 3µm, while for channels above 3µm a Lambertian BRDF is calculated from the surface emissivity. The user populates a logical array indicating for which channels and profiles RTTOV should calculate BRDFs.

For downwelling emitted and scattered radiation, RTTOV requires a "diffuse reflectance". For channels below 3µm, this is interpolated from a fixed water reflectance spectrum for sea surfaces, or otherwise is derived from the BRDF for land and sea-ice surfaces. For channels above 3µm it is calculated from the surface emissivity. As for emissivity and BRDF, the user populates a logical array indicating the channels and profiles for which RTTOV should compute the diffuse reflectance.

The user may optionally provide input emissivities, BRDFs, and diffuse reflectances for any channel. For land surfaces, monthly climatological emissivity and BRDF atlases are available which can provide emissivity and BRDF values to be passed into RTTOV. The available atlases are the University of Wisconsin, CAMEL v2 and v3 (based on a single year of MODIS and ASTER data), and CAMEL climatology v2 and v3 (based on combined multi-year data) emissivity atlases in the infrared, the TELSEM2 and CNRM emissivity atlases in the microwave, and the CMS BRDF atlas for visible and near-IR channels.

In all cases the emissivities and reflectances computed/used by RTTOV are returned in the *rttov_emis_refl* structure. References for the emissivity models and atlases are given in the RTTOV user guide.

As described in [RD-1], RTTOV v14 allows for each profile to be associated with multiple different surfaces, each with different properties, representing an inhomogeneous field of view. The emissivity, BRDF, and diffuse reflectance inputs and outputs described above are specified for each surface independently via an array of objects of type *rttov_emis_refl*.

The *rttov_user_check_emis_refl* subroutine allows users to validate the emissivity and reflectance inputs before calling RTTOV.

Input and output

The table below lists the arguments accepted by the **rttov_direct** subroutine. Here **nchanprof** is the size of the **chanprof(:)** array (i.e. the total number of radiances to compute), **nprofiles** is the size of the **profiles(:)** array (i.e. the number of profiles to process in each call to RTTOV), **nsurfaces** is the number of surfaces in each profile, and **nchannelsrec** is the number of reconstructed radiances to compute (for PC-RTTOV). The **rttov_parallel_direct** subroutine has the same arguments as **rttov_direct** (except for the "trajectory" **traj*** arguments) plus an optional argument **nthreads** to specify the number of threads.

Type	In/Out	Variable	Description
Integer	Intent(out)	errorstatus	Return code.
Type(rttov_options)	Intent(in)	opts	RTTOV options structure
Type(rttov_coefs)	Intent(in)	coefs	RTTOV coefficient structure.
Type(rttov_chanprof)	Intent(in)	chanprof(nchanprof)	Chanprof structure.
Type(rttov_profile)	Intent(in)	profiles(nprofiles)	Profiles structure.
Type(rttov_emis_refl)	Intent(inout)	emis_refl(nsurfaces)	Input/output surface emissivity and reflectance structure.
Type(rttov_transmission)	Intent(inout)	transmission	Output transmittances (0-1).
Type(rttov_radiance)	Intent(inout)	radiance	Output radiances (mW/cm^{-1}/sr/m^2, degK, and BRF/unitless).
Type(rttov_radiance2)	Intent(inout), optional	radiance2	Secondary output radiances (mW/cm^{-1}/sr/m^2).
Real	Intent(in), optional	refl_cloud_top(nchanprof)	Cloud top BRDF for UV/VIS channels for simple cloud scheme (RTTOV provides defaults if omitted).
Type(rttov_opt_param)	Intent(in), optional	aer_opt_param	Aerosol optical parameter input profiles if **opts%scatt%aerosols** and **opts%scatt%user_aer_opt_param** are true.
Type(rttov_opt_param)	Intent(in), optional	hydro_opt_param	Hydrometeor optical parameter input profiles if **opts%scatt%hydrometeors** and **opts%scatt%user_hydro_opt_param** are true.
Type(rttov_reflectivity)	Intent(inout), optional	reflectivity	Output radar reflectivities if **opts%scatt%hydrometeors** and **opts%scatt%radar** are true.
Type(rttov_emis_retrieval_terms)	Intent(inout), optional	emis_retrieval_terms	Output structure for data necessary for dynamic surface emissivity retrievals.
Type(rttov_diagnostic_output)	Intent(inout), optional	diag_output	Output structure for additional diagnostic output from simulations.
Type(rttov_pccomp)	Intent(inout), optional	pccomp	Structure to hold output PC scores and reconstructed radiances.
Integer	Intent(in), optional	channels_rec(nchannelsrec)	Channels for which to compute reconstructed radiances if **opts%pcrttov%rec_rad** is .true.
Type(rttov_traj)	Intent(inout), optional	traj	Trajectory structure to hold temporary data: may improve performance on some architectures for certain types of simulation.
Type(rttov_traj_dyn)	Intent(inout), optional	traj_dyn	As above
Type(rttov_traj_sta)	Intent(inout), optional	traj_sta	As above

The interfaces for the TL, AD and K models are similar to those for the direct model calls shown above. They have some additional arguments for the TL and AD/K variables, but these use the same derived types as the direct model.

All simulations require a gas optical depth coefficient file (prefix *rtcoef*) for the sensor being simulated. Aerosol/hydrometeor simulations additionally require an aerosol and/or hydrometeor optical property file (prefix *rttov_aertable*/*rttov_hydrotable*) for the sensor unless the user chooses to provide profiles of the required optical properties explicitly. The MFASIS-NN fast visible cloud model requires a neural network coefficient file (prefix *rttov_mfasis_nn_hydro*) for the sensor being simulated. PC-RTTOV requires a PC coefficient file (prefix *pccoef*) relevant to the sensor and kind of simulation being performed.

Subroutines are provided for users to (de)allocate and initialise all the derived types listed in the tables above at runtime. Detailed descriptions of the derived types are given in the RTTOV user guide.

2. SOFTWARE FUNCTIONS AND MODULE/SUBROUTINE DESIGN

2.1 User-level functionality

RTTOV v14 provides the following primary user-level functions:

- Subroutine to read RTTOV coefficient files, hydrometeor/aerosol optical property files, MFASIS-NN coefficient files, and PC-RTTOV coefficient files.
- Subroutine to check that user-specified options are consistent with the loaded coefficients.
- Subroutines to check validity of a user profile, user input emissivity/reflectance data, and user input scattering optical property profiles.
- Subroutines to (de)allocate and initialise RTTOV input/output structures.
- Subroutines to call the direct, tangent linear (TL), adjoint (AD) and Jacobian (K) models.
- Subroutines to call the direct, TL, AD and K models using multiple threads (OpenMP).
- Subroutines to initialise and deallocate the emissivity and BRDF atlases.
- Subroutines to access initialised emissivity and BRDF atlases.

In addition, there are a number of ancillary subroutines which may be helpful for users. The user guide provides a full list of the user level subroutines and the interfaces to all user-level subroutines.

2.2 RTTOV direct model design

The *rttov_direct* (direct model) subroutine performs the following tasks:

- Initialises primary outputs.
- Performs checks on inputs to catch inconsistent or invalid options and arguments.
- Allocates and initialises internal data structures.
- Populates the profile data structure used by RTTOV internally. Among other tasks, this converts input profiles to the units used internally by RTTOV.
- If the *check_profiles* Boolean option is true, checks input profiles, emissivities/reflectances, and optical property profiles (if supplied) for unphysical/invalid values.
- Carries out ancillary calculations related to profile geometry.
- If the *gas_opdep_calc* Boolean option is true:
    - Populates the profile data structure on coefficient levels. This involves interpolation of the input profiles onto the coefficient pressure levels.
    - Checks profiles on coefficient levels against the optical depth regression limits, and if the *verbose* Boolean is true and the *apply_reg_limits* Boolean is false, outputs warnings when limits are exceeded.

    - If the *apply_reg_limits* boolean is true, clips the (interpolated) profiles to the regression limits. This clipping is applied to any extrapolated profile data at the top of the profile when the RTTOV interpolator is used.
    - Performs ancillary calculations related to profile geometry on the interpolated profiles.
    - Calculates gas absorption optical depth predictors on coefficient levels.
    - Calculates gas absorption optical depths on coefficient levels.
    - Interpolates gas absorption optical depths onto the input pressure levels.
- Applies gamma correction to gas optical depths.
- If the *clw_data* Boolean option is true, calculates optical depth due to cloud liquid water absorption for microwave sensors.
- If requested, calculates Rayleigh extinction for UV/visible/near-IR channels.
- If hydrometeor simulations requested, calculates cloud columns due to selected cloud overlap assumption.
- Allocates internal data structures which depend on the number of cloud columns.
- If PC-RTTOV aerosol or hydrometeor simulations requested, applies regression limits for input aerosol or hydrometeor profiles.
- If hydrometeor/aerosol simulations requested, compute the optical properties for hydrometeors and/or aerosols required for the chosen solver(s).
- Calculates atmospheric transmittances.
- Carry out surface emissivity and reflectance calculations:
    - Calculates surface emissivities, BRDFs, and diffuse reflectances for all surfaces, where requested by user.
    - Compute surface leaving radiances for all surfaces.
    - Compute surface radiance and reflectance quantities for all surfaces combined.
- Compute atmospheric radiance emission.
- Call the relevant routine(s) that implement the requested solar and/or thermal solvers.
- If requested, populate the *emis_retrieval_terms* output structure.
- If selected, calculate the non-local thermodynamic equilibrium (NLTE) bias correction.
- For PC-RTTOV, calculate PC scores and if requested the reconstructed radiances and brightness temperatures.
- For all other simulation types, calculate the output top of atmosphere cloudy radiances, brightness temperatures, and reflectances for each channel as appropriate, and the radar reflectivities if requested.
- Deallocate all internally allocated memory.

Figure 2 below gives the calling tree for *rttov_direct* which follows the above description.

2.3 Tangent linear (TL), adjoint (AD) and Jacobian (K) models

The TL model is based on the forward model. The functionality of each subroutine is the same as in the direct model, but they calculate the tangent linear as an additional output.

The AD and K first call the direct model to get the forward "trace" and then respectively the AD or K subroutines are called in the reverse sequence order.

The user guide provides details on what the TL, AD and K models compute and how they should be called. The same routine names as for the forward model are used but appended with "*_tl*", "*_ad*", and "*_k*".

2.4 Calling trees

Figures 1 and 2 below show the calling trees for a user program calling the RTTOV v14 direct model, and for the *rttov_direct* subroutine. Only subroutines called from the top-level subroutine are shown.

Calling tree for user program calling RTTOV v14

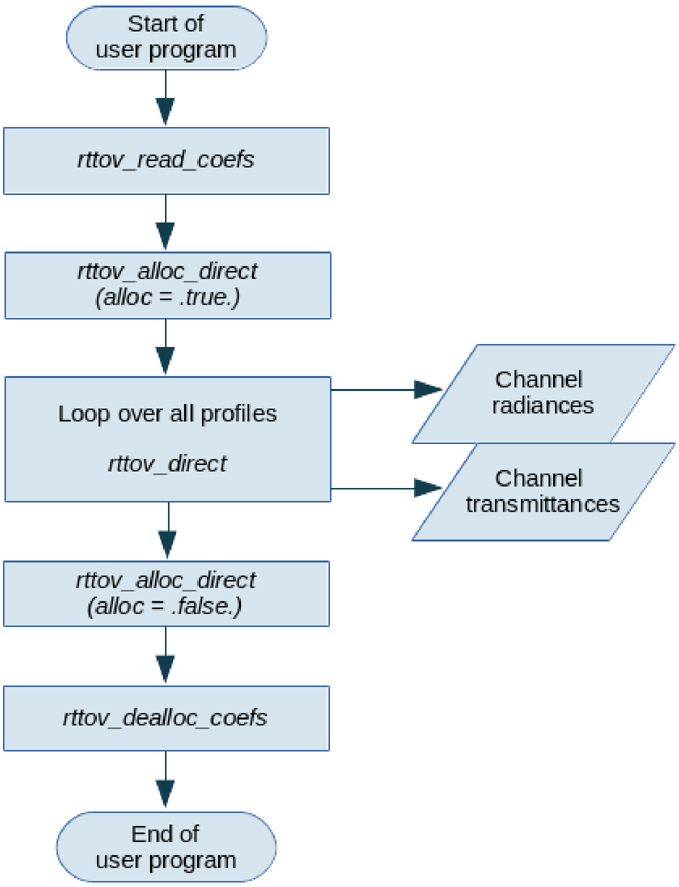
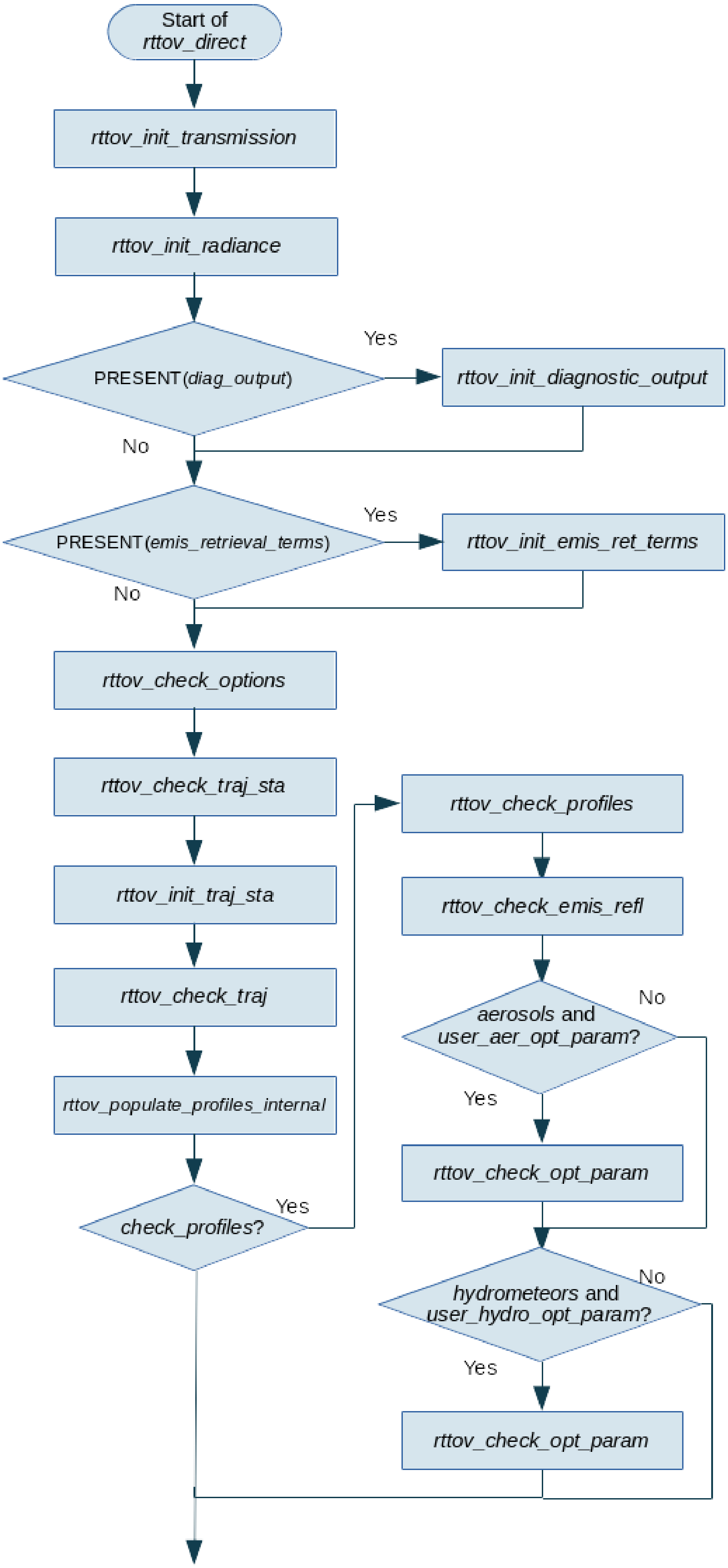
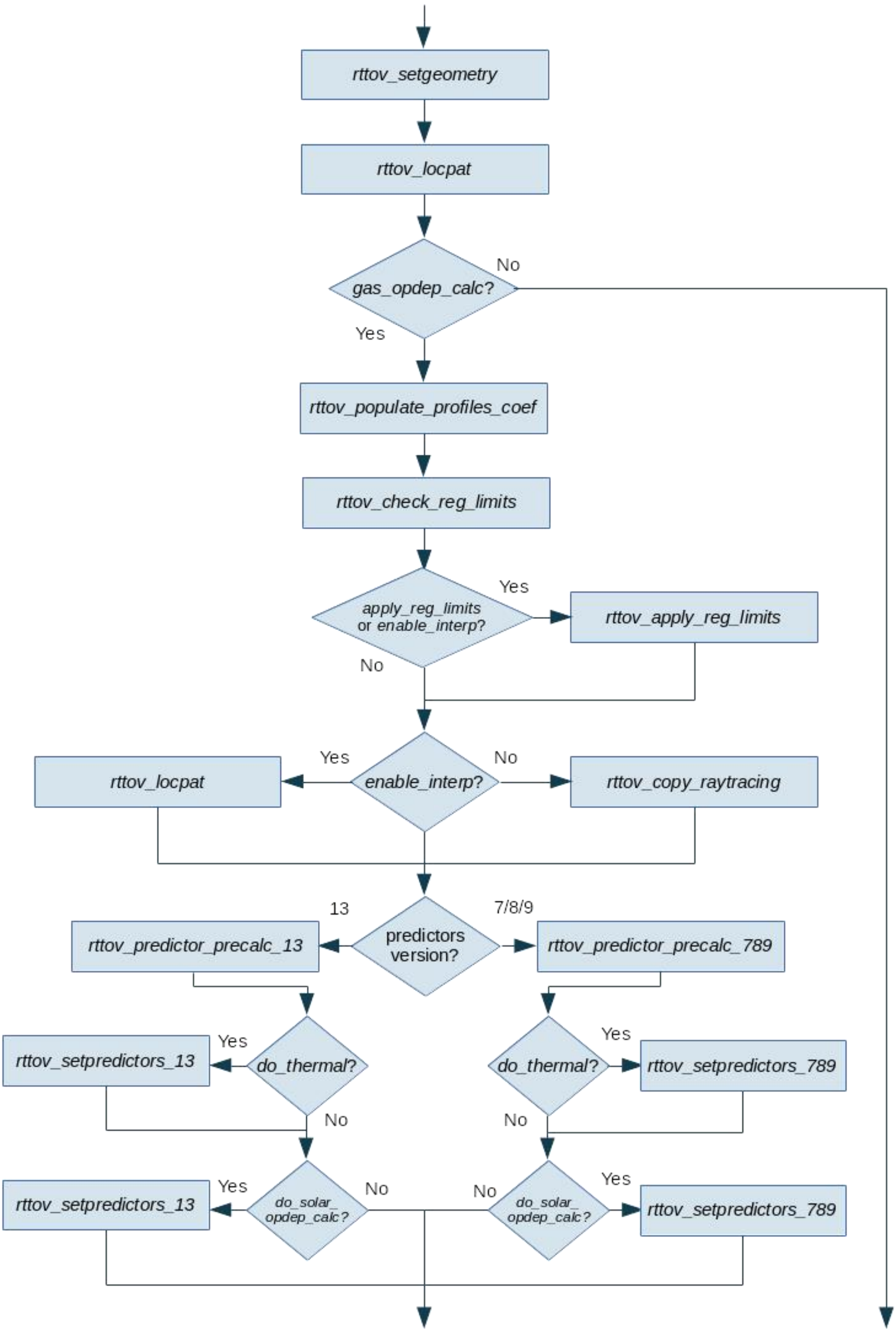


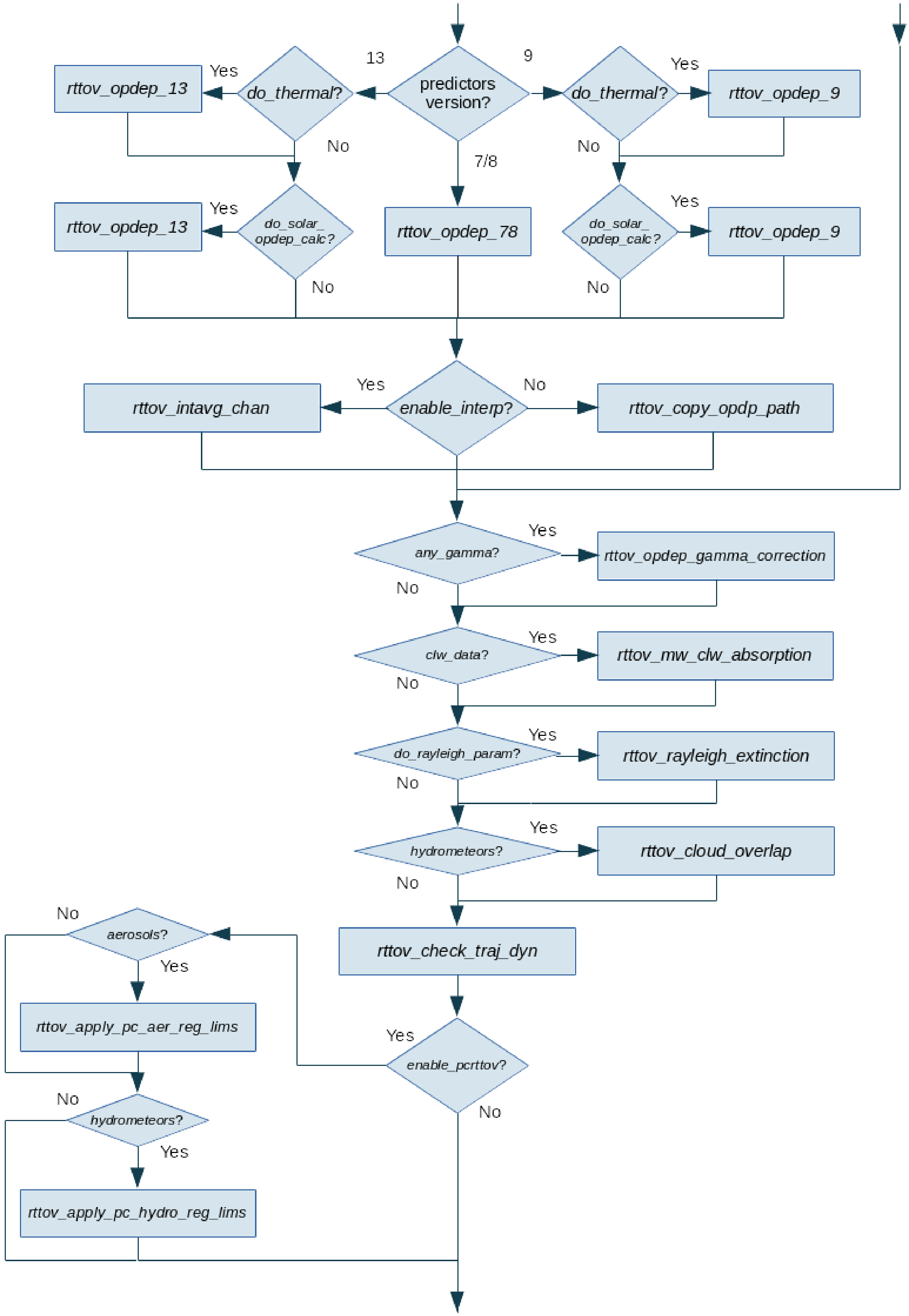
Figure 1. Process diagram of user program calling RTTOV v14 forward model.

Calling tree for *rttov_direct*

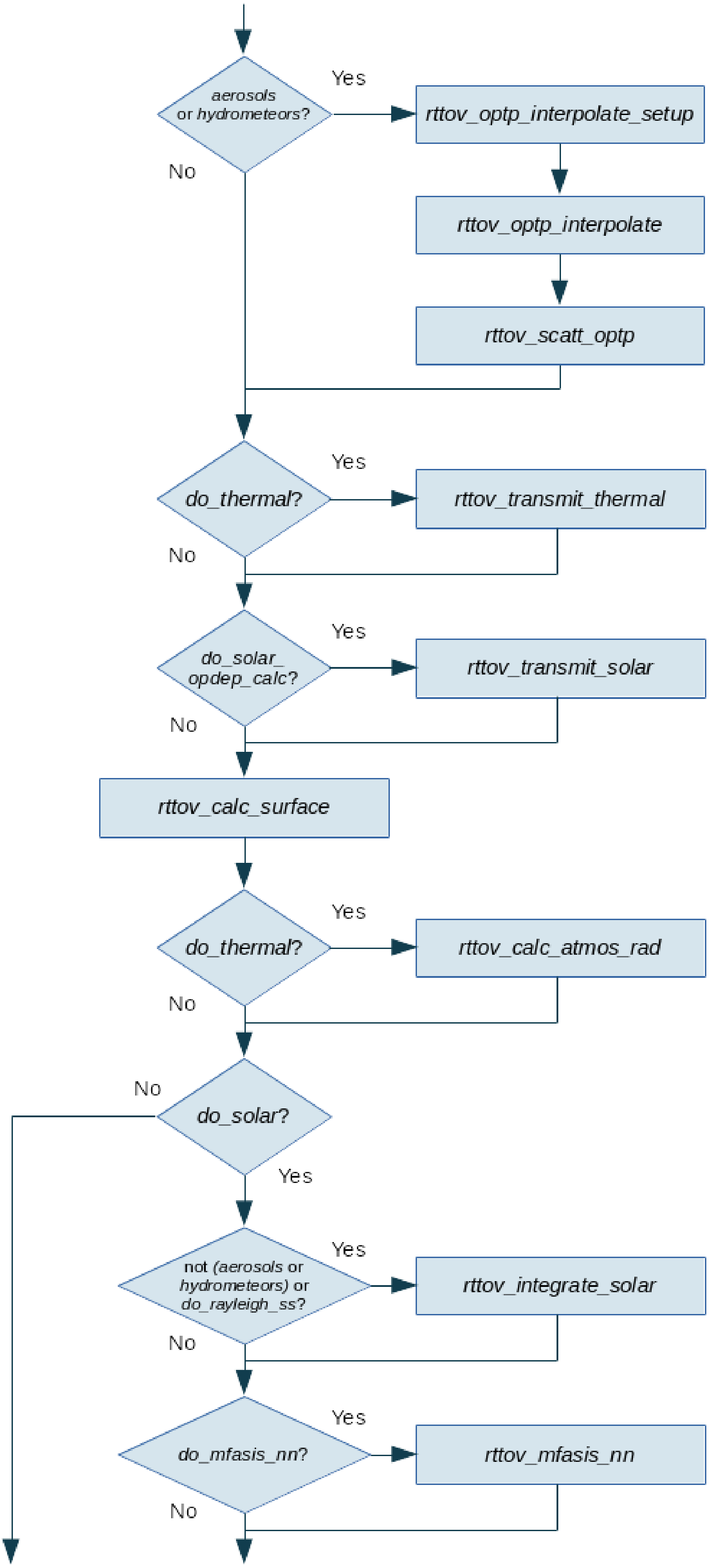




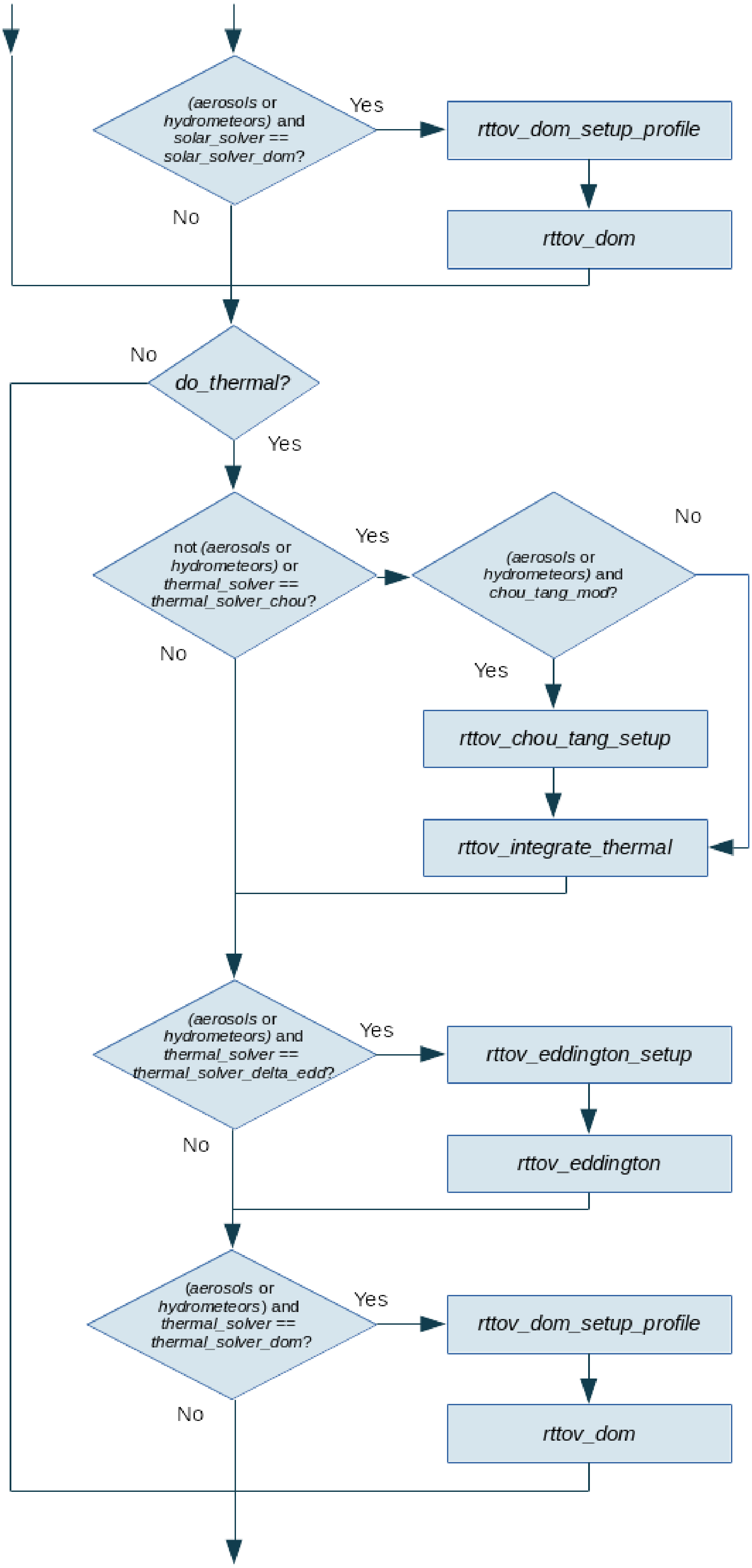
rttov_setgeometry
rttov_locpat
gas_opdep_calc?
No
Yes
rttov_populate_profiles_coef
rttov_check_reg_limits
apply_reg_limits
or enable_interp?
Yes
rttov_apply_reg_limits
No
rttov_locpat
Yes
enable_interp?
No
rttov_copy_raytracing
13
predictors
version?
7/8/9
rttov_predictor_precalc_13
rttov_predictor_precalc_789
rttov_setpredictors_13
Yes
do_thermal?
No
do_thermal?
Yes
rttov_setpredictors_789
No
rttov_setpredictors_13
Yes
do_solar_
opdep_calc?
No
No
do_solar_
opdep_calc?
Yes
rttov_setpredictors_789



predictors version?
13
9
7/8
do_thermal?
Yes
No
rttov_opdep_13
rttov_opdep_9
do_solar_opdep_calc?
rttov_opdep_78
enable_interp?
rttov_intavg_chan
rttov_copy_opdp_path
any_gamma?
rttov_opdep_gamma_correction
clw_data?
rttov_mw_clw_absorption
do_rayleigh_param?
rttov_rayleigh_extinction
hydrometeors?
rttov_cloud_overlap
rttov_check_traj_dyn
enable_pcrttov?
aerosols?
rttov_apply_pc_aer_reg_lims
rttov_apply_pc_hydro_reg_lims



aerosols or hydrometeors?
Yes
No
rttov_optp_interpolate_setup
rttov_optp_interpolate
rttov_scatt_optp
do_thermal?
Yes
No
rttov_transmit_thermal
do_solar_ opdep_calc?
Yes
No
rttov_transmit_solar
rttov_calc_surface
do_thermal?
Yes
No
rttov_calc_atmos_rad
No
do_solar?
Yes
not (aerosols or hydrometeors) or do_rayleigh_ss?
Yes
No
rttov_integrate_solar
do_mfasis_nn?
Yes
No
rttov_mfasis_nn


(aerosols or hydrometeors) and solar_solver == solar_solver_dom?
Yes
No
rttov_dom_setup_profile
rttov_dom
do_thermal?
No
Yes
not (aerosols or hydrometeors) or thermal_solver == thermal_solver_chou?
Yes
No
(aerosols or hydrometeors) and chou_tang_mod?
No
Yes
rttov_chou_tang_setup
rttov_integrate_thermal
(aerosols or hydrometeors) and thermal_solver == thermal_solver_delta_edd?
Yes
No
rttov_eddington_setup
rttov_eddington
(aerosols or hydrometeors) and thermal_solver == thermal_solver_dom?
Yes
No
rttov_dom_setup_profile
rttov_dom

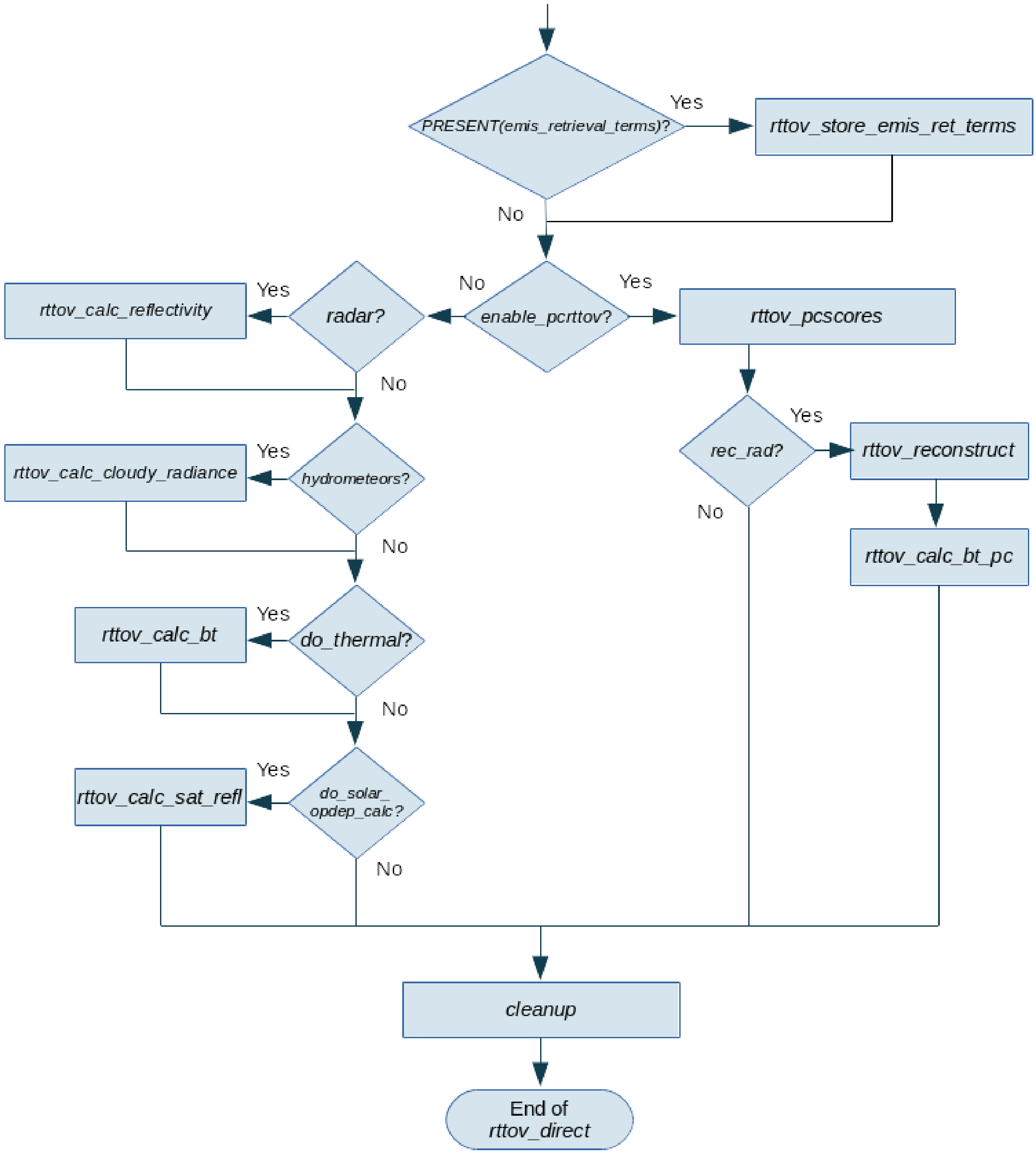


Figure 2. Calling tree for rttov_direct subroutine

3. IMPLEMENTATION ASPECTS

3.1 External libraries

netCDF

RTTOV can be compiled and run without any external dependencies. However, for reasons of efficiency the coefficient files for hyperspectral sounders, and the emissivity and BRDF atlases (excluding TELSEM2) are delivered in netCDF format. To make use of the full capabilities of RTTOV it is necessary to compile against the netCDF library (v4.1 or later). (NB Users can request ASCII versions of the coefficient files via the helpdesk. Atlas data files are only available in netCDF format).

RTTOV includes a Perl script (*build/Makefile.PL*) which automatically generates the Makefiles: to enable netCDF support this script is run with the argument *RTTOV_NETCDF=1*.

To manage compilation with/without netCDF support enabled, the calls to netCDF subroutines are switched in/out at compile-time by the presence/absence of the _RTTOV_NETCDF macro which may be supplied to the compiler. The user must edit the file *build/Makefile.local* to provide the compiler with the location of their netCDF installation. This file already includes the relevant lines which pass this macro to the compiler.

F2PY

The Python interface to RTTOV requires f2py, and by extension, so does the RTTOV GUI as the latter depends upon the former. If the user has f2py installed, they must pass *RTTOV_F2PY=1* to the *build/Makefile.PL* Perl script and then the relevant wrapper code will be compiled.

LAPACK

RTTOV uses several subroutines from the LAPACK library, and the source code for these are included in the package. Some users find this causes problems at compile-time if they are introducing RTTOV into a larger software system in which a version of the LAPACK library is already being linked. Users can choose to exclude the internal RTTOV LAPACK source code, and compile RTTOV against an external LAPACK library. To do this they must pass RTTOV_USER_LAPACK=1 to the *build/Makefile.PL* script (so that the LAPACK source code is excluded from the Makefiles) and edit the file *build/Makefile.local* to provide the compiler with the location of their LAPACK library.

3.2 Compiler requirements

RTTOV requires a Fortran2008 compiler. The code is mainly Fortran90/95 with some commonly supported features from F2003 and F2008. The user guide lists compatible compilers and the full list of platforms/compilers with which RTTOV v14 is successfully tested is given in the Test Log accompanying the Test Plan.

3.3 Required and optional third-party software

RTTOV is self-contained (aside from the external libraries noted above) and so no additional third-party software is required beyond common applications found on typical Unix/Linux systems.

3.4 Build method

Compilation of RTTOV involves the following steps:

- Edit *build/Makefile.local* to point to the user's netCDF installation (optional, required to use netCDF format coefficient files and atlases)
- Edit *build/Makefile.local* to point to the user's LAPACK library (optional)
- Call *build/Makefile.PL* from within the *src/* directory (optional, required if netCDF support and/or Python interface are required and/or a user LAPACK library is being used)
- Choose (and if necessary, edit) the appropriate compiler flag file in *build/arch/* (this contains several sets of flags for compilers with which RTTOV has been tested)
- Run *make* to compile RTTOV.

An interactive shell script *build/rttov_compile.sh* is provided to simplify this procedure for users. In this case the user must edit *build/Makefile.local* (if necessary) and then run *rttov_compile.sh*: the script prompts the user for several inputs and then executes the necessary commands to compile RTTOV.

The above build system is a very flexible custom method developed many years ago for RTTOV. The introduction of the *rttov_compile.sh* script significantly reduced the number of NWP SAF Helpdesk queries related to compiling RTTOV. However, based on user feedback, it is planned to investigate CMake as an alternative build system to the above approach for v14.1 or v14.2. If implemented, it will exist alongside the above system.

3.5 Run-time considerations

The memory requirements of RTTOV vary a great deal with the kind of simulation being performed. For the more memory-intensive types of simulation (for example scattering or PC-RTTOV) the user may find they have to pass fewer profiles to RTTOV in each call than they would for clear-sky simulations, for example.