RTTOV v12 Users Guide

James Hocking, Peter Rayer, David Rundle and Roger Saunders
Met Office, Exeter, UK
&
Marco Matricardi and Alan Geer
ECMWF
&
Pascal Brunel and Jérôme Vidot
MétéoFrance

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.

Copyright 2019, EUMETSAT, All Rights Reserved.

Change record			
Version	Date	Author / changed by	Remarks
0.1	12/08/16	JH	First draft for v12 beta.
1.0	05/12/16	JH	Updates after beta.
1.1	31/01/17	JH	Updates after DRI.
1.2	03/04/18	JH	Updates for v12.2.
1.3	05/03/19	JH	Updates for v12.3.

TABLE OF CONTENTS

1. INTRODUCTION AND SCOPE 5

2. OVERVIEW OF RTTOV V12 5

3. SUMMARY OF RTTOV CONCEPTS 12

4. CHANGES FROM PREVIOUS VERSIONS 14

4.1. CHANGES FROM RTTOV V12.2 14
4.2. CHANGES FROM RTTOV V12.1 16
4.3. CHANGES FROM RTTOV V11 18

5. FORTRAN-90 UNIX/LINUX INSTALLATION AND TESTING 22

5.1 UNPACKING THE CODE 23
5.2 COMPILING THE CODE 25
5.3 RUNNING THE TEST SUITE 29

6. RTTOV TYPES, SUBROUTINES, LIBARARIES AND EXECUTABLES 33

6.1. LINKING YOUR CODE AGAINST RTTOV 33
6.2. DOXYGEN DOCUMENTATION 34

7. RUNNING RTTOV V12 FOR YOUR APPLICATIONS 37

7.1. SET RTTOV OPTIONS 38
7.2. INITIALISE COEFFICIENT STRUCTURES 38
7.3. SET UP INPUT PROFILES 40
7.4. SPECIFYING THE CHANNELS TO SIMULATE 46
7.5. SPECIFYING SURFACE EMISSIVITY 47
7.6. SPECIFYING SURFACE REFLECTANCE FOR SOLAR SIMULATIONS 50
7.7. ALLOCATION OF TRAJECTORY STRUCTURES. 52
7.8. OUTPUT ARRAYS FROM RTTOV V12 52
7.9. CALLING THE TANGENT LINEAR (TL), ADJOINT (AD) AND JACOBIAN (K) MODELS 56
7.10. MULTI-THREADED EXECUTION 58
7.11. RTTOV PERFORMANCE 58
7.12. SUMMARY OF STEPS FOR RUNNING RTTOV V12 60

8. DETAILS OF SPECIFIC RTTOV CAPABILITIES 61

8.1. SIMULATION OF CLEAR AIR RADIANCES FOR INFRARED AND MICROWAVE CHANNELS 61
8.2. SIMULATION OF CLEAR AIR RADIANCES FOR VISIBLE AND NEAR-INFRARED CHANNELS 61
8.3. SIMPLE CLOUD 63
8.4. DEFINITION OF SURFACE EMISSIVITY 63
8.5. SIMULATION OF VISIBLE AND IR CLOUD-AFFECTED RADIANCES 65
8.6. SIMULATION OF VISIBLE AND IR AEROSOL-AFFECTED RADIANCES 73
8.7. SIMULATION OF MICROWAVE RADIANCES SCATTERED BY CLOUD AND PRECIPITATION 76
8.8. SIMULATION OF HYPERSPECTRAL IR SOUNDER RADIANCES USING PC-RTTOV 78
8.9. SIMULATION OF HYPERSPECTRAL IR SOUNDER RADIANCES USING HTFRTC 80
8.10. INCLUSION OF NON-LOCAL THERMODYNAMIC EQUILIBRIUM EFFECTS 82
8.11. OPTION TO TREAT SURFACE AS A LAMBERTIAN REFLECTOR 83
8.12. ZEEMAN EFFECT FOR SSMIS AND AMSU-A 84
8.13. SIMULATION OF SSU RADIANCES 85
8.14. SIMULATIONS WITH VARIABLE SO_2 85

9. LIMITATIONS OF RTTOV V12 86

10. REPORTING AND KNOWN BUGS FOR RTTOV V12 86

11. FREQUENTLY ASKED QUESTIONS 86

12. GLOSSARY 87

13. REFERENCES 88

14. ANNEXES 92

Annex A - Coefficient information and conversion tools 92
1. RTTOV_COEF_INFO.EXE 92
2. RTTOV_MFASIS LUT_INFO.EXE 92
3. RTTOV_CONV_COEF_.EXE 92
4. RTTOV_ASCII2BIN_SCATTCOEF.EXE 94
5. RTTOV11_CONV_COEF_11TO12.EXE 94
6. RTTOV11_CONV_COEF_12TO11.EXE 94
Annex B – RTTOV_ERRORHANDLING interface 95
Annex C – Coefficient allocation and deallocation subroutines 96
1. RTTOV_READ_COEFS interface 96
2. RTTOV_READ_COEFS_HTFRTC interface 98
3. RTTOV_DEALLOC_COEFS interface 98
4. RTTOV_READ_SCATTCOEFFS interface 98
5. RTTOV_DEALLOC_SCATTCOEFFS interface 99
Annex D – RTTOV allocation/deallocation and initialisation subroutines 100
1. RTTOV_ALLOC_DIRECT interface 100
2. RTTOV_ALLOC_TL interface 102
3. RTTOV_ALLOC_AD interface 103
4. RTTOV_ALLOC_K interface 104
5. RTTOV_ALLOC_PROF interface 105
6. RTTOV_INIT_PROF interface 105
7. RTTOV_ALLOC_RAD interface 105
8. RTTOV_INIT_RAD interface 106
9. RTTOV_ALLOC_TRANSMISSION interface 106
10. RTTOV_INIT_TRANSMISSION interface 106
11. RTTOV_ALLOC_PCCOMP interface 107
12. RTTOV_INIT_PCCOMP interface 107
13. RTTOV_ALLOC_OPT_PARAM interface 108
14. RTTOV_INIT_OPT_PARAM interface 108
15. RTTOV_ALLOC_TRAJ interface 109
16. RTTOV_ALLOC_SCATT_PROF interface 110
17. RTTOV_INIT_SCATT_PROF interface 110
18. RTTOV_ALLOC_EMIS_RET_TERMS interface 110
19. RTTOV_SCATT_SETUPINDEX interface 111
Annex E – Optical parameter calculation subroutines 112
1. RTTOV_BPR_INIT interface 112
2. RTTOV_BPR_CALC interface 112
3. RTTOV_BPR_DEALLOC interface 112
4. RTTOV_LEGCOEF_CALC interface 113
Annex F – Emissivity atlas subroutines 114

1. RTTOV_SETUP_EMIS_ATLAS interface 114
2. RTTOV_GET_EMIS interface 115
3. RTTOV_DEALLOCATE_EMIS_ATLAS interface 117
Annex G – BRDF atlas subroutines 118
1. RTTOV_SETUP_BRDF_ATLAS interface 118
2. RTTOV_GET_BRDF interface 118
3. RTTOV_DEALLOCATE_BRDF_ATLAS interface 119
Annex H – RTTOV_GET_PC_PREDICTINDEX interface 120
Annex I – RTTOV_DIRECT interface 121
Annex J – RTTOV_K interface 123
Annex K – RTTOV_TL interface 125
Annex L – RTTOV_AD interface 127
Annex M – RTTOV_SCATT interface 129
Annex N – RTTOV Utility routines 132
1. RTTOV_USER_OPTIONS_CHECKINPUT interface 132
2. RTTOV_USER_PROFILE_CHECKINPUT interface 132
3. RTTOV_PRINT_OPTS interface 133
4. RTTOV_PRINT_OPTS_SCATT interface 133
5. RTTOV_PRINT_INFO interface 133
6. RTTOV_PRINT_PROFILE interface 134
7. RTTOV_PRINT_CLD_PROFILE interface 134
8. RTTOV_PRINT_RADIANCE_QUALITY interface 134
9. CREATE_AER_CLIM_PROF.EXE 135
10. RTTOV_AER_CLIM_PROF 135
11. RTTOV_ZUTILITY 136
12. RTTOV_OBS_TO_PC.EXE 138
13. RTTOV_SCATT_EMIS_RETRIEVAL 138
14. RTTOV_MAKE_SCAERCOEF.EXE 139
15. RTTOV_SCALE_REF_GAS_PROF 139
Annex O – RTTOV v12 derived types 140

1. Introduction and Scope

This document is structured as follows. Section 2 gives a broad overview of the RTTOV v12 fast radiative transfer model. Section 3 provides a brief introduction to the key concepts involved in running RTTOV v12 and it is recommended to all users. Section 4 provides details of all changes since RTTOV v11: this is intended to be a reference for users upgrading from v11. Section 5 gives the instructions for compiling RTTOV, verifying the build and, more generally, for running the test suite. Section 6 provides a reference for the RTTOV libraries, executables, subroutines and derived types. Section 7 provides a detailed step-by-step guide for implementing RTTOV in your own application. Section 8 gives details of the various simulation capabilities of RTTOV: you should consult the parts of this section relevant to your application. Section 9 lists the limitations of RTTOV v12. The procedure for reporting bugs or learning about known bugs is given in section 10. Finally a frequently asked questions (FAQ) section and glossary are provided in sections 11 and 12. This document relates to version 12 of the RTTOV code and all its sub-versions (12.x). The document is only updated with new minor releases of RTTOV v12: the document version is given in the header. To obtain a copy of the RTTOV v12 code go to http://nwpsaf.eu/site/register/ and register on the website. Once you are registered you can log-in and click on the "Software Downloads" link to subscribe to RTTOV v12: once you agree to the licence you will be able to download the code immediately.

RTTOV v12 is an evolution of RTTOV v11, adding and upgrading many features as documented here. The RTTOV v11 code is still available, but cannot be guaranteed to be upgraded for new instruments and capability. The format of the coefficient files has changed since RTTOV v11 and as such RTTOV v10/v11 coefficient files cannot be read by RTTOV v12 and vice versa. Exectuables are included in the package to convert RTTOV v10/v11-format *rtcoef* files to RTTOV v12 format and vice versa (where possible): these are described in Annex A. Existing RTTOV v11 coefficient files will remain available on the website. However the latest v12 coefficients provided with the package and available via the RTTOV v12 web page are the recommended ones for RTTOV v12.

The RTTOV v12 scientific and validation report describes or gives links to the scientific basis of the model and also describes in more detail any new scientific changes made. It also documents the test results carried out on the new code before delivery. The most up to date versions of these reports, including this user guide, can be viewed at the NWP SAF web site: http://nwpsaf.eu/site/software/rttov/ in pdf format on the RTTOV v12 page. There is also a RTTOV v12 performance report which documents the run times of RTTOV v12 on a few platforms and compares these to the equivalent RTTOV v11 run times.

The NWP SAF web forum can be found here: http://nwpsaf.eu/site/forums/. Information about RTTOV including bugs and updates is posted here and users are encouraged to post questions and comments about the software.

2. Overview of RTTOV v12

This section gives a brief overview of the RTTOV v12 model. More details can be found in the references given in this section. RTTOV v12 is a development of the fast radiative transfer model for TOVS, RTTOV, originally developed at ECMWF in the early 90's (Eyre, 1991) for TOVS. Subsequently the original code has gone through several developments (e.g. Saunders *et al.*, 1999; Matricardi *et al.*, 2001), more recently within the EUMETSAT NWP Satellite Application Facility (SAF), of which RTTOV v12 is the latest version (Saunders *et al.*, 2018). The model allows rapid simulations (~1 ms for 40 channel ATOVS on a desktop PC) of radiances for satellite visible, infrared or microwave nadir scanning radiometers given an atmospheric profile of temperature, variable gas concentrations, cloud and surface properties, referred to as the state vector. The only mandatory variable gas for RTTOV v12 is water vapour. Optionally ozone, carbon dioxide, nitrous oxide, methane, carbon monoxide and sulphur dioxide can be variable with all other constituents assumed to be constant. The state vector for RTTOV v12 is given in Annex O. RTTOV v12 can accept input profiles on any defined set of pressure levels. The range of temperatures and water vapour concentrations over which the optical depth computations are valid depends on the training datasets which were used. This is defined in the coefficient file and for RTTOV v12 is mainly based on the 91-level 83 diverse profile dataset from ECMWF analyses for temperature, water vapour and ozone (http://nwpsaf.eu/site/software/atmospheric-profile-data/). For other gases a range of profile datasets were used based on models and measurements. The limits for temperature, water vapour and ozone are given in Table 1. These are derived from the strict profile dataset minimum/maximum envelopes by applying a stretching factor (+/-10% for temperature max/min respectively, and +/-20% for each gas max/min). The RTTOV v12 coefficient files contain the *strict* min/max envelopes: the limits that are applied within RTTOV are calculated when the

coefficients are read in. More details on the profile datasets used for the different gases can be found in Matricardi (2008). For RTTOV v12 some trace gas datasets were stretched in order to improve their validity over the whole satellite era from 1970 and into the 2020s. This is described in the RTTOV v12 Science and Validation Report. The training profiles used for sulphur dioxide are also described in the Science and Validation Report. The majority of RTTOV v12 coefficient files are based on the 54 levels shown in Table 1, though coefficients for some hyperspectral sounders are also available on 101 levels.

The spectral range of the RTTOV v12 model in the visible/infrared is 0.4-50 microns (200 – 25000 cm^{-1}), governed by the range of the LBLRTM line-by-line datasets on which the coefficients are based. In the microwave the frequency range is from 10 – 800 GHz which is covered using the Liebe-89 MPM line-by-line model. The full list of currently supported platforms and sensors is given in Tables 2 and 3. This list is updated as new sensors are launched. For the visible/IR sensors, the channel order can either be decreasing or increasing with wavelength and in some cases (e.g. MTSAT imager) it is not monotonic wavelength order, rather being defined according to instrument convention. The channel order is indicated in Table 3. New or updated coefficient files will be made available from the RTTOV pages on the NWP SAF web site for the latest RTTOV versions. Coefficients for sensors that are not currently supported may be requested via the NWP SAF helpdesk.

An important feature of the RTTOV model is that it not only computes the forward (or direct) radiative transfer calculation but also the gradient of the radiances with respect to the state vector variables at the location in state space specified by the input state vector values. Given a state vector, **x,** a radiance vector, **y**, is computed:

$$\mathbf{y} = H(\mathbf{x}) \tag{1}$$

where H is the radiative transfer model (also referred to as the observation operator). The Jacobian matrix **H** gives the change in radiance $\boldsymbol{\delta y}$ for a change in any element of the state vector $\boldsymbol{\delta x}$ assuming a linear relationship about a given atmospheric state $\mathbf{x_0}$:

$$\boldsymbol{\delta y} = \mathbf{H}(\mathbf{x_0})\boldsymbol{\delta x} \tag{2}$$

The elements of **H** contain the partial derivatives $\partial \mathbf{y}_i/\partial \mathbf{x}_j$ where the subscript i refers to channel number and j to position in state vector. The Jacobian gives the top-of-atmosphere radiance change for each channel given unit perturbations at each respective level of the profile vectors and in each of the surface/cloud parameters. It shows clearly, for a given profile, which layers in the atmosphere are most sensitive to changes in temperature and variable gas concentrations for each channel. **rttov_k** (and its associated subroutines ending in *K*) compute the $\mathbf{H(x_0)}$ matrix for each input profile.

It is not always necessary to store and access the full Jacobian matrix **H** and so the *RTTOV* package has routines to only output the *tangent linear* values $\boldsymbol{\delta y}$**,** the change in top of atmosphere radiances y_n for each channel n, for a given change in atmospheric profile, $\boldsymbol{\delta x}$**,** about an initial atmospheric state $\boldsymbol{x_0}$.

$$\delta y(x_0) = \left[\delta x \frac{\partial y_1}{\partial x}, \delta x \frac{\partial y_2}{\partial x}, \delta x \frac{\partial y_3}{\partial x}, ...\delta x \frac{\partial y_{nchan}}{\partial x} \right] \tag{3}$$

The tangent linear routines all have *TL* as an ending. Conversely the adjoint routines (ending in *AD*) compute the change in any scalar quantity up to *nel* elements of the state vector (e.g. T, q, ozone, surface variables etc) $\boldsymbol{\delta x}$ for an assumed atmospheric state, $\mathbf{x_0}$**,** given a change in the radiances, $\delta \boldsymbol{y}$.

$$\delta x(x_0) = \left[\delta y \frac{\partial x_1}{\partial y}, \delta y \frac{\partial x_2}{\partial y}, \delta y \frac{\partial x_3}{\partial y}, ...\delta y \frac{\partial x_{nel}}{\partial y} \right] \tag{4}$$

These routines are normally used as part of the variational assimilation of radiances. Information about the RTTOV TL, AD and K models is given in section 7.9 below and some general information on TL/AD and K codes is available at: http://cimss.ssec.wisc.edu/itwg/groups/rtwg/fastrt.html. If you only want to compute radiances with the forward model the *TL/AD/K* routines are not required.

The core of RTTOV simulates clear-sky radiances. For channels with a significant thermally emitted contribution, the top of the atmosphere upwelling clear-sky radiance, $L^{Clr}(\nu,\theta)$, at a frequency ν and viewing angle θ can be written as:

$$L^{Clr}(\nu,\theta) = \tau_s(\nu,\theta)\varepsilon_s(\nu,\theta)B(\nu,T_s) + \int_{\tau_s}^{1} B(\nu,T)d\tau + (1-\varepsilon_s(\nu,\theta))\tau_s^2(\nu,\theta)\int_{\tau_s}^{1}\frac{B(\nu,T)}{\tau^2}d\tau \quad (5)$$

where τ_s is the surface to space transmittance, ε_s is the surface emissivity and $B(\nu,T)$ is the Planck function for a frequency ν and temperature T.

The transmittances, τ, are computed by means of a linear regression in optical depth based on variables from the input profile vector as described in Matricardi et al. (2001) for RTTOV v7 predictors, Matricardi (2003) for RTTOV v8 predictors and those given in Matricardi (2005) or the RTTOV v9 science plan for RTTOV v9 predictors. The code supports any of these predictor sets with the selection being made according to the coefficient file supplied to the program. Table 4 gives more information on the coefficient files available for each instrument type and the underlying predictor sets. More details on the performance of the different predictor sets are given in the RTTOV v9 Science and Validation Report. No changes to the optical depth predictors were made for RTTOV v12, but SO_2 has been added as a new optional variable gas as described in the RTTOV v12 Science and Validation Report.

In addition to the clear-air simulations described above (and in section 8.1) there are options to include solar radiation (section 8.2), for a simple visible/IR cloud scheme based on a single cloud top pressure and cloud fraction (section 8.3), for visible/infrared cloud- and aerosol-affected radiances (sections 8.5 and 8.6) and for cloud- and precipitation-affected microwave radiances (section 8.7).

Level Number	Pressure hPa	Tmax K	Tmin K	Qmax ppmv*	Qmin ppmv*	O_3max ppmv*	O_3min ppmv*	O_3Ref ppmv*
1	0.01	245.95	143.66	5.24	0.91	1.404	0.014	0.296
2	0.01	252.13	154.19	6.03	1.08	1.410	0.069	0.321
3	0.03	263.71	168.42	7.42	1.35	1.496	0.108	0.381
4	0.06	280.12	180.18	8.10	1.58	1.670	0.171	0.527
5	0.13	299.05	194.48	8.44	1.80	2.064	0.228	0.769
6	0.23	318.64	206.21	8.59	1.99	2.365	0.355	1.074
7	0.41	336.24	205.66	8.58	2.49	2.718	0.553	1.471
8	0.67	342.08	197.17	8.34	3.01	3.565	0.731	1.991
9	1.08	340.84	189.50	8.07	3.30	5.333	0.716	2.787
10	1.67	334.68	179.27	7.89	3.20	7.314	0.643	3.756
11	2.50	322.50	176.27	7.75	2.92	9.191	0.504	4.864
12	3.65	312.51	175.04	7.69	2.83	10.447	0.745	5.953
13	5.19	303.89	173.07	7.58	2.70	12.336	1.586	6.763
14	7.22	295.48	168.38	7.53	2.54	12.936	1.879	7.109
15	9.84	293.33	166.30	7.36	2.46	12.744	1.322	7.060
16	13.17	287.05	163.47	7.20	2.42	11.960	0.719	6.574
17	17.33	283.36	161.49	6.96	2.20	11.105	0.428	5.687
18	22.46	280.93	161.47	6.75	1.71	9.796	0.278	4.705
19	28.69	282.67	162.09	6.46	1.52	8.736	0.164	3.870
20	36.17	279.93	162.49	6.14	1.31	7.374	0.107	3.111
21	45.04	273.15	164.66	5.90	1.36	6.799	0.055	2.478
22	55.44	265.93	166.19	6.21	1.30	5.710	0.048	1.907
23	67.51	264.70	167.42	9.17	1.16	4.786	0.043	1.440
24	81.37	261.95	159.98	17.89	0.36	4.390	0.038	1.020
25	97.15	262.43	163.95	20.30	0.01	3.619	0.016	0.733
26	114.94	259.57	168.59	33.56	0.01	2.977	0.016	0.604
27	134.83	259.26	169.71	102.24	0.01	2.665	0.016	0.489
28	156.88	260.13	169.42	285.00	0.01	2.351	0.013	0.388
29	181.14	262.27	170.63	714.60	0.01	1.973	0.010	0.284
30	207.61	264.45	174.11	1464.00	0.01	1.481	0.013	0.198
31	236.28	270.09	177.12	2475.60	0.01	1.075	0.016	0.145
32	267.10	277.93	181.98	4381.20	0.01	0.774	0.015	0.110
33	300.00	285.18	184.76	6631.20	0.01	0.628	0.015	0.086
34	334.86	293.68	187.69	9450.00	1.29	0.550	0.016	0.073
35	371.55	300.12	190.34	12432.00	1.52	0.447	0.015	0.063
36	409.89	302.63	194.40	15468.00	2.12	0.361	0.015	0.057
37	449.67	304.43	198.46	18564.00	2.36	0.284	0.015	0.054
38	490.65	307.20	201.53	21684.00	2.91	0.247	0.015	0.052
39	532.58	312.17	202.74	24696.00	3.67	0.199	0.015	0.050
40	575.15	315.55	201.61	27480.00	3.81	0.191	0.012	0.050
41	618.07	318.26	189.95	30288.00	6.82	0.171	0.010	0.049
42	661.00	321.71	189.95	32796.00	6.07	0.128	0.009	0.048
43	703.59	327.95	189.95	35328.00	6.73	0.124	0.009	0.047
44	745.48	333.77	189.95	37692.00	8.71	0.117	0.008	0.046
45	786.33	336.46	189.95	39984.00	8.26	0.115	0.008	0.045
46	825.75	338.54	189.95	42192.00	7.87	0.113	0.008	0.043
47	863.40	342.55	189.95	44220.00	7.53	0.111	0.007	0.041
48	898.93	346.23	189.95	46272.00	7.23	0.108	0.006	0.040
49	931.99	349.24	189.95	47736.00	6.97	0.102	0.006	0.038
50	962.26	349.92	189.95	51264.00	6.75	0.099	0.006	0.034
51	989.45	350.09	189.95	49716.00	6.57	0.099	0.006	0.030
52	1013.29	350.09	189.95	47208.00	6.41	0.094	0.006	0.028
53	1033.54	350.09	189.95	47808.00	6.29	0.094	0.006	0.027
54	1050.00	350.09	189.95	47640.00	6.19	0.094	0.006	0.027

*The gas units here are ppmv with respect to **dry** air because RTTOV is trained on gas profiles in these units and this is the context in which these limits are applied within RTTOV.

Table 1. Pressure levels adopted for RTTOV v12 54 level coefficients and profile limits within which the transmittance calculations are valid. The default ozone profile is also given in the right hand column. Note that some coefficients are based on a standard 101 level pressure profile which extends down to 1100 hPa.

Platform	RTTOV ID	Sat ID range	Platform	RTTOV ID	Sat ID range
NOAA*	1	1 - 20	Himawari	31	8 - 9
DMSP	2	8 - 19	MTG	32	1
Meteosat	3	1 - 7	Saral	33	1
GOES	4	4 - 19	Metop-SG	34	1
GMS	5	1 - 5	Landsat	35	4,5,7,8
FY2	6	2 - 5,7	Jason	36	2
TRMM	7	1	GPM	37	1
ERS	8	1 - 2	*INSAT-1*	*38*	*1 - 4*
EOS	9	1 - 2	*INSAT-2*	*39*	*1 - 5*
METOP	10	1 - 2	INSAT-3	40	4 - 5
ENVISAT	11	1	Reserved	41	-
MSG	12	1 - 4	*DSCOVR*	*42*	*1*
FY1	13	3 - 4	CLARREO	43	1
ADEOS	*14*	*2*	TICFIRE	44	1
MTSAT	15	1 - 2	Reserved	45	-
CORIOLIS	16	1	ISS	46	1
JPSS (SNPP)	17	0	HJ1	47	2
GIFTS	*18*	*1*	GEOKOMPSAT2	48	1
Sentinel3	19	1	GCOM-C	49	1
MeghaTropique	20	1	SMOS	50	1
Kalpana	*21*	*1*	ORS	51	6
Meteor	22	25	FY4	52	1
FY3	23	1 - 4	TROPICS	53	0
COMS	24	1	GF5	54	1
METEOR-M	25	1 - 2	HY2	55	1
GOSAT	*26*	*1*	CloudSat	56	1
CALIPSO	27	1	*CloudCore*	*57*	*1*
Reserved	28	-	*FORUM*	*58*	*1*
GCOM-W	29	1	*TEMPEST-D*	*59*	*1*
Nimbus	30	3 - 7			

*"NOAA-5" is TIROS-N

Table 2. Platforms supported by RTTOV as of March 2019. Platforms in italics are not yet supported in the RTTOV v12 distribution but can be requested.

Sensor	RTTOV ID	Sensor Chans => RTTOV v12 Chans (IR/MW-only)	Sensor Chans => RTTOV v12 Chans (VIS/NIR/IR)
HIRS	0	1-19 => 1-19	-
MSU	1	1-4 => 1-4	-
SSU	2	1-3 => 1-3	-
AMSU-A	3	1-15 => 1-15	-
AMSU-B	4	1-5 => 1-5	-
AVHRR	5	3b-5 => 1 to 3	1-6 => 1-6
SSMI	6	1-7 => 1-7	-
VTPR1	7	1-8 => 1-8	-
Spare	8	-	-
TMI	9	1-9 => 1-9	-
SSMIS	10	1-24 => 1-24*	-
AIRS	11	1-2378 => 1-2378	1-2378 => 1-2378
HSB	12	1-4 => 1-4	-

MODIS	13	(20-25, 27-36) => 1-16	1-36 => 1-36
ATSR	14	1-3 => 1-3	1-7 => 1-7
MHS	15	1-5 => 1-5	-
IASI	16	1-8461 => 1-8461	1-8461 => 1-8461
AMSR-E	17	1-12 => 1-12	-
GMS imager	18	1 => 1 1-3 => 1-3	1-2 => 1-2 1-4 => 1-4
ATMS	19	1-22 => 1-22	-
MVIRI	20	1-2 => 1-2	-
SEVIRI	21	4-11 => 1-8	1-12 => 1-12
GOES imager	22	2-5 => 1-4	1-5 => 1-5
GOES sounder	23	1-18 => 1-18	-
MTSAT imager	24	1-4 => 1-4	1-5 => 1-5
FY2-3/4 VISSR	25	2-5 => 1-4	1-5 => 1-5
FY1 MVISR	26	1-3 => 1-3	-
CrIS	27	1-1305 => 1-1305	1-1305 => 1-1305
CrIS-FSR	28	1-2211 => 1-2211	1-2211 => 1-2211
VIIRS	29	16-22 => 1-7	1-22 => 1-22
WINDSAT	30	1-16 => 1-16	-
GIFTS	*31*	-	-
SSM-T1	*32*	*1-7 => 1-7*	-
SSM-T2	33	1-5 => 1-5	-
SAPHIR	34	1-6 => 1-6	-
MADRAS	35	1-9 => 1-9	-
Reserved	36	-	-
VHRR	*37*	*2-3 => 1-2*	*1-3 => 1-3*
INSAT imager	38	3-6 => 1-4	1-6 => 1-6
INSAT sounder	39	1-18 => 1-18	1-19 => 1-19
MWTS	40	1-4 => 1-4	-
MWHS	41	1-5 => 1-5	-
IRAS	42	1-20 => 1-20	1-26 => 1-26
MWRI	43	1-10 => 1-10	-
ABI	44	7-16 => 1-10	1-16 => 1-16
COMS MI	45	2-5 => 1-4	1-5 => 1-5
MSUMR	46	1-3 => 1-3	-
TANSO-FTS	*47*	-	-
IIR	48	1-3 => 1-3	-
ESA MWR	49	1-2 => 1-2	-
Reserved	50-53	-	-
SCAMS	54	1-5 => 1-5	-
SMMR	55	1-10 => 1-10	-
AHI	56	7-16 => 1-10	1-16 => 1-16
MTG IRS	57	1-1738 => 1-1738	1-1738 => 1-1738
AltiKa	58	1-2 => 1-2	-
IASI-NG	59	1-16921 => 1-16921	1-16921 => 1-16921
Landsat TM	60	6 => 1	-
MTG FCI	61	9-16 => 1-8	1-16 => 1-16
AMSR1	*62*	*1-16 => 1-16*	-
AMSR2	63	1-14 => 1-14	-
FY2-2 VISSR	64	1-2 =>1-2	-
SLSTR	65	7-9 => 1-3	1-9 => 1-9
TIRS	66	10-11 => 1-2	-
AMR	67	1-3 => 1-3	-
OLI	68	-	1-9 => 1-9
IRIS	69	1-862 => 1-862	-
ICI	70	1-13 => 1-13	-
GMI	71	1-13 => 1-13	-
MWTS-2	72	1-13 => 1-13	-
MWHS-2	73	1-15 => 1-15	-

ASTER	74	10-14 => 1-5	(1-2,3N,3B,4-14) => 1-15
Reserved	75	-	-
MTVZA-GY	76	1-29 => 1-29	-
MetImage	77	11-20 => 1-9	1-20 => 1-20
MWS	78	1-24 => 1-24	-
MWI	79	1-26 => 1-26	-
EPIC	*80*	-	-
MRIR	81	2-5 => 1-4	-
SI	82	1-579 => 1-579	1-579 => 1-579
Reserved	83-86	-	-
MERSI-1	87	20 => 1	-
MERSI-2	88	20-25 => 1-6	1-25 => 1-25
ECOSTRESS	89	1-5 => 1-5	-
IRMSS	90	4 => 1	-
OLCI	91	-	See coef file headers
THIR	92	1-2 => 1-2	-
AMI	93	7-16 => 1-10	1-16 => 1-16
IKFS2	94	1 - 2701 => 1-2701	1 - 2701 => 1-2701
LI	95	-	1-2 => 1-2
SGLI	96	18-19 => 1-2	-
HIRAS	*97*	-	-
GIIRS	*98*	-	-
AGRI	99	7/8-14 => 1-7 (coefs include only one 3.9um channel)	1-7/8,9-14 => 1-13 (coefs include only one 3.9um channel)
PMR	100	-	-
MIRAS	101	1-2 => 1-2	-
COWVR	102	1-12 => 1-12	-
TROPICS	103	1-12 => 1-12	-
VIMS	104	9-12 => 1-4	-
DPR	105	1-4 => 1-4	-
HY2 MWRI	106	1-9 => 1-9	-
CPR	107	1 => 1	-
SAPHIR NG	108	-	-
FORUM	*109*	-	-
TEMPEST-D	*110*	-	-
Reserved	111	-	-

*channels 19-21 are only simulated accurately with Zeeman coefficient file

The channel order in the coefficient files can be determined using the **rttov_coef_info.exe** executable (see Annex A), or by looking at the following web page which gives the spectral response functions and passbands used by RTTOV and the linked pages show the channel order in the coefficient files:
https://nwpsaf.eu/site/software/rttov/download/coefficients/spectral-response-functions/

Table 3. Instruments supported by RTTOV v12 as of March 2019. Sensors in italics are not yet supported in the RTTOV v12 distribution but can be requested. "IR/MW only" refers to v7/v8 predictor files; "VIS/NIR/IR" refers to v9 predictor solar-compatible files.

3. Summary of RTTOV concepts

This section provides a brief overview of the steps required when running RTTOV. Existing users should read this section for information on coefficient files. New users should read this section to familiarise themselves with the key concepts related to RTTOV. You should also look through the code contained in **src/test/example_fwd.F90** and/or **src/test/example_k.F90** as these provide fully-commented templates for running the RTTOV forward and K models. The **src/test/** directory also contains examples for running other kinds of simulation. A detailed description of the steps involved in running RTTOV is given in section 7. There is an "RTTOV Quick Start Guide" in the **docs/** directory which is intended to guide new users through a simple clear-sky forward model simulation.

All aspects of RTTOV simulations can be configured at run-time via the **rttov_options** structure (described fully in Annex O). This provides a list of logical flags and other settings to control various aspects of the simulation such as whether solar radiation should be included, whether clouds and/or aerosol scattering should be included for IR instruments, and whether the RTTOV interpolator should be used.

Calculation of clear-sky optical depths is carried out by a predictor-based regression scheme. The coefficients for the optical depth regression are instrument-specific and are stored in RTTOV coefficient files whose names begin "*rtcoef_*". There are three flavours of predictors denoted "v7", "v8" and "v9" which determine the variable trace gases available for the simulation and whether solar simulations are available. The coefficients for v9 predictors are trained over a wider range of zenith angles to allow for solar simulations which means the maximum allowable zenith angle is larger (potentially relevant for geostationary satellites) compared to v7 and v8 predictors. The predictor versions are summarised in Table 4.

The latest RTTOV coefficients have been trained using an updated profile dataset (see the RTTOV v12 Science and Validation Report). The reference profiles for trace gases (CO2, CH4 and N2O in particular) are appropriate for contemporary simulations. When simulating older instruments you may wish to specify a more appropriate CO2 profile. Coefficients are now available with v8 predictors (variable ozone and CO2) for all IR sensors that enable you to vary the CO2 profile. In addition the v9 predictor coefficients for visible/IR sensors now have variable ozone and CO2. Previously they only allowed variable ozone: the new coefficients reproduce radiances from the line-by-line model as well as or better than the ozone-only coefficients. The only exception to this is the 3.9µm channel for SEVIRI so ozone-only SEVIRI v9 predictor coefficients are available (with "o3" in the filename).

RTTOV accounts for thermal emission for all channels at wavelengths above 3µm (referred to as "thermal" channels). Thermal emission is ignored at wavelengths below 3µm. Solar radiation is only included in channels at wavelengths below 5µm. These are referred to as "solar" channels. Channels below 3µm are solar-only channels, while those in the range 3-5µm are mixed thermal+solar channels.

Important notes on v7/v8 vs v9 predictor coefficients:

- Where both v7/v8 and v9 predictor files exist for an instrument (see Table 3), the v7/v8 predictor files support *only* the IR channels, while the v9 predictor file supports *all* visible, near-IR and IR instrument channels with wavelengths above 0.4µm.
- The RTTOV channel numbering in the coefficient files begins at one. The v7 and v8 predictor files have the same channel numbering, but in many cases the channel numbering for the IR channels is *not the same* for the v7/v8 and v9 predictor files. Table 3 provides information on how the instrument channel numbers map to the RTTOV channel numbers and this can also be found by examining the headers of the coefficient file.
- Simulated IR radiances for the same options and profile will show small differences when different predictor versions are used. It is entirely reasonable to use the v9 predictor coefficients for simulating IR channels only. However it is not the case that, for example, v9 predictors are generally superior to v7 or v8 predictors. For IR-only simulations you may find the v7 or v8 predictors are the optimal choice. There are plots illustrating the fit of the optical depth regression in terms of BT for each coefficient file on the RTTOV web site: https://www.nwpsaf.eu/site/software/rttov/download/coefficients/comparison-with-lbl-simulations/

The accuracy of simulations for very broad channels (e.g. SEVIRI channel 4 at 3.9 microns) is poor with significant biases noted (~1-2K) (see e.g. Brunel and Turner, 2003). To mitigate this, the line-by-line optical depths in the coefficient generation are weighted with the Planck function across the instrument channel and the coefficients are then computed for these Planck-weighted optical depths resulting in much reduced biases. Whether coefficients are Planck-weighted or not for a channel can be determined by examining the PLANCK_WEIGHTED section in the coefficient file (if it is not present there are no Planck-weighted channels).

The following page provides the definitive reference for RTTOV v12 coefficient files:
http://nwpsaf.eu/site/software/rttov/download/coefficients/coefficient-download/

The coefficients files are read in using the **rttov_read_coefs** subroutine. If Principal Component (PC), cloud or aerosol simulations are being performed, the additional associated coefficients for those simulations are read in the same call.

You must then allocate some derived types or structures to hold various input and output quantities. These structures are all described fully in Annex O. The main ones are:

- **chanprof** structure – array holding a list of channel and profile indices to simulate
- **profile** structure – array holding the input atmospheric and surface variables
- **emissivity** structure – array holding the surface emissivity
- **reflectance** structure – array holding the surface reflectance
- **radiance** structure – to hold the output simulated radiances
- **transmittance** structure – to hold the output simulated transmittances

In a typical application you would then loop over input profiles. For each profile, the input profile data is read into the profile, emissivity and reflectance structures. Then RTTOV is called and the simulated radiances are written out or stored. It is possible to pass multiple profiles into RTTOV in a single call if desired. There is no fixed limit on the number of profiles you can pass into RTTOV in one go: this is limited only by the memory available (some kinds of simulations take more memory than others). Once all simulations are complete, you should call a number of deallocation subroutines to release allocated memory.

	v7 predictors	**v8 predictors**	**v9 predictors**
Max zenith angle	75 degrees	75 degrees	85* degrees
Solar computations?	No	No	Yes
MW instruments	All MW sensors. Coefs on 54 levels except for Zeeman coef files. Variable gases: H_2O	N/A	N/A
Hyperspectral sounders	Coefs on 54 and 101 levels Variable gases: H_2O, O_3	Coefs on 101 levels Variable gases: H_2O, O_3, CO_2	Coefs on 101 levels Variable gases: H_2O, O_3, CO_2, CO, NO_2, CH_4, SO_2
IR instruments (including those with visible/near-IR channels)	All IR instruments except SSU, PMR IR channels only Coefs on 54 levels Variable gases: H_2O, O_3	All IR instruments IR channels only Coefs on 54 levels (SSU on 51 levels) Variable gases: H_2O, O_3, CO_2	Selected instruments (see final column in Table 3) Coefs on 54 levels Visible/near-IR/IR channels Variable gases: H_2O, O_3, CO_2

Table 4. Flavours of RTTOV coefficients.

**For some instruments the coefficients are trained up to a slightly smaller maximum angle as it improves the overall accuracy of the optical depth calculations: see the comments in the coefficient file headers.*

4. Changes from previous versions

4.1. Changes from RTTOV v12.2

This section provides details of the differences between RTTOV v12.2 and v12.3. Users will be able to switch v12.3 for v12.2 in their Fortran application without having to make any code changes and the outputs will be the same. Users of the C++/Python wrapper will have to make minor modifications to their existing code to make it work with v12.3 (see below). Users running MFASIS or HTFRTC simulations will need to download new coefficient files for use with v12.3. RTTOV v12.3 provides a number of new options and capabilities: these are summarised here and described more fully elsewhere in the user guide.

Main new/updated features in v12.3

Clear-sky/general simulation updates:

- New CAMEL climatology atlas for IR emissivities: emissivity data based on 2000-2016 climatology, improved treatment of snow, and includes emissivity standard deviations computed from multi-year climatology (see sections 7.5, 8.4 and Annex G).
- New option **opts%rt_all%lambertian_fixed_angle** to switch between fixed or parameterised effective angle for downwelling radiance with the Lambertian surface option (see section 8.11).
- A new **specularity** profile skin variable has been added which specifies the weighting for a linear combination of specular and Lambertian downwelling radiances with the Lambertian surface option (see section 8.11).
- A new subroutine has been added (**rttov_scale_ref_gas_prof**) which makes it easy to use scaled copies of the RTTOV reference profiles for optional trace gases in simulations (see Annex N).

Visible/IR scattering updates: (sections 8.5 and 8.6)

- MFASIS has been modified to allow for variable water vapour in affected sensor channels (currently applies to SEVIRI 0.8 micron channel only) which greatly improves the fit to the training data. New MFASIS LUTs for all sensors must be downloaded from the RTTOV coefficients web page for use with v12.3.
- An updated parameterisation of the Baran ice cloud property database has been added for visible/IR cloudy simulations which is more spectrally consistent and extends the parameterisation to the far-IR.
- A new option **opts%rt_ir%grid_box_avg_cloud** has been added. If false (the default) then input cloud concentrations for visible/IR cloud simulations should represent the concentration of cloud in the cloudy fraction of the layer (the behaviour of previous versions of RTTOV). If the option is true cloud concentrations for visible/IR cloud simulations should be layer grid-box-average values (consistent with RTTOV-SCATT), which is commonly the quantity output by GCMs. This does not affect simulations with **opts%rt_ir%user_cld_opt_param** true.
- A new executable has been created (**rttov_make_scaercoef.exe**) which enables the generation of custom aerosol optical property files for use with RTTOV (see Annex N).

Technical updates:

- The wrapper has been updated to support the new options, the **specularity** profile variable, and to enable simulations with user-generated aerosol optical property files. **NB** The **specularity** profile variable has been added to the "skin" variable array for the **Profile** and **Profiles** classes and this means that user code which calls the RTTOV v12.2 wrapper will need minor updates to work with v12.3. See the wrapper user guide.
- The GUI has been updated to support the new code options, the new **specularity** profile variable, and to enable choice among the available emissivity atlases.

HTFRTC updates (see section 8.9):

- all RTTOV trace gases may now optionally vary in the simulation; ozone is now optional
- options to calculate overcast radiances and simple cloud (as RTTOV does)
- option to supply emissivities; IR emissivity atlas interface has been updated to work with HTFRTC
- support for all RTTOV gas units options, **do_lambertian** option with new **specularity** profile variable, and the **plane_parallel** option
- multi-threaded simulations via parallel interfaces enabled
- NB a new static coefficient file must be downloaded from the coefficients web page for use with v12.3.

Additional changes since v12.1

Changes in behaviour

The CrIS FSR instrument ID has changed from 27 to 28.

When **apply_reg_limits** is true, RTTOV does not print out warnings when input profiles exceed regression limits (this reverts to the behaviour of v12.1 and earlier).

The **rttov_user_profile_checkinput** subroutine no longer warns when ice water contents are above zero, but below the minimum value for the selected **ice_scheme** (such small values are unlikely to be problematic).

The **rttov_print_opts** subroutine output is more concise: irrelevant options are not printed (e.g. cloud-related options are not printed if **addclouds** is false).

Some internal changes in RTTOV have been made to improve the validity checks on input data: in particular the checking of input profile data for unphysical values now includes tests on cloud/aerosol variables (these were not checked previously).

Other new options

The number of PC scores to compute for PC-RTTOV may now be specified in a new option **opts%rt_ir%pc%npcscores**. If the value of this option is <1 then RTTOV derives **npcscores** from the size of the arrays in **pccomp** which is the previous behaviour. The advantage of using this option is that **pccomp** can be sized to the maximum required, but RTTOV can then be called on fewer profiles without having to re-size **pccomp**. This is a common scenario when passing batches of different numbers of profiles to RTTOV.

A new option **opts%rt_all%rad_down_lin_tau** has been added which switches between linear-in-tau and layer-average radiance expressions for the downwelling specular radiance calculation. The difference in TOA radiances between the two options is very small (order 1mK), but the layer-average is slightly faster and the code is simpler and is consistent with Lambertian downwelling term which *always* uses the layer-average source term. The removal of the linear-in-tau code for downwelling radiances is under consideration for v13.

A new option **opts%rt_mw%clw_cloud_top** has been added which allows the lower pressure limit for MW CLW absorption calculations to be changed. Previously this was hard-coded at 322hPa above which input CLW values were ignored, and this remains the default (see section 7.3).

A new option **opts%rt_ir%rayleigh_single_scatt** has been added which allows the Rayleigh single-scattering calculation to be enabled/disabled for visible channels (the TOA radiances still include extinction due to Rayleigh scattering regardless of this option).

A new option **opts%rt_ir%cldstr_low_cloud_top** has been added which allows the upper pressure limit for the **opts%rt_ir%cldstr_simple** option to be changed for visible/IR cloud simulations. This setting applies to *all* channels being simulated. Both options should be used with caution: the **cldstr_simple** option is primarily intended for mid- to upper-tropospheric channels. (See Annex O for a description of these options in the options structure).

New RTTOV-SCATT option **opts_scatt%cc_threshold** which specifies the effective cloud fraction threshold below which scattering is ignored. Previously this was hard-coded at 0.05, and this remains the default. See section 8.7.

Bug fixes

The bugs in RTTOV v12.2 listed on the website have been fixed:
https://www.nwpsaf.eu/site/software/rttov/rttov-v12/code-updates/

4.2. Changes from RTTOV v12.1

This section provides details of the differences between RTTOV v12.1 and v12.2. The vast majority of users will be able to switch v12.2 for v12.1 in their application without having to make any code changes and the outputs will be the same aside from certain specific differences which are described below. Users of HTFRTC will have to update their code for v12.2 as the interface for HTFRTC calls has changed. RTTOV v12.2 provides a number of new options and capabilities: these are summarised here and described more fully elsewhere in the user guide.

Main new/updated features in v12.2

Clear-sky/general simulation updates:

- New option for solar sea surface BRDF model (section 7.6).
- New options for MW cloud liquid water (CLW) absorption calculations: new permittivity parameterisations and new option to perform calculations on user levels rather than coefficient levels (section 7.3).
- MW optical depth coefficient files now contain band correction coefficients.

RTTOV-SCATT (section 8.7):

- New option to carry out calculations on radiances instead of brightness temperatures.
- New optional output structure containing information required to perform all-sky emissivity retrievals. A new subroutine is included which carries out the retrieval calculation.
- New options for CLW permittivity in "Mietable" generation code (same options as for CLW absorption) and updated Mietable files using new recommended option.
- New options for non-spherical particle optical properties in "Mietable" generation code: inclusion of the ARTS single scattering database including many options for representing ice, snow, hail and graupel.
- New Mietable file for MetopSG ICI.
- New OpenMP parallel interface to RTTOV-SCATT.

Visible/IR scattering simulations:

- New option to use MFASIS fast visible cloud scattering parameterisation (section 8.5).
- New cloud liquid water optical properties parameterised in terms of particle effective diameter (section 8.5).
- Explicit cloud/aerosol optical properties are optionally active variables in the TL/AD/K models (section 7.9).
- New *scaer* files containing optical properties for 9 CAMS aerosol species (section 8.6).

Principal Components simulations:

- New PC-RTTOV coefficients enabling all variable trace gases (except SO2) and aerosol simulations using OPAC aerosol properties (section 8.8).
- The HTFRTC interface has been rewritten to improve performance (section 8.9).

Technical updates:

- Python wrapper is now compatible with both Python 2 and 3. The wrapper has also been updated to enable MFASIS simulations and to work with the new CAMS aerosol species (see docs/rttov-wrapper.pdf).
- The RTTOV GUI has been updated to enable MFASIS simulations and to work with the new CAMS aerosol species (see docs/rttov_gui_v12.pdf)
- Build system updated to optionally allow compilation against an external LAPACK library (section 5.2).

Additional changes since v12.1

Changes in behaviour and interfaces

In order to enable the TL/AD/K of the explicit aerosol/cloud optical properties, new optional arguments were added to the **rttov_tl, rttov_ad, rttov_k, rttov_parallel_tl, rttov_parallel_ad, rttov_parallel_k, rttov_alloc_tl, rttov_alloc_ad,** and **rttov_alloc_k** subroutines. The vast majority of user code will be unaffected, but it is possible that you will have to modify your code to specify optional arguments to these subroutines by name.

Existing user code for RTTOV v12.1 using the object oriented wrapper (the "Rttov" and similar C++ classes or the pyrttov package) will work with v12.2. However to support the new capabilities the underlying Fortran interface has changed so user code which calls the wrapper interface subroutines directly will need to be updated for v12.2.

There is a change in behaviour related to the TL/AD/K of surface emissivity and BRDF when physical emissivity/BRDF models are used. This affects the TL/AD/K when using FASTEM (any version, all surface types), TESSEM2 (sea surfaces), IREMIS (sea surfaces), the solar BRDF model (sea surfaces). In v12.1 any non-zero emissivity or BRDF TL input was used in preference to the TL calculated by each of these models from the input **profiles_tl**. RTTOV v12.2 *always* computes the TL interally in these cases and ignores any input **emissivity_tl** and **reflectance_tl**. In v12.1 RTTOV calculated emissivity and reflectance AD/K outputs for these cases. In v12.2 the **emissivity_ad/k** and **reflectance_ad/k** outputs are always zero for these cases representing the fact that RTTOV computed the surface emissivity/BRDF from the input profile variables. This does not affect other cases, for example when **calcemis/calcrefl** are false or when using ISEM.

New outputs and options

There are new outputs in the **rttov_radiance** structure. The **radiance%plane_parallel** logical is set to true if RTTOV used a strict plane-parallel geometry (true for example for DOM or MFASIS simulations). The **radiance%quality(:)** integer array has one element per simulated radiance and is used as a bit-mask to flag possible issues with radiances such as when optical depth regression limits are exceeded (see section 7.8). There is a new subroutine **rttov_print_radiance_quality** which prints a human-readable interpretation of a given quality output (see Annex N).

In previous versions of RTTOV the input surface elevation (**profiles(:)%elevation**) was associated with the pressure level equal to or immediately below the surface pressure (**profiles(:)%s2m%p**). The geopotential height of each pressure level is computed relative to this fixed point. There is a new option **opts%config%fix_hgpl** which, if true, associates the surface elevation with the 2m pressure regardless of whether or not it lies on a pressure level. Strictly speaking this is more correct and it eliminates certain very small discontinuities in forward model radiances as the 2m pressure varies. The impact on forward model radiances is small (up to ~0.01K, usually much smaller).

RTTOV-SCATT

It is recommended to use the new **opts_scatt%lradiance** option to carry out computations in radiance. The RTTOV-SCATT code has also been updated to apply band corrections.

The **lgradp** switch to make pressure an active TL/AD/K variable is now an option for RTTOV-SCATT.

There are new subroutines **rttov_print_opts_scatt** and **rttov_print_cld_profile** which can be used to print out the contents of the **opts_scatt** and RTTOV-SCATT **cld_profiles** structures (similar to the existing subroutines for the RTTOV options and profiles structures). These may be useful for debugging (see Annex N).

Principal Components simulations

The new PC-RTTOV coefficient files must be used with the latest associated *rtcoef* file. In fact for IASI (*rtcoef_metop_2_iasi_pcrttov_compat.H5*) this is the same as the the one released with RTTOV v12.1 except that the PC compatibility version has been updated. RTTOV performs checks at runtime to ensure the PC coefficients are compatible with the options.

The interface to HTFRTC has been rewritten for RTTOV v12.2. The HTFRTC code is now included within the RTTOV source so there is no need to compile the HTFRTC model separately. There is a new subroutine **rttov_read_coefs_htfrtc** which is used to read HTFRTC data files before calling **rttov_direct** or **rttov_k**. Users of HFRTC should read section 8.9 which gives full details of the new implementation.

Bug fixes

The bugs in RTTOV v12.1 listed on the website have been fixed:
https://www.nwpsaf.eu/site/software/rttov/rttov-v12/code-updates/

4.3. Changes from RTTOV v11

This section provides details of the differences in the user interface between RTTOV v11 and RTTOV v12.1. In particular, this section should be useful if you wish to replace v11 with v12 in your application as it lists the changes to the user-interface. Many of the changes are superficial and have been made to improve the consistency in the interface to RTTOV.

Coefficient files

The format of the optical depth ("*rtcoef*") coefficient files has been updated for v12. Executables are available to convert between v10/v11 format and v12 format files (see Annex A). The visible/IR aerosol and cloud scattering coefficient files have also changed for v12. The Mietable files for RTTOV-SCATT remain the same as those for v11.

Note that in previous versions of RTTOV the optical depth coefficient files contained the input profile "regression limits": by default RTTOV outputs warnings if the input profiles exceed these limits. The limits are calculated from the strict minimum/maximum envelope of the diverse training profile set by applying a stretching factor (+/-10% for temperature max/min respectively, and +/-20% for each gas max/min). The RTTOV v12 coefficient files contain the *strict* min/max envelopes: the regression limits that are applied within RTTOV are calculated when the coefficients are read in and the same stretching factors are applied as in v11. As such there is no change in behaviour for RTTOV v12 compared to v11 in this respect, but this is one of the differences between the v11 and v12 coefficient files.

Changes to RTTOV derived type names

Some derived types have been renamed to improve consistency:

- **profile_type -> rttov_profile** (input profile data for RTTOV)
- **radiance_type -> rttov_radiance** (output radiance data from RTTOV)
- **radiance2_type -> rttov_radiance2** (additional output radiance data from RTTOV)
- **transmission_type -> rttov_transmission** (output transmittance data from RTTOV)
- **profile_cloud_type -> rttov_profile_cloud** (input cloud profile data for RTTOV-SCATT)

Changes to the *rttov_options* structure

The **opts%rt_mw%do_lambertian** and **opts%rt_ir%do_lambertian** options have been removed. The **opts%rt_all%do_lambertian** flag applies to all sensor types and should be used instead. See below regarding the change in the behaviour of this option.

The default FASTEM version is now 6. The value in **opts%rt_mw%fastem_version** *must* be a valid FASTEM version number (1-6) or zero for the new TESSEM2 model.

The default IR sea surface emissivity model is the new physically-based model "IREMIS" which depends on wind speed and skin temperature as well as zenith angle. All visible/IR coefficient files contain coefficients for this model (SSU and PMR are the only exceptions). The older ISEM model is still available. The model is specified in **opts%rt_ir%ir_sea_emis_model**: 1=>ISEM, 2=>IREMIS.

There are various new options related to the new multiple-scattering code (see sections 8.5 and 8.6 and Annex O): the default treatment for scattering of solar radiation is the new multiple scattering solver.

Changes to the *rttov_profile* structure

The **profiles(:)%gas_units** "compatibility-mode" option (zero) has been removed. A value of zero or less now corresponds to ppmv over dry air. The values for kg/kg over moist air (one) and ppmv over moist air (two) remain the same as for v11.3. The default is now ppmv over moist air (two).

The **profiles(:)%snow_frac** variable has been moved to **profiles(:)%skin%snow_fraction**. As in v11.3 this is only used by the IR emissivity atlases. The **profiles(:)%soil_moisture** profile variable has been moved to **profiles(:)%skin%soil_moisture**, but is still unused.

There is a new option to select the units of input cloud/aerosol profiles for visible/IR simulations. This is specified in the logical flag **profiles(:)%mmr_cldaer**. By default this is true so that the units for clouds and aerosols are kg/kg. Setting the flag to false reverts to the units used in previous RTTOV versions (g/m^3 for clouds, cm^{-3} for aerosols). See sections 8.5 and 8.6.

The profiles(:)%ish variable has been renamed **profiles(:)%ice_scheme** and there are just two valid values now: 1 => use SSEC database properties for ice cloud (new in v12); 2 => use 2014 Baran ice cloud parameterisation (same as v11 for IR channels, but now also extended to visible wavelengths). See section 8.5. The old Hexagonal/Aggregate and Baran 2013 parameterisations have been removed.

Changes to *rttov_opt_param* structure (visible/IR scattering optical properties)

The index order in the explicit optical property arrays for VIS/IR scattering has been reversed for efficiency. These structures also have new members related to the new scattering options (see section 8.5).

Changes to subroutine interfaces

The **rttov_read_coefs** subroutine has a new final optional argument named **path** which can be used to specify the directory containing the coefficient files when you use the **instrument** argument to this subroutine to specify the coefficient file via the platform and sensor IDs. Previously the coefficient file had to be in the current directory.

Previously some allocation subroutines took **nlayers** as an argument and some took **nlevels**. Now all affected RTTOV allocation subroutines take **nlevels** as an argument. This changes the interface to: **rttov_alloc_rad**, **rttov_alloc_transmission** and **rttov_alloc_opt_param**. This doesn't change the size of any of the RTTOV input or output arrays. See Annex D.

The **rttov_alloc_opt_param** subroutine takes an additional **nmom** argument specifying the number of Legendre coefficients for the phase functions (see Annex D and sections 8.5 and 8.6).

The **rttov_alloc_direct**, **rttov_alloc_tl**, **rttov_alloc_ad**, and **rttov_alloc_k** subroutines take additional optional arguments to specify the number of Legendre coefficients for explicit aerosol and cloud optical property structures (see Annex D).

The **radiancedata** and **radiancedata2** arguments **to rttov_direct**, **rttov_tl**, **rttov_ad**, **rttov_k**, the parallel interfaces, **rttov_alloc_direct**, **rttov_alloc_tl**, **rttov_alloc_ad** and **rttov_alloc_k** have been renamed **radiance** and **radiance2** (see Annexes D, I, J, K and L).

The order of the arguments to the **rttov_bpr_init** and **rttov_bpr_calc** subroutines have changed so that the **err** argument is now first to be consistent with other subroutines (see Annex E). The **rttov_bpr_calc** subroutine now accepts an optional **nthreads** argument: if RTTOV is compiled with OpenMP this subroutine can exploit multiple cores/CPUs.

The **rttov_read_scattcoeffs** and **rttov_scatt_setupindex** subroutines now take the top-level coefficient structure (derived type **rttov_coefs**) as an argument like all other subroutines instead of the optical depth coefficient structure (derived type **rttov_coef**). See Annexes C and D.

The **rttov_read_scattcoeffs** subroutine now takes **opts_scatt** (RTTOV-SCATT options structure) as an argument and respects the setting of the **opts_scatt%config%verbose** option. This subroutine has a new optional argument **file_coef** which is the path to an ASCII or binary Mietable file. This allows Mietables to be read in a similar manner to other RTTOV coefficient files by specifying the filename explicitly. If this argument is omitted the Mietable reading works as in v11 (the filename is constructed based on the supplied optical depth coefficients structure). In this latter case there is now an optional **path** argument to specify an alternative directory for the Mietable file. See Annex C.

The **rttov_alloc_scatt_prof** subroutine now returns an error status (similar to other allocation subroutines) to indicate whether the allocation was successful (see Annex C).

The subroutine interfaces to the land surface emissivity and BRDF atlases have also changed: these are described below.

List of subroutines with interface changes:

rttov_read_coefs	rttov_bpr_calc	rttov_read_scattcoeffs
rttov_alloc_rad	rttov_direct	rttov_scatt_setupindex
rttov_alloc_transmission	rttov_tl	rttov_alloc_scatt_prof
rttov_alloc_opt_param	rttov_ad	rttov_setup_emis_atlas
rttov_alloc_direct	rttov_k	rttov_get_emis
rttov_alloc_tl	rttov_parallel_direct	rttov_deallocate_emis_atlas
rttov_alloc_ad	rttov_parallel_tl	rttov_setup_brdf_atlas
rttov_alloc_k	rttov_parallel_ad	rttov_get_brdf
rttov_bpr_init	rttov_parallel_k	rttov_deallocate_brdf_atlas

Changes to emissivity and BRDF atlases

The TELSEM atlas has been replaced by the new TELSEM2 atlas which includes emissivities for climatological sea-ice and may be used for all instruments supported by RTTOV (including MetopSG ICI). The atlas files are still in ASCII format and TELSEM2 can be used *without* compiling RTTOV against the HDF5 library (unlike RTTOV v11). The CNRM MW atlas has also been updated: data are now available for AMSU, MHS, SSMI/S and ATMS and the input files are now in HDF5 format. For IR sensors the UWIRemis atlas remains available and the new CAMEL IR atlas has been added.

The emissivity and BRDF atlases now work differently. You must declare an instance of derived type **rttov_emis_atlas_data** for emissivity atlases (defined in **mod_rttov_emis_atlas**) or **rttov_brdf_atlas_data** for the BRDF atlas (defined in **mod_rttov_brdf_atlas**) to hold the atlas data for a particular month/instrument. These data structures are then passed into the RTTOV atlas interface subroutines. This enables data from different atlases for different months/instruments to be initialised and used concurrently.

As a result of this update the interfaces to the atlas interface subroutines have changed significantly. The atlas "version" numbers have been replaced by atlas "ID" numbers: for the IR atlases, 1 => UWIRemis, 2 => CAMEL atlas; for the MW atlases, 1 => TELSEM2, 2 => CNRM MW atlas. See Annexes F and G and section 7.5 and the example code in **src/test/example_atlas_fwd.F90**.

Other changes to behaviour and outputs

When calling the TL model for visible and IR sensors, if the input **emissivity_tl(:)%emis_in** or **reflectance_tl(:)%refl_in** is non-zero then this input perturbation will be used instead of any calculated emissivity/reflectance perturbation that would result from RTTOV's internal emissivity/reflectance calculations. This makes the behaviour for VIS/IR sensors consistent with the FASTEM TL for MW sensors which has behaved this way in previous versions of RTTOV. *NB In RTTOV v12.2 this behaviour was changed: see above for details.*

For solar radiation over sea surfaces where **calcrefl(:)** is true RTTOV provides a calculation of the BRDF for the direct surface-reflected solar beam using a sunglint model. In RTTOV v11 this same BRDF was used when calculating the downward-scattered surface-reflected radiation contribution. However this sunglint BRDF is not appropriate for this radiance contribution: it is too large within the sunglint region and is too small (almost zero) outside the sunglint region. In RTTOV v12, for visible/near-IR channels the BRDF for scattered radiation is taken from a fixed reflectance spectrum for water (the same one used by the BRDF atlas for sea surfaces) and for channels with both thermal emission and solar contributions this BRDF is computed as (1-emissivity)/π which is consistent with the treatment of downwelling atmospheric emission in these channels. In addition, for visible/near-IR channels, the BRDF derived from the water reflectance spectrum is added to the sunglint BRDF for the direct surface-reflected beam to mitigate a dark bias observed in simulated reflectances outside the sunglint region. This changes the radiances from those in RTTOV v11 for all visible/near-IR channels where the sunglint model is used and for mixed thermal+solar channels with single-scattering for clouds/aerosols where the sunglint model is used. This does not apply to the new DOM solver for which the surface is treated as strictly Lambertian.

The **opts%rt_all%do_lambertian** option is not applicable with sea surface emissivity models and as such it is not applied for channels where the IR or MW sea surface emissivity models are being used.

When the interpolator is used, any profile values obtained by extrapolating the input profile up to the top of the coefficient profile (in most cases 0.005hPa) are always clipped to the regression limits for the purposes of the optical

depth regression. Previously the default constant-value extrapolation could result in values beyond the range of the training profiles.

The internal calculation for the local path angle along the surface-satellite path has been modified: previously the calculated path angle at the surface was not identical to the user-specified zenith angle and, for solar simulations, the surface-satellite and surface-sun path angles were not identical when the satellite and solar zenith angles were identical. These inconsistencies have been removed. In most cases the impact is of the order of hundredths of a Kelvin, but in some cases at the highest zenith angles differences of a few tenths of a Kelvin have been observed.

Some other internal changes have been made which have a small impact (a few hundredths of a Kelvin or less) on simulated radiances:

- Values of some physical constants have been updated with latest values from NIST (http://physics.nist.gov/cuu/Constants/)
- In previous versions of RTTOV the 2m water vapour variable (**profiles(:)%s2m%q**) was never used with v9 predictor coefficient files. The 2m q variable is now used for any coefficient file if the **opts%rt_all%use_q2m** option is true.
- The method of calculating geopotential height has been modified to avoid numerical problems if the input profile has very thick layers near the top of the atmosphere.

Additional capabilities of RTTOV v12 over RTTOV v11

Visible/IR scattering updates (sections 8.5 and 8.6):

- Discrete Ordinates Method (DOM) multiple-scattering solver for clouds and aerosols: there are independent options available for the treatment of scattering for thermal emission and solar radiation.
- Ice optical properties from the SSEC database replace the old Hexagonal/Aggregate ice shapes.
- Baran2014 ice parameterisation extended to visible/near-IR wavelengths.
- New option to supply cloud and aerosol inputs as mass mixing ratios.

Surface emissivity and BRDF updates (sections 7.5 and 8.4 and Annexes F and G):

- New MW sea surface emissivity model, TESSEM2, intended for use with ICI.
- New IR sea surface emissivity model including wind speed and Tskin dependency.
- Updated TELSEM MW land surface and sea-ice emissivity atlas (TELSEM2).
- Updated CNRM MW land surface emissivity atlas.
- New IR land surface emissivity atlas (CAMEL).
- More flexible interface to land surface emissivity and BRDF atlases allowing data from different atlases and for different months and instruments to be loaded concurrently.

Other science updates:

- New coefficients available allowing SO_2 as an optional trace gas (section 8.14).
- Updated NLTE correction (section 8.10).
- Updated PC-RTTOV coefficients including the NLTE correction (section 8.8).
- Capability to call the HTFRTC PC-based RT model through a PC-RTTOV-like interface (see docs/rttov-htfrtc_interface.pdf).

Technical updates:

- The Python/C++ wrapper now exploits the more flexible interface to the emissivity/BRDF atlases. The wrapper also now supports calls to RTTOV-SCATT and visible/IR scattering simulations where the optical properties are explicitly provided to RTTOV (see docs/rttov-wrapper.pdf).
- The GUI has been updated to support the new capabilities of RTTOV v12 (see docs/rttov_gui_v12.pdf).
- Improved consistency in subroutine interfaces and derived type naming (see above for the differences to v11).
- Optimisation of gas optical depth calculation in particular for v9 predictors with all trace gases.
- Doxygen markup has been added to document user-level subroutines and data types (see Section 6).

5. FORTRAN-90 UNIX/LINUX installation and testing

RTTOV v12 is designed for UNIX/Linux systems. The software is now successfully tested on Intel, IBM, Cray and Apple Mac systems and for a range of Fortran 90 compilers listed in the report on the RTTOV platforms/compilers tested.

The following system components are needed before running RTTOV v12:

- UNIX or Linux operating system
- Fortran 90 compiler
- Perl v5.6 or later
- make utilities
- gzip and gunzip
- About 100 Mbytes of free disk space is the minimum required (although more is necessary for hyperspectral IR sounder coefficient files and atlas data files).
- Memory requirements are strongly dependent on the instrument being simulated and the kind of simulations being performed: the simplest clear-sky direct model simulations require of the order of 10Mbytes.
- It is recommended to compile RTTOV against the HDF5 library (v1.8.8 or higher) so that all RTTOV features are available (see section 5.2 below).
- The Python interface and the RTTOV GUI require that f2py is installed. The GUI has additional requirements: see the GUI user guide in the `docs/` directory. The Python wrapper is compatible with Python 2 and 3, but Python 2 is recommended for the GUI (see Known Issues below).
- HTFRTC requires RTTOV to be compiled against the NetCDF v4 library.

The RTTOV v12 package is available as a compressed tar file named **rttov123.tar.gz** which should be copied to your 'top' RTTOV directory (e.g. ~user/rttov12) and extracted. This will create a number of sub-directories (see below).

RTTOV v12 will not work with older versions of some compilers. The following list gives the versions of several common compilers known to be compatible:

- gfortran – v4.4.6 and later
- ifort – v16.0.1 and later
- pgf90 – v16.10, v18.7
- NAG – v6.1, v6.2
- Cray Fortran – v8.3.4
- IBM – xlf95 v13.1 *(RTTOV v12 beta tested)*

Known compiler issues:

- The RTTOV GUI can be compiled and run with Python 3, but issues such as frozen windows have been observed when running the GUI under Python 3 and so Python 2 is strongly recommended for the GUI. (Note that the Python wrapper is fully compatible with Python 2 and 3).
- NAG v6.0: cannot read nested derived types in namelists (compiler bug) and so cannot run the test suite. The test and example programs outside of the test harness (*rttov_test.pl*) run fine which suggests the core of RTTOV is compiled successfully and can be used with this compiler. NAG v6.1 does not have this problem.
- RTTOV v12 is known to NOT work with:
  - gfortran v4.4.5 and earlier
  - pgf90 v11.7

5.1 Unpacking the code

This is achieved using the command:

```
$ tar xvf rttov123.tar.gz
```

The following subdirectories are created:

`brdf_data/`	BRDF atlas data (data must be downloaded from web site)
`build/`	Scripts used in building RTTOV and files containing flags for various compilers/architectures
`data/`	Various ancillary data files
`docs/`	Documentation
`emis_data/`	Emissivity atlas data (data must be downloaded from web site)
`gui/`	RTTOV GUI source code
`src/`	The RTTOV source code
`rtcoef_rttov12/`	RTTOV v12 coefficient files (see below)
`rttov_test/`	test scripts, input profiles for tests, and reference output for tests
`wrapper/`	example code calling RTTOV from Python and C++

As described in section 3 and Table 4 there are three predictor versions on which RTTOV coefficients are based. A comprehensive list of RTTOV v12 coefficients is available via the RTTOV v12 web page which provides details of which features are supported by each file:

http://nwpsaf.eu/site/software/rttov/download/coefficients/coefficient-download/

You should refer to this web page to determine the most appropriate coefficient file for your application.

The `rtcoef_rttov12/` directory contains sub-directories for each kind of coefficient file:

`rttov7pred54L/`	v7 predictor files on 54 levels (most IR/MW optical depth predictor coefficient files)
`rttov7pred101L/`	v7 predictor files on 101 levels (hi-res IR sounder files)
`rttov8pred51L/`	v8 predictor files on 51 levels (SSU only, allows variable O_3 and CO_2)
`rttov8pred54L/`	v8 predictor files on 54 levels (allows variable O_3 and CO_2)
`rttov9pred54L/`	v9 predictor files on 54 levels (VIS/NIR/IR sensors, solar calculations, variable O_3 and CO_2)
`rttov9pred101L/`	v9 predictor files on 101 levels (hi-res IR sounder files allowing more variable gases)
`cldaer_ir/`	cloud and aerosol scattering coefficient files for coefficient files containing only IR channels
`cldaer_visir/`	cloud and aerosol scattering coefficient files for v9 predictor coefficient files containing visible and IR channels
`mfasis_lut/`	MFASIS look-up table (LUT) files
`mietable/`	MW scattering coefficient files
`pc/`	Principal Components (PC-RTTOV) coefficient files
`htfrtc/`	HTFRTC coefficient files

In addition to the coefficient files listed in Table 4 there are:

- Zeeman coefficient files for SSMI/S (by default expected in the **rttov7pred54L/** directory even though they are based on different numbers of levels)
- PC coefficients for IASI, AIRS and IASI-NG. The PC-RTTOV coefficients must be used with the v9 predictor 101 level optical depth coefficient files with which they were trained. These are in the **rttov9pred101L/** directory and contain "**pcrttov_compat**" in the filename.
- Aerosol and cloud scattering coefficients for most visible/IR sensors; note that the channels in the scattering coefficients file must match the channels in the *rtcoef* and so there are separate scattering files for use with IR-only coefficient files and visible+IR coefficient files.
- MFASIS LUT files for use with the MFASIS fast visible scattering model.
- Scattering coefficients for most MW instruments.
- Input files for the HTFRTC model.

The RTTOV distribution includes all of the VIS/IR and MW coefficient files supported by RTTOV at the time of the release. The hi-res IR sounder coefficient files, scattering, PC-RTTOV, and HTFRTC files are not included in the distribution and nor are the emissivity and BRDF atlas datasets. These are all available from the RTTOV web site.

For the purposes of running the test suite you should ensure coefficient files are placed in the appropriate directories. An interactive script **rttov_coef_download.sh** is available in the **rtcoef_rttov12/** directory which can be used to download any or all coefficients into the standard locations in the coefficients directory. Note that you only need to download the coefficients required for the simulations you wish to carry out. For example, there is no need to download any hi-res IR sounder coefficients unless you want to run simulations for an instrument of that kind.

Compatibility with v10/v11:

- The format of the *rtcoef* optical depth coefficient files has changed for v12 and as such v10/v11-format files cannot be used directly with v12. Executables to convert coefficients between v10/v11-format and v12-format are described in Annex A.
- The IR scattering coefficient files are not backwards-compatible and cannot be converted for use with v10/v11.
- The MW scattering coefficient "mietable" files remain the same as for v10/v11.
- The format of PC-RTTOV coefficient files has not changed since RTTOV v11. You can use RTTOV v11 PC-RTTOV coefficients with RTTOV v12. It is important to use the corresponding v9 predictor 101 level optical depth coefficient files with which the PC coefficients were trained: in RTTOV v12 these have "pcrttov_compat" in their filename and, at the time of release, the v12 AIRS and IASI-NG files are identical to the corresponding RTTOV v11 coefficient files. The v12 IASI *rtcoef* file for PC-RTTOV includes the new NLTE coefficients, but aside from that is identical to the corresponding v11 file.

All coefficients are supplied in ASCII format except for hypersepctral IR sounders and MFASIS LUT files: due to the large file sizes the HDF5 format is used. It is possible to convert coefficient files between ASCII, HDF5 and Fortran unformatted ("binary") formats, the latter two being more efficient (though note that the binary format is not portable between systems). It is also possible to extract a subset of channels which reduces the file size and can improve performance, particularly for hyperspectral sounders. The **rttov_conv_coef.exe** program performs these tasks and is described in Annex A.

RTTOV-SCATT Mietable files may be converted to binary format using the **rttov_asciii2bin_scattcoef.exe** executable (see Annex A).

HTFRTC files cannot be converted to alternative formats.

5.2 Compiling the code

An interactive shell script is available to compile RTTOV v12 which asks some questions and then runs the necessary commands to do the compilation:

```
$ cd src
$ ../build/rttov_compile.sh
```

There must be a file in the **build/arch/** directory containing the compilation flags you wish to use. There are a number of example files for various common compilers or you can create a new one: more details on this are given below in the section "Creating an architecture configuration file". In order to make use of multi-threaded execution via the **rttov_parallel_*** routines RTTOV must be compiled with OpenMP. This involves supplying a suitable flag to an appropriate compiler. There are compiler flag files in **build/arch/** for compiling with OpenMP support with gfortran, pgf90, ifort and NAG.

RTTOV may be compiled immediately without requiring any external libraries. However some features of RTTOV have external dependencies:

Reading HDF5 coefficient files	: requires the HDF5 library.
Emissivity*/BRDF atlases	: require the HDF5 library.
RTTOV GUI	: requires the HDF5 library.
Python interface and RTTOV GUI	: require that f2py is installed.
HTFRTC	: requires the NetCDF v4 library.

**The TELSEM2 MW emissivity atlas may be used without any external dependencies as the atlas data files are in ASCII format, but the other atlases require the HDF5 library.*

Compiling with the HDF5 library is recommended as all functionality is then available. ***Before compiling with HDF5 you must first edit the* build/Makefile.local *file with the location of the HDF5 library.*** This involves specifying the path to the library installation in the "HDF5_PREFIX" variable and uncommenting one "FFLAGS_HDF5" definition and one "LDFLAGS_HDF5" definition appropriate to your build of the library. Similarly, if you want to compile against the NetCDF library (to use HTFRTC) or against an external LAPACK library, you must also edit the relevant sections of **build/Makefile.local** with the details of these libraries before running the **rttov_compile.sh** script.

Once the code is compiled you will find **bin/** and **lib/** directories in your top-level RTTOV directory containing the RTTOV binaries and libraries. One library is created for each subfolder within **src/** and you should link all required libraries in your application (at the very least **librttov12_main** and **librttov12_coef_io** – see section 6 for more information). Tables 7 and 8 list all libraries and executables produced by the build process.

The file **src/test/Makefile_examples** is an example stand-alone Makefile for the **example_*.F90** demonstration programs. It has a section at the top which describes the variables that should be edited with paths appropriate for your system. This is intended as a demonstration of how to link your own code against the RTTOV libraries: the **example_*.F90** executables are compiled by the RTTOV build process so it is **not** necessary to use this Makefile to compile the example code.

Notes on compiling with the HDF5 library:

1. The HDF library must be built with the Fortran interface (see the HDF documentation).
2. In **build/Makefile.local** the FFLAGS_HDF5 variable defines the _RTTOV_HDF macro. It is important to supply this macro to the compiler so that the sections of code which do HDF5 I/O are included in the compilation. For most Linux-based Fortran compilers this is achieved by passing –D_RTTOV_HDF as seen in the **FFLAGS_HDF5** variable, but for XLF on AIX it is passed using –WF,–D_RTTOV_HDF.
3. Note that if you do NOT compile with the HDF5 library, you must NOT supply the _RTTOV_HDF macro to the compiler (so the HDF5 lines in **Makefile.local** must be commented out).
4. If the code was previously compiled without HDF5 first then you should select a clean compilation if the recompilation fails.
5. Before running RTTOV ensure the HDF5 library you compiled RTTOV against is in your **$LD_LIBRARY_PATH** or equivalent.

Compiling RTTOV manually

It is recommended to use the **build/rttov_compile.sh** script described in the previous section, but manual compilation of RTTOV is possible. ***As noted above, if compiling with HDF5 you must first edit the* build/Makefile.local *file with the location of the HDF5 library***, and likewise for the NetCDF library.

RTTOV makes use of some LAPACK subroutines: the source code for these is included with RTTOV and by default you do not have to worry about this. However if you wish instead to compile against an external library containing the LAPACK subroutines you can specify this library in **build/Makefile.local** in a similar manner to the HDF5 library.

The general compilation procedure is then as follows:

```
$ cd src
$ ../build/Makefile.PL RTTOV_HDF=1 RTTOV_F2PY=1 RTTOV_USER_LAPACK=1
$ make ARCH=myarch INSTALLDIR=myinstalldir
```

The arguments in *italics* are optional.

The second step (running **Makefile.PL** to regenerate the RTTOV Makefiles) is not required if you are compiling RTTOV for the first time "out-of-the-box" and you do not require either HDF5 or Python-related code to be compiled and you do not want to compile against an external LAPACK library. However, if you are compiling RTTOV with the HDF5 library or a LAPACK library or you want to compile the RTTOV GUI or the RTTOV Python interface, you must run **Makefile.PL** with one or more of the arguments shown above:

RTTOV_HDF=1 - required if compiling RTTOV against the HDF5 library
RTTOV_F2PY=1 - required if compiling the RTTOV GUI or the RTTOV Python interface
RTTOV_USER_LAPACK=1 - required if compiling RTTOV against an external LAPACK library

The arguments to "make" are:

ARCH - this argument is optional: if omitted RTTOV is compiled with gfortran, otherwise "myarch" should correspond to the name of one of the files in the build/arch/ directory. These files contain build flags for various common compilers/platforms. You can add new ones: see below for details. Using *-openmp flags will enable multi-threaded execution via the RTTOV parallel interfaces: this is generally recommended.

INSTALLDIR - by default the build process creates output directories (e.g. **lib/** and **bin/**) in the top-level RTTOV directory. You can optionally specify "myinstalldir" to be another path relative to the top-level RTTOV directory to contain the **lib/**, **bin/**and other subdirectories (useful if compiling RTTOV using more than one set of compiler flags).

Some examples are given below, all run from within the **src/** directory:

Compile RTTOV with gfortran without any external dependencies:

```
$ make
```

Compile all RTTOV code excluding the GUI and Python interface with gfortran compiler flags with the HDF5 library:

First edit `build/Makefile.local` with the location of your HDF5 installation.

```
$ ../build/Makefile.PL RTTOV_HDF=1
$ make
```

Compile all RTTOV code with gfortran-openmp compiler flags with the HDF5 library:

First edit `build/Makefile.local` with the location of your HDF5 installation.

```
$ ../build/Makefile.PL RTTOV_HDF=1 RTTOV_F2PY=1
$ make ARCH=gfortran-openmp
```

Compilation: more details

This section is not essential reading, but contains some additional information about compilation which you may find useful.

General notes

Programs should be compiled with the C-style preprocessor options enabled to make use of the #include statements for subroutine declarations. Note for most compilers this implies you need **.F90** as the file extension which is what is provided. For users with HP compilers it may be necessary to convert the **.F90** file extensions to **.f90** for all the routines.

Specifying an installation directory

As described above, by default the build process creates new directories (**bin/**, **lib/** etc) in the top-level RTTOV directory. It is possible to specify a subdirectory where the new directories will be placed by supplying the INSTALLDIR argument to "make". This feature is useful if compiling RTTOV with different compiler flags or with different compilers. After compilation and testing the build directories can be moved to an arbitrary location (i.e. outside the RTTOV directory).

Specifying external dependencies

The file **build/Makefile.local** is used to specify the locations of external libraries, in particular the HDF5 library. As noted above, for the HDF code to be compiled requires the _RTTOV_HDF macro to be passed to the compiler. It is equally important that if the HDF code is *not* required this macro is *not* supplied to the compiler.

The file contains templates for linking against the HDF5 library, the HTFRTC model (see the separate user guide for information about this), and a LAPACK library. An example for DrHook is also included: note that in the case of DrHook, the RTTOV source code includes **yomhook.F90**, a dummy routine, which must be removed from the **src/main/** directory if you want to run with DrHook enabled.

Regenerating the Makefiles

If the RTTOV code dependencies change for some reason or, if source files are added or removed from the **src/** directory, or if you want to include (or remove) the HDF5 capability the Makefiles must be regenerated. This is easily achieved as follows:

```
$ cd src/
$ ../build/Makefile.PL RTTOV_HDF=1 RTTOV_F2PY=1 RTTOV_USER_LAPACK=1
$ make ARCH=myarch INSTALLDIR=mydir clean
```

where RTTOV_HDF=1, RTTOV_F2PY=1 and RTTOV_USER_LAPACK=1 are as described above. It is important to remember that if RTTOV_HDF=1 was supplied to **Makefile.PL**, then **Makefile.local** must supply the _RTTOV_HDF macro to the compiler, and likewise, if RTTOV_HDF=1 is *not* supplied to **Makefile.PL**, **Makefile.local** must *not* supply the macro.

It is good practice to do a "`make clean`" after running **Makefile.PL** to avoid problems when recompiling.

Creating an architecture configuration file

If the required architecture is not included in the **build/arch/** directory bundled with RTTOV or if you would like to customise the installation of RTTOV it is possible to create a new configuration file. This configuration file must be installed in the **build/arch/** directory and define the following macros:

`FC` : the name of the Fortran 95 compiler.
`FC77` : the name of the Fortran 77 compiler; this might be the Fortran 95 compiler, possibly with some special options.
`CC`: the name of the C compiler.
`LDFLAGS_ARCH` : specific flags to pass to the linker.
`FFLAGS_ARCH` : specific flags for the Fortran compiler.
`CFLAGS_ARCH`: specific flags for the C compiler.
`AR` : the command to create a library from object files.

NB The Fortran 77 and C compilers are used to compile specific source files. However, the RTTOV v12 software must be compiled with a Fortran 95 compiler.

This configuration file may also define the following macros:

`FFLAG_MOD`: this is the flag used by the Fortran 95 compiler to locate module files; it defaults to -I, but it is possible to override this setting.
`CPP`: the name of the pre-processor; defaults to cpp.
Specific flags for some RTTOV source files; defining **FFLAGS_ARCH_a** will force the build system to compile unit **a.F90** with these specific flags.

In order to use the GUI or the Python interface RTTOV must be compiled with f2py support. The following macros should be defined:
`F2PY`: this defines the f2py command and specifies the Fortran compiler being used (see the f2py documentation for relevant compiler names).
`F2PYFLAGS_ARCH`: compiler flags to pass to the F2PY compilation. This should specify the PIC (position independent code) flag in the appropriate form for the relevant Fortran compiler.
`F2PYLDFLAGS_ARCH`: linker flags for the F2PY compilation.

The existing files in **build/arch/** provide useful templates.

5.3 Running the test suite

You should first navigate to the **rttov_test/** directory which contains the relevant scripts and data for testing RTTOV. Unless otherwise specified all files and directories referred to in this section may be found in this directory.

RTTOV has a comprehensive and flexible test suite based on the **rttov_test.exe** executable which is run via the **rttov_test.pl** script. This allows most aspects of RTTOV to be configured and tested from the command-line and can compare the simulated outputs with reference data. Some shell scripts are included which run the test suite for a range of instruments, profiles and options. These are described in the "Core RTTOV testing" section below. The output from these tests can be visualised using a graphical interface written in Python (**rttov_test_plot.py**) which is described in the RTTOV test suite documentation in the **docs/** directory.

In addition there are several stand-alone shell scripts which can be used to run the **example_*.exe** demonstration programs and also the additional test and example executables for RTTOV-SCATT and the emissivity and BRDF atlases. These are also described below.

Verifying the RTTOV build

NB Due to the way the Intel Fortran compiler manages memory, users compiling with ifort on Linux may need to increase the stack size by executing the following command before all tests will run correctly:

```
$ ulimit -s unlimited
```

In addition, if running the PC-RTTOV K model with multiple threads under ifort, you may need to increase the OpenMP stack size as well to allow tests to run:

```
$ export OMP_STACKSIZE=1000M
```

A number of shell scripts are provided which run the RTTOV test suite for various instruments to test particular aspects of RTTOV. Some of the scripts require coefficient files to be downloaded from the website (for example for hi-res IR sounders, visible/IR or MW scattering coefficients or PC-RTTOV coefficients). Note that the majority of hi-res IR sounder tests defined in the test suite expect coefficient files in HDF5 format.

In order to verify your RTTOV installation you can run the following script without downloading any additional coefficient files:

```
$ ./test_rttov12.sh ARCH=myarch BIN=bindir
```

The **ARCH** parameter should match the one used when you compiled RTTOV. The **BIN** parameter is optional. It is only required if the **INSTALLDIR** parameter was supplied when compiling RTTOV i.e. if the location of **bin/** is not in the top-level RTTOV directory. If specified **BIN** must give the location of the directory containing binary executables relative to the top-level RTTOV distribution directory (e.g. if you specified **INSTALLDIR=install/gfortran** when building RTTOV then you should use **BIN=install/gfortran/bin**).

It is also possible to execute the test suite in a distributed computing environment (e.g. supercomputer) via a scheduler: you can optionally supply, for example, **SCHED_CMD=aprun**. Each executable run by **rttov_test.pl** will be called via **aprun** in this case. Additional arguments can be supplied using quotes: e.g. **SCHED_CMD="aprun --abc"**.

The script above runs the RTTOV direct, TL, AD and K models for a range of instruments and compares the results to the supplied reference data. The test suite reports whether each individual test was successful or not. There may be cases where there are differences in the least significant digits between test output and the reference output due to compiler-dependent rounding errors (especially in the Jacobian output from the K model): these will be reported as differences, but are not cause for concern. The **rttov_test/** directory contains several other shell scripts can be used to test particular types of RTTOV simulations, but note that some will require you to download the relevant coefficient files from the website:

`test_fwd.sh`	tests the forward model for a wide range of instruments
`test_rttov12.sh`	tests the full code (direct/TL/AD/K) for a range of instruments
`test_rttov12_hires.sh`	tests the full code for hi-res IR sounders
`test_solar.sh`	tests visible/near-IR solar simulations
`test_multi_instrument.sh`	tests RTTOV running for multiple instruments together
`test_pc.sh`	tests the PC-RTTOV calculations
`test_htfrtc.sh`	tests the HTFRTC calculations

You can run any of these scripts in exactly the same way as described above for **test_rttov12.sh**. Note that the tests expect HDF5 format coefficient files for hyperspectral IR sounders. The script **test_core.sh** calls all of the above scripts and this is run in the same way as the individual scripts. However it is not necessary to run all the scripts to verify your RTTOV installation: calling **test_rttov12.sh** is sufficient.

There are also two scripts which test the coefficient I/O (which take the same arguments as the other scripts):

`test_coef_io.sh`	tests the coefficient input/output code (this test has no reference data)
`test_coef_io_hdf.sh`	tests the HDF5 coefficient input/output code (this test has no reference data)

It is not generally necessary to run these unless you need to verify that the coefficients are being read correctly on your system. These require hyperspectral IR sounder coefficient files in ASCII format so you would usually have to convert the HDF5 files to ASCII format – same filename, but with extension "**.dat**" – before calling these scripts (see **rttov_conv_coef.exe** in Annex A).

A full description of the RTTOV v12 test suite may be found in **docs/rttov-test.pdf** (under the top-level RTTOV directory). A brief overview is given here, but it is not necessary to read this to use RTTOV.

The **tests.0/** directory contains data required to run the tests: for each instrument this defines profile data, the channel and profile lists, specification of surface emissivity and reflectance, a reference to the RTTOV coefficients, and so on. Note that a small number of tests defined in **tests.0/** are intended for developers only and the relevant coefficient files are not available (this mostly involves coefficient files that have historically been available, but are no longer generated). It is certainly not necessary to run every test defined in **tests.0/** and the tests which are run by the scripts listed above use coefficients which are available in the package or on the website (though some files need to be converted to ASCII format as noted above).

Test outputs for the **myarch** architecture are located in **tests.1.myarch/** – these are created when the tests are run. Test reference output created on the NWP SAF test platforms is held in directories with names ending in **.2**.

The **rttov_test.exe** binary executable created during the building of RTTOV (and located in the **bin/** directory of the build) is used to run one or more tests. It is controlled by the **rttov_test.pl** perl script. A typical test run involves a command like:

```
$ ./rttov_test.pl ARCH=myarch BIN=bindir TEST_LIST=hirs/001,avhrr/001 DIRECT=1
```

The **ARCH** and **BIN** parameters are the same as described above. The **TEST_LIST** parameter provides a list of tests defined in **tests.0/** to run. In this case, only the direct code is being tested (**DIRECT=1**). The test suite documentation provides a complete list of parameters which may be supplied to **rttov_test.pl** which allow almost all aspects of RTTOV to be tested. The full list of parameters may be listed by typing:

```
$ ./rttov_test.pl ARCH=myarch HELP=1
```

Running examples of code calling RTTOV v12

Several examples of running the RTTOV forward model and an example of calling the K model are provided which are intended to form a basis for your own applications. They are all in the **src/test/** directory:

- **example_fwd.F90** – simple example for clear-sky simulations
- **example_k.F90** – simple example calling K model for clear-sky simulations.
- **example_atlas_fwd.F90** – same as example_fwd.F90, but demonstrates use of emissivity and BRDF atlases
- **example_cld_file_fwd.F90** – visible/IR cloud scattering simulations using a cloud coefficient file.
- **example_cld_param_fwd.F90** – visible/IR cloud scattering simulations by passing the scattering parameters into RTTOV explicitly.
- **example_cld_mfasis_fwd.F90** – visible MFASIS cloud scattering simulations.
- **example_aer_file_fwd.F90** – visible/IR aerosol scattering simulations using a aerosol coefficient file.
- **example_aer_param_fwd.F90** – visible/IR aerosol scattering simulations by passing the scattering parameters into RTTOV explicitly.
- **example_rttovscatt_fwd.F90** – demonstrates calling RTTOV-SCATT
- **example_pc_fwd.F90** – demonstrates calling PC-RTTOV
- **example_htfrtc_fwd.F90** – demonstrates calling HTFRTC

Each of these programs may be run via a shell script in the **rttov_test/** directory with the name **run_example_*.sh** corresponding to the executable name. Near the top of each script is a small section where inputs may be configured such as the coefficient file and its location and the name of the input file(s) for profile data. The scripts may be run by typing (for example):

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

The **ARCH** and **BIN** arguments are described above in the "Verifying the RTTOV build" section. Test reference outputs are in **test_example.2/**. Input files for the script are in **test_example.1/**, and this is also where the test outputs are written. The outputs consist of files named **output_example_*.dat.myarch** and diff files named **diff_example_*.myarch** showing the differences between the test outputs and the reference outputs. The diff files should typically have zero size. In some cases they might show differences in the least significant digits which result from compiler-dependent factors and are not cause for concern.

RTTOV-SCATT testing and example code

The **test_rttovscatt.sh** shell script may be used to verify the RTTOV-SCATT code. You may need to edit the first few lines of this script to specify the location of the RTTOV coefficient files (by default assumed to be in **rtcoef_rttov12/rttov7pred54L/** and **rtcoef_rttov12/mietable/**). The script may then be run by typing:

```
$ ./test_rttovscatt.sh ARCH=myarch BIN=bindir
```

Test reference output is in **test_rttovscatt.2/**. Input files for the script are in the **test_rttovscatt.1/** directory, and this is also where the test output is written. The output consists of files named **output.NN.rttov_scatt.myarch** and **diff.NN.myarch** (where NN is 01, 02, etc), the latter being diff files showing differences compared to the test reference data. The script will exit cleanly if no internal errors are found. The diff files should typically have zero size if no errors occurred.

There is also an example program **mw_scatt/example_rttovscatt.F90** demonstrating how to perform direct and Jacobian calculations with RTTOV-SCATT. Once **test_rttovscatt.sh** has been run, the required links to coefficient files are set up within **test_rttovscatt.1/**. You may then call **example_rttovscatt.exe** (located in **bin/**) from this directory to run the example code. There is no reference output for this example program.

Emissivity and BRDF atlas testing

To use the atlases RTTOV must be compiled with HDF5 (see section 5.2). The only exception is the TELSEM2 atlas as the data files for this atlas are in ASCII format. There is one test script per atlas:

- `test_uwiremis_atlas.sh` UWIRemis IR emissivity atlas
- `test_camel_atlas.sh` CAMEL 2007 IR emissivity atlas
- `test_camel_clim_atlas.sh` CAMEL climatology IR emissivity atlas
- `test_telsem2_atlas.sh` TELSEM2 MW atlas and interpolator
- `test_cnrm_mw_atlas.sh` CNRM MW atlas
- `test_brdf_atlas.sh` BRDF atlas

These test scripts each run a test program for the respective atlas. The test programs initialise the atlas, return emissivity/BRDF values for a series of profiles/locations and then deallocate the atlas. The emissivities/BRDFs are written to an output file. You may need to edit the first few lines of each script to specify the location of the RTTOV coefficient files (by default assumed to be in **rtcoef_rttov12/rttov7pred54L/**), and the location of the emissivity and BRDF atlas data files (by default assumed to be in **emis_data/** and **brdf_data/**). Note that the IR emissivity atlas tests require the all of the IR atlas files to be downloaded (including the covariance files and the angular correction files).

The scripts are all run in the same way, for example:

```
$ ./test_uwiremis_atlas.sh ARCH=myarch BIN=bindir
```

For the emissivity atlases test reference output is in **test_emis_atlas.2/**. Input files for the scripts are in the **test_emis_atlas.1/** directory, and this is also where the test output is written. The output consists of files named **output_*_atlas.NN.myarch** according to the name of the atlas where NN is 01, 02, etc. The scripts also write diff files named **diff_*_atlas.NN.myarch** showing the difference between the test output and the reference output. The difference files should have zero size.

Similarly, the BRDF atlas test reference output is in **test_brdf_atlas.2/**. Input files for the scripts are in the **test_brdf_atlas.1/** directory, and this is also where the test output is written. The output consists of a file named **output_brdf_atlas.1.myarch**. The script also writes a diff file named **diff_brdf_atlas.1.myarch** showing the difference between the test output and the reference output. The difference files should have zero size.

6. RTTOV types, subroutines, libararies and executables

This section summarises the output from the build process and the derived types and subroutines intended for your own programs.

The build process produces a library for every directory within **src/** that is included in the build target provided to **make**. Table 7 lists all libraries created in the **lib/** directory by the compilation and the associated user-level subroutines contained therein. All user-level subroutine interfaces are detailed in the Annexes.

Table 6 provides a list of derived types (structures) which you may use in your application. All types are defined in the **src/main/rttov_types.F90** module. Annex O provides details of all the derived types. The two exceptions to this are **rttov_emis_atlas_data** which is stored in **src/emis_atlas/mod_rttov_emis_atlas.F90** and **rttov_brdf_atlas_data** which is stored in **src/brdf_atlas/mod_rttov_brdf_atlas.F90**. These are used to hold data for the land surface atlases.

Table 8 gives a list of all executables produced by the build process in **bin/** and their purpose. Most test executables are intended to be called via the Perl and shell scripts found in **rttov_test/**. See section 5.3 for more information on calling test programs. Aside from the test programs, the most commonly-used executables are the coefficient conversion tools (see Annex A).

When calling RTTOV from Python the file **rttov_wrapper_f2py.so** must be in your current directory or in your **$PYTHONPATH**. Similarly, when using the GUI the file **rttov_gui_f2py.so** must be in the current directory or **$PYTHONPATH**. These files are found in the **lib/** directory after compilation. See the separate user guides for the wrapper and the GUI in the **docs/** directory.

6.1. Linking your code against RTTOV

When linking against RTTOV you only need to link against the RTTOV libraries which contain subroutines that you have used (see Table 7). The order in which the libraries are linked is important: if library *A* depends on library *B* then *A* must appear before *B* in the linking step. You can link the RTTOV libraries as follows:

```
-lrttov12_wrapper -lrttov12_mw_scatt -lrttov12_brdf_atlas
-lrttov12_emis_atlas -lrttov12_other -lrttov12_parallel
-lrttov12_coef_io -lrttov12_hdf -lrttov12_main
-lhdf5hl_fortran -lhdf5_hl -lhdf5_fortran -lhdf5
```

The file **src/test/Makefile_examples** is an example Makefile for the **example_*.F90** executables which may be used as a template for compiling your own code which calls RTTOV. As noted in section 5.2 the example executables are built when RTTOV is compiled: this example Makefile is only for demonstration purposes.

6.2. Doxygen documentation

Doxygen markup has been added to the user-level subroutines and data types. If you have Doxygen installed you can generate the documentation by running the following from the top-level RTTOV directory:

```
$ doxygen docs/doxygen_config_user
```

The resulting documentation can be found in **docs/doxygen_doc_user/**. In particular HTML documentation can be found in **docs/doxygen_doc_user/html/index.html**. This may be helpful as a reference when reading sections 7 and 8: section 7 describes the technical aspects of calling RTTOV including subroutine calls while section 8 describes the various kinds of simulation RTTOV can carry out.

Type name	**Purpose**
rttov_options	RTTOV options structure to configure simulations.
rttov_options_scatt	Limited set of options for configuring RTTOV-SCATT.
rttov_coefs	Coefficients structure for optical depth, cloud, aerosol and PC coefficients.
rttov_scatt_coef	Coefficients structure for RTTOV-SCATT Mie table coefficients.
rttov_chanprof	Define channel/profile indexes to simulate.
rttov_profile	Input profile.
rttov_profile_cloud	Input cloudy profile for RTTOV-SCATT.
rttov_opt_param	Explicit aerosol/cloud optical parameter profiles.
rttov_emissivity	Input/output surface emissivity values.
rttov_reflectance	Input/output surface BRDF values and cloud-top BRDF for simple cloud scheme.
rttov_traj	Holds various internal variables for repeated RTTOV calls (optional).
rttov_radiance	Calculated radiances.
rttov_radiance2	Secondary direct model radiances (optional).
rttov_transmission	Calculated transmittances.
rttov_scatt_emis_retrieval_type	Optional RTTOV-SCATT output structure containing data enabling all-sky emissivity retrievals.
rttov_pccomp	Calculated PC scores and reconstructed radiances.
rttov_emis_atlas_data	Holds data for an emissivity atlas (from *mod_rttov_emis_atlas.F90*)
rttov_brdf_atlas_data	Holds data for a BRDF atlas (from *mod_rttov_brdf_atlas.F90*)

Table 6. User-level derived types.

Library name	**User-level subroutines**	**Description**
librttov12_main.a	rttov_direct, rttov_tl, rttov_ad, rttov_k rttov_alloc_direct, rttov_alloc_tl rttov_alloc_ad, rttov_alloc_k rttov_alloc_prof, rttov_init_prof rttov_alloc_rad, rttov_init_rad rttov_alloc_transmission rttov_init_transmission rttov_alloc_pccomp, rttov_init_pccomp rttov_alloc_traj, rttov_alloc_opt_param rttov_init_opt_param rttov_user_options_checkinput rttov_user_profile_checkinput rttov_errorhandling	Core library, always required. Interfaces to the direct, TL, AD, K models. Allocation and initialisation subroutines for RTTOV structures. User "checkinput" subroutines to verify input options and profiles. Subroutine to set the logical unit for error messages.
librttov12_hdf.a	N/A	Required if RTTOV was compiled with HDF5 capability.
librttov12_coef_io.a	rttov_read_coefs, rttov_read_coefs_htfrtc rttov_dealloc_coefs rttov_get_pc_predictindex	Coef input/output, always required. Read coefficient files, deallocate memory and get PC-RTTOV predictor channel sets.
librttov12_parallel.a	rttov_parallel_direct, rttov_parallel_tl rttov_parallel_ad, rttov_parallel_k	Parallel interfaces for multi-threaded execution using OpenMP.
librttov12_mw_scatt.a	rttov_scatt, rttov_scatt_tl, rttov_scatt_ad rttov_parallel_scatt, rttov_parallel_scatt_tl rttov_parallel_scatt_ad rttov_read_scattcoeffs rttov_dealloc_scattcoeffs rttov_alloc_scatt_prof rttov_init_scatt_prof rttov_alloc_emis_ret_terms rttov_scatt_setupindex rttov_scatt_emis_retrieval	Interface to RTTOV-SCATT for MW scattering simulations. Allocation and initialisation subroutines for RTTOV-SCATT structures. Subroutines for reading RTTOV-SCATT coefficient files and deallocating memory. Subroutine for performing all-sky emissivity retrieval calculation.
librttov12_emis_atlas.a	rttov_setup_emis_atlas, rttov_get_emis rttov_deallocate_emis_atlas	Interface to emissivity atlases.
librttov12_brdf_atlas.a	rttov_setup_brdf_atlas, rttov_get_brdf rttov_deallocate_brdf_atlas	Interface to BRDF atlas.
librttov12_other.a	rttov_bpr_calc, rttov_bpr_init rttov_bpr_dealloc rttov_legcoef_calc rttov_print_opts, rttov_print_opts_scatt rttov_print_profile rttov_print_cld_profile rttov_print_info rttov_print_radiance_quality rttov_scale_ref_gas_prof rttov_aer_clim_prof rttov_zutility	Ancillary subroutines: • calculate "bpr" parameter for IR scattering simulations with explicit optical parameters • calculate Legendre coefficients for a phase function for visible/IR scattering simulations • print out contents of options, profile and coef structures for debugging • print info for radiance quality output • generate scaled copies of the RTTOV background trace gas profiles • generate climatological aerosol profiles • obtain values of magnetic field strength for Zeeman simulations
librttov12_wrapper.a	See separate wrapper user guide for wrapper API.	Link against this library when compiling your own C++ code which uses the RTTOV wrapper.
rttov_wrapper_f2py.so		F2PY library for Python interface to RTTOV.
rttov_gui_f2py.so	N/A	F2PY library for RTTOV GUI.
The remaining libraries are not intended for linking: librttov12_test.a, librttov12_mw_scatt_coef.a, librttov12_coef_io_11.a, librttov12_gui.a		

Table 7. Libraries produced by the build process and associated user-level subroutines.

Executable name	**Purpose**
Useful executables.	
rttov_coef_info.exe	Print out information about a given *rtcoef_* coefficient file.
rttov_mfasis_lut_info.exe	Print out information about a given MFASIS LUT file.
rttov_conv_coef.exe	Convert coefficients between formats (ASCII, binary, HDF5) and extract channels to reduce file sizes.
rttov11_conv_coef_11to12.exe	Convert v10/v11 *rtcoef* coefficient files to v12 format.
rttov11_conv_coef_12to11.exe	Convert v12 *rtcoef* coefficient files to v10/v11 format.
rttov_ascii2bin_scattcoef.exe	Convert ASCII RTTOV-SCATT Mie tables to binary format.
rttov_make_scaercoef.exe	Generate visible/IR *scaercoef* aerosol optical property files.
rttov_scatt_make_coef.exe	Generate RTTOV-SCATT Mie table files.
create_aer_clim_prof.exe	Generate a file containing climatological aerosol profiles from combinations of the RTTOV pre-defined particle types.
rttov_obs_to_pc.exe	Demonstrating conversion of observations to PC-space for PC assimilation applications.
Executables for test suite and example program files: these should be run via the supplied scripts in rttov_test/.	
rttov_test.exe	Main test suite executable: should be called via rttov_test.pl.
example_fwd.exe	Example program demonstrating simple forward model call.
example_k.exe	Example program demonstrating simple K model call.
example_atlas_fwd.exe	Example program demonstrating forward model call with use of emissivity and BRDF atlases.
example_cld_file_fwd.exe	Example program demonstrating forward model call with clouds specified using pre-defined particle types.
example_cld_param_fwd.exe	Example program demonstrating forward model call with cloud parameter profiles specified explicitly.
example_cld_mfasis_fwd.exe	Example program demonstrating forward model call for MFASIS cloud simulations.
example_aer_file_fwd.exe	Example program demonstrating forward model call with aerosols specified using pre-defined particle types.
example_aer_param_fwd.exe	Example program demonstrating forward model call with aerosol parameter profiles specified explicitly.
example_rttovscatt_fwd.exe	Example program demonstrating forward model call for RTTOV-SCATT.
example_pc_fwd.exe	Example program demonstrating forward model call for PC-RTTOV.
example_htfrtc_fwd.exe	Example program demonstrating forward model call for HTFRTC.
example_rttovscatt.exe	Example program demonstrating calling RTTOV-SCATT.
rttovscatt_test.exe	RTTOV-SCATT test executable, should be run using the test_rttovscatt.sh script.
rttov_uwiremis_atlas_test.exe	Test program for UWIRemis IR emissivity atlas.
rttov_camel_atlas_test.exe	Test program for CAMEL IR emissivity atlas.
rttov_camel_clim_atlas_test.exe	Test program for CAMEL climatology IR emissivity atlas.
rttov_telsem2_atlas_test.exe	Test program for TELSEM2 MW emissivity atlas.
rttov_cnrm_mw_atlas_test.exe	Test program for CNRM MW emissivity atlas.
rttov_brdf_atlas_test.exe	Test program for BRDF atlas.
Additional executables.	
rttov_test_get_pc_predictindex.exe	Test program for rttov_get_pc_predictindex subroutine, can be used to obtain the PC-RTTOV predictor channel numbers.
rttov_gui_test_run.exe	Test program for GUI functionality (not generally required).
rttov_make_opt_param.exe	Used by test suite to create explicit optical property input files from cloud/aerosol profiles and associated coefficient files.

Table 8. Executables produced by the RTTOV build process.

7. Running RTTOV v12 for your applications

RTTOV v12 includes interfaces to allow much RTTOV functionality to be called from Python, C or C++ code. This allows you to run RTTOV without writing any Fortran code. There is a separate document in the **docs/** directory which describes the wrapper interface. However, it is still important to be aware of the contents of this section of the user guide as you will not be able to run RTTOV correctly without understanding it.

There are several example programs which can serve as a basis for your application:

- **example_fwd.F90** clear-sky with/without solar radiation
- **example_k.F90** clear-sky K model simulation
- **example_atlas_fwd.F90** as example_fwd but using the emissivity and BRDF atlases
- **example_cld_file_fwd.F90** visible/IR cloud scattering via pre-defined particle types
- **example_cld_param_fwd.F90** visible/IR cloud scattering by explicitly providing optical parameters
- **example_cld_mfasis_fwd.F90** visible MFASIS cloud scattering
- **example_aer_file_fwd.F90** visible/IR aerosol scattering via pre-defined particle types
- **example_aer_param_fwd.F90** visible/IR aerosol scattering by explicitly providing optical parameters
- **example_rttovscatt_fwd.F90** RTTOV-SCATT
- **example_pc_fwd.F90** PC-RTTOV
- **example_htfrtc_fwd.F90** HTFRTC

These may be found in the **src/test/** directory and section 5.3 describes how to run them.

The module **rttov_types** defines the derived types used by RTTOV that you will need in your program. These are listed in Table 6 and Annex O gives full details. There are also two additional derived types contained in **mod_rttov_emis_atlas** and **mod_rttov_brdf_atlas** which are required when using the emissivity and BRDF atlases: see section 6, sections 7.5 and 7.6, and **example_atlas_fwd.F90**.

The module **parkind1** defines the standard RTTOV integer, real and logical kinds (**jpim**, **jprb** and **jplm** respectively). The default RTTOV integer kind is 32-bit and the default real kind is double precision (64-bit): it is *not* recommended to change the latter to single precision (32-bit). This is particularly true if running the AD or K models as this can significantly affect the adjoint/Jacobian output.

The **rttov_types** and **parkind1** modules should always be used in your code which calls RTTOV. You will also see that the example code uses the **rttov_const** module: this contains a large number of parameters (constants) used by RTTOV, some of which may be useful in your program. The files **rttov_types.F90**, **parkind1.F90** and **rttov_const.F90** can be found in **src/main/**.

The **rttov_options** derived type (in module **rttov_types**) holds a number of flags controlling various aspects of RTTOV. It is also important to allocate the various input and output arrays for **rttov_direct** to the correct dimensions (see **example_fwd**). Figure 1 gives a process diagram of what routines to call when running RTTOV v12.

It is recommended that you look at the header section of the coefficient file for the sensor you wish to simulate as there is useful information there such as the definition of channel numbers and the polarisation (for MW sensors) assumed for each channel for that instrument etc. You can use the **rttov_coef_info.exe** executable (see Annex A) to view the headers of any coefficient file.

The following sections describe the recommended steps to be taken in coding a program which calls RTTOV v12. These involve preparing structures and arrays with the necessary input data for RTTOV, and creating appropriate structures and arrays to hold the output of the model. Users of RTTOV v11 will notice that the interfaces to some routines have changed slightly, but the fundamental sequence of calls remains the same. To provide some context, the syntax for calling **rttov_direct** is provided here with the arguments containing input data highlighted in bold. The subroutine interface is described fully in Annex I.

```
call rttov_direct(errorstatus, chanprof, opts, profiles, coefs, transmission,
radiance, radiance2, calcemis, emissivity, calcrefl, reflectance, aer_opt_param,
cld_opt_param, traj, pccomp, channels_rec)
```

7.1.Set RTTOV options

You must first declare a variable of the **rttov_options** derived type, say **opts**. This structure contains a number of flags with which you can configure various aspects of RTTOV. The full structure is described in Annex O. It is important to set options covering the widest range of required types of simulation *before* reading the coefficients and allocating RTTOV structures. For example, if cloudy IR simulations are required then set **opts%rt_ir%addclouds** to true at the start before reading the coefficients and allocating the **profiles**(:) structure. This ensures that the cloudy coefficients are read in and the cloudy profile array variables are allocated. You can subsequently set the **addclouds** option to false to run clear-sky simulations without reallocating structures or re-reading coefficients. If you wish to read MFASIS LUT files you must set **opts%rt_ir%vis_scatt_model=3** (i.e. select MFASIS as the solar scattering model) *before* reading the coefficient files in.

You can also initialise the logical unit for error/warning messages. This is performed by **rttov_errorhandling**, an optional subroutine which can be called at any time (see Annex B).

7.2.Initialise coefficient structures

The **coefs** structure (derived type **rttov_coefs**) contains data read from the optical depth (*rtcoef*) coefficient file and, optionally data from the visible/IR cloud and/or aerosol (*sccldcoef*, *scaercoef*) coefficient files, MFASIS LUTs (*rttov_mfasis_*), and the PC-RTTOV (*pccoef*) coefficient file for the instrument being simulated.

You should declare an instance of the **rttov_coefs** type and read the coefficients for the desired instrument by calling **rttov_read_coefs** (see Annex C). After reading the coefficients you may wish to call the subroutine **rttov_user_options_checkinput** (see Annex N) which checks the consistency of the input options with the coefficient file and reports any issues: this can be useful for debugging purposes.

If fast performance is required when reading the coefficient files it is better to access Fortran unformatted or HDF5 formatted coefficient files. The program **rttov_conv_coef.exe** (located in the **bin/** directory of the build) can be used to convert between the three formats. It can also produce ASCII, unformatted or HDF5 coefficient files for a subset of channels which can be useful in particular for hyperspectral sounders. The command-line arguments for this tool are described in Annex A.

Another program, **rttov11_conv_coef_11to12.exe**, is available to convert version 10/11 coefficient files to version 12-compatible files: this routine is also documented in Annex A.

The coefficient file defines the variable trace gases allowed in the input profile (see Table 4). There are two points to be aware of in specifying what gaseous absorption needs to be included in the computation. The first is that the flag in the options structure for the gas of interest must be set to .true. The second is that the coefficient file supplied must contain the coefficients for the gas of interest. The fewer gases simulated, the faster the code will run. In all cases, water vapour is a mandatory input. All other trace gases (O_3, CO_2, CO, N_2O, CH_4, SO_2) are optional. Clear-sky MW simulations may also use a cloud liquid water input profile which is treated purely as an absorber in a similar manner to the gases.

If the coefficient file contains coefficients for a particular gas (e.g. **coefs%coef%nozone** > 0) then a profile may be supplied for that gas. If you wish to provide profiles for the gas the relevant flag in **opts** (e.g. **opts%rt_ir%ozone_data**) should be set to true. If the **opts** flag is set to false the RTTOV reference profile is used instead. This is necessary because the coefficients for the fixed gases will not include any gases which have specific variable gas coefficients and so the calculation will be in error if the variable gas calculation is not included. Note none of the MW coefficients include ozone as a variable gas. The reference profiles are contained in the REFERENCE_PROFILE section of the ASCII coefficient files: this section lists a "reference" profile and a "background" profile for each gas. It is the latter which is used when an optional gas input is omitted and in fact for all gases except SO2 the reference and background profiles are identical. For HDF5 coefficients the background gas profiles are in the /COEF/BKG_PRFL_MR dataset.

The predictors also determine whether solar computations are allowed. The v9 predictors were specifically designed to work well over the wider range of zenith angles required for solar simulations. Therefore only v9 predictor files are compatible with solar simulations. For all non-hires IR sensor v9 predictor coefficient files, the only optional variable trace gases are ozone and CO2.

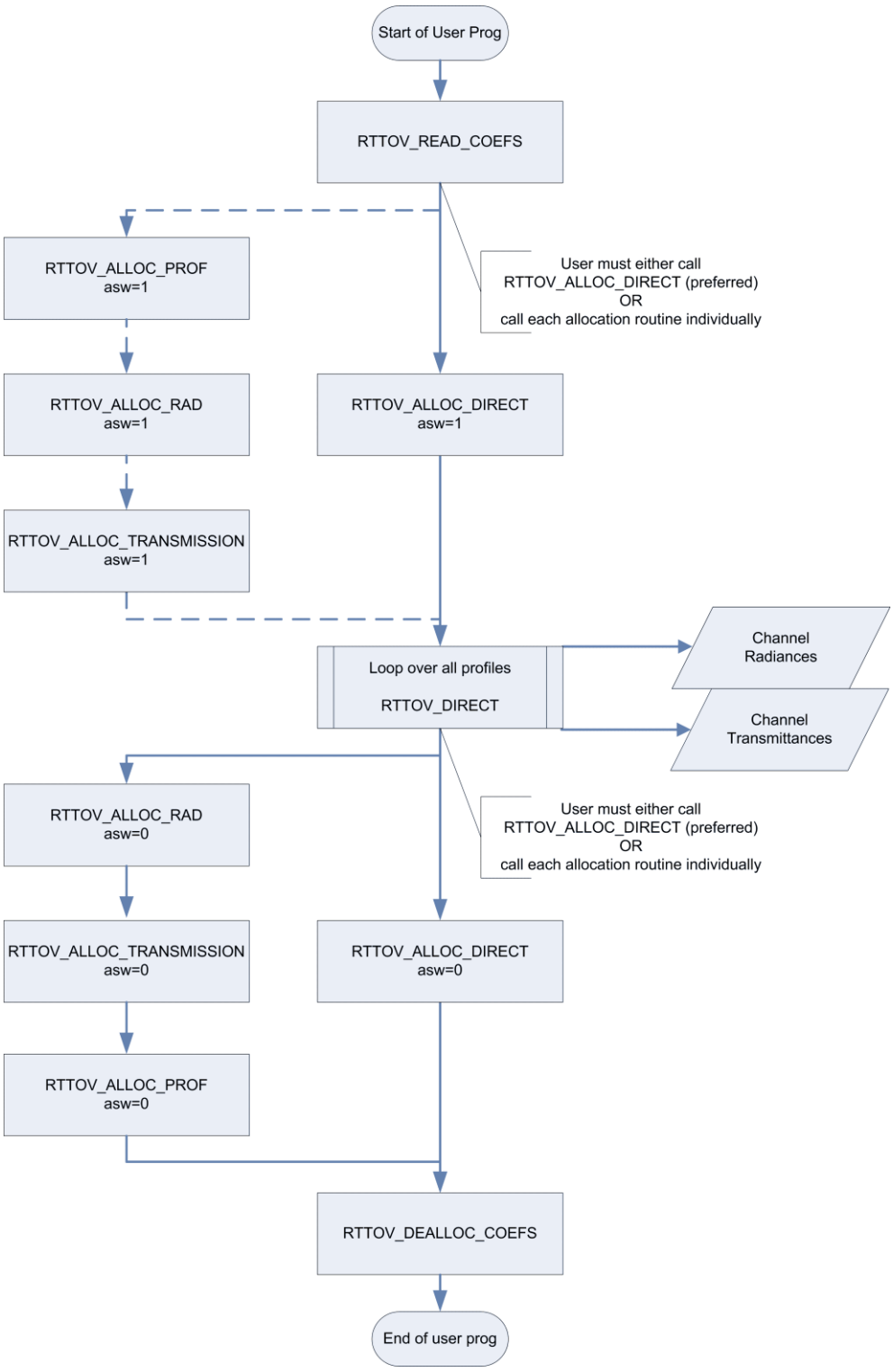

Figure 1. Process diagram of user program calling RTTOV v12 forward model.

7.3. Set up input profiles

The program **src/test/example_fwd.F90** is a useful guide on how to set up the input profile(s). Allocate a **profiles** array (derived type **rttov_profile**) with size equal to the number of profiles you wish RTTOV to process in each call. The **rttov_alloc_prof** subroutine (see Annex D) should be called to allocate the various arrays within the profile structure. This can also initialise all the profile variables to zero if requested, which is recommended. Alternatively the **rttov_alloc_direct/tl/ad/k** subroutines can be used to allocate all input arrays to the direct, TL, AD and K models (see Annex D). The members of the array **profiles** as listed in Table 12 should be populated with data: some are mandatory as indicated and some are optional depending on various factors such as the flags set in **opts** and the coefficient file being used. Table 12 also indicates which profile variables are treated as constants for the tangent linear and which are active in the TL calculation (and hence which may be non-zero in the Jacobian).

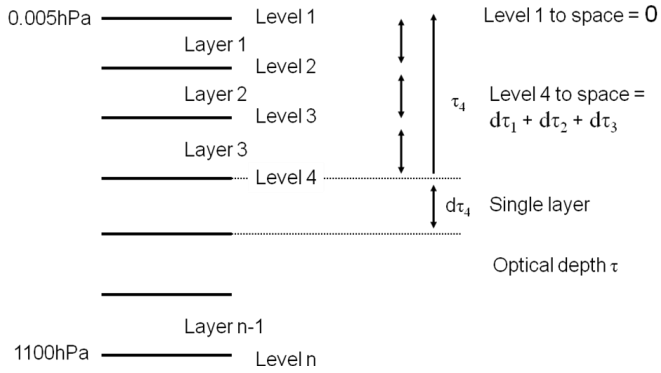

Figure 2. Internal RTTOV coefficient levels and optical depth computations.

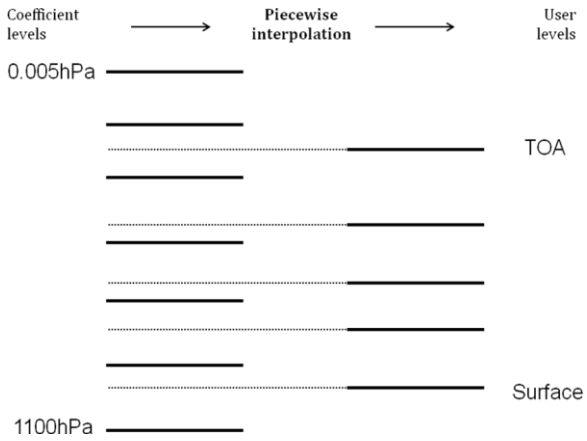

Figure 3. Interpolation to and from to user levels.

Pressure levels and interpolation

RTTOV carries out the optical depth calculation on a fixed set of levels (usually 54 or 101 levels) defined by the coefficient file. You can provide input profiles on these fixed sets of pressure levels (see Table 1 for the 54 levels) in which case no interpolation is required. However the input profiles may be supplied on arbitrary pressure levels (see for example Figures 2 and 3). In this case you must set **opts%interpolation%addinterp** to true. Note that this increases the run-time of RTTOV, although the interpolation is faster if all input profiles use the same pressure levels. RTTOV interpolates the input profile data onto the fixed coefficient levels, calculates the optical depths on the coefficient levels and then interpolates the optical depths back onto the input pressure levels. The radiative transfer integration is carried out on the input pressure levels. Input profiles can be on different pressure levels, but for each call to RTTOV the number of levels must be the same for all profiles.

The interpolation method is given in Rochon *et al.* (2007) and is also described in the RTTOV v9 Science and Validation Report (see the RTTOV website). An important feature of the Rochon interpolation scheme is that all input levels in an interpolation contribute to the interpolated profile with appropriate weight which means that Jacobians do

not exhibit excessively large or small sensitivity to particular levels purely as a result of the interpolation. However, it has been found that the Rochon interpolation scheme can result in significant oscillations in the computed temperature Jacobians in the case when the input levels are more densely spaced than the coefficient levels. RTTOV provides additional interpolation options which are intended to mitigate this effect. These options are listed in Table 9 below. The interpolation option is set in the **opts%interpolation%interp_mode** variable. The default is the standard Rochon interpolator that was originally implemented in RTTOV v9. Despite the oscillations observed in the Jacobians it is worth noting that the default interpolation option (mode 1) has not been observed to have a negative impact in applications such as 1DVar retrievals or data assimilation systems. This is a result of the vertical correlations in the background smoothing out the impact of the oscillations in the Jacobians. Therefore if you are happily using the original interpolator then there is no compelling reason to change. However, modes 4 and 5 result in smooth temperature Jacobians and may be considered as alternatives when the input profile levels are more densely spaced than the coefficient levels. In the opposite case (coefficient levels are more dense), then modes 1 or 3 are recommended. A more detailed description and comparison of the interpolation modes is given in Hocking (2014). You may wish to carry out experiments to determine the best interpolation option for your application. You can of course interpolate your input profiles onto the RTTOV coefficient levels before input to RTTOV (and therefore switch off the RTTOV interpolation) if you find this beneficial for your application.

interp mode	Profile interp (user->coef levels)	Optical depth interp (coef->user levels)	Description
1	Rochon	Rochon on optical depths	Default, original RTTOV interpolation method, Jacobians may show oscillations. Reasonable choice when user levels are sparse compared to coef levels.
2	Log-linear	Log-linear on optical depths	May be beneficial in execution time for direct-model calculations, but not suitable for TL/AD/K.
3	Rochon	Log-linear on optical depths	Similar to mode 1, but with somewhat reduced oscillations. Reasonable choice when user levels are sparse compared to coef levels.
4	Rochon	Rochon on weighting function	No oscillations, but most computationally expensive method. Reasonable choice when user levels are dense compared to coef levels.
5	Rochon	Log-linear on weighting function	No oscillations, but Jacobians may show small "artefacts" due to interpolation, only slightly more expensive than mode 1. Reasonable choice when user levels are dense compared to coef levels.

Table 9. Interpolation options available in RTTOV.

There are several points which need to be considered if the internal profile interpolation is used:

i. Ideally the input profile should cover the whole atmosphere with an adequate number of levels, at least close to the number of coefficient levels or more. A coarse layering will reduce the accuracy of the calculations. Certainly, for accurate simulations, the input profile should span the range of pressures over which the weighting functions of the channels being simulated are significantly greater than zero.

ii. If there are pressure levels in the input profile below the surface then they may contribute to the simulated radiance. In particular, the level immediately below the surface will contribute in many situations as the values of profile variables on the levels immediately above and below the surface may be used to determine the values for the partial layer above the surface. However, due to the interpolation method, it is possible for levels even below this to contribute to the simulation.

iii. If the input profile is below the top of the coefficient file the input profile is extrapolated assuming a constant value of the uppermost user value for all levels above the top. Alternatively there is an option to carry out extrapolation based on the coefficient file regression limits (see below). In both cases the extrapolated values are clipped so that extrapolated values never exceed the regression limits.

iv. When calling the tangent linear model with interpolation, **profiles_tl%p** (i.e. the pressure perturbations in the perturbation profile – see Annex K) can be non-zero (and for sigma levels should be). In this case **opts%interpolation%lgradp** must be set to true. If the interpolation is not used, input levels are assumed to be on fixed pressure levels, so **profiles_tl%p** should be zero. This applies similarly to the adjoint and Jacobian models: the output **profiles_ad/_k%p** will be zero unless interpolation is used and **lgradp** is true.

Specification of the surface pressure

The location of the surface is defined by the surface pressure in **profile%s2m%p**. There is no restriction on the value of the surface pressure (other than that it must be physically reasonable). RTTOV accounts for emission, absorption and (where relevant) scattering from the top of the atmosphere down to the surface pressure. Ideally the surface pressure should be less than or equal to the bottom pressure level in the input profile. If the surface pressure lies between two input pressure levels, RTTOV calculates the contribution from the partial layer bounded by the surface pressure and the first pressure level above this. RTTOV also allows the surface pressure to lie below the bottom pressure level: in this case the calculated optical depth of the bottom layer is extrapolated down to the surface pressure.

The RTTOV profile structure also includes an elevation variable (**profile%elevation**): this is used in the calculations which determine the local path angles of the radiation in each atmospheric layer. It is the surface pressure that determines the surface location within the vertical profile rather than the elevation.

Trace gases

Table 4 in section 3 lists the gases available with each type of coefficient file. If you are supplying profiles for an optional trace gas you must set the corresponding options flag to true: e.g. if you supply ozone data then you must set **opts%rt_ir%ozone_data** to true. If the flag is true you must supply a valid input profile. If a coefficient file with trace gas coefficients is used and suitable trace gas profile concentrations are not available then you can set the corresponding data flag to false. In this case RTTOV uses the reference trace gas profile for the calculation. This applies to all gases except water vapour which is always mandatory.

RTTOV v12.3 has a new subroutine **rttov_scale_ref_gas_prof** which provides a method of supplying scaled copies of the RTTOV background profiles for the optional gases. This could be useful, for example, if you do not have a CO2 profile, but wish to specify a lower concentration corresponding to a specific time in the past: this new subroutine allows you to specify the maximum CO2 ppmv value and the RTTOV background profile will be scaled to have this maximum value. In order to use this subroutine, you must specify the relevant gas flags (e.g. **ozone_data** or **co2_data**) that you wish to include in the simulation. You should then populate the **profiles(:)** structure with all of the data (**p(:)**, **t(:)**, **q(:)**, etc) except for the trace gas(es) for which you intend to use the scaled reference profiles. Just before the call to RTTOV, you call **rttov_scale_ref_gas_prof** passing the coefficients structure, the **profiles(:)** array, and specifying the relevant arguments for scaling the gas(es) of interest. The subroutine then populates those gas arrays in the **profiles(:)** structures. Annex N provides mores information about the subroutine.

It is possible to specify the units of the gas abundances in the input profile structure. This is set in the **profiles(:)%gas_units** variable: all profiles passed into RTTOV in a single call must use the same gas units. The units apply to all trace gas input profiles and the 2m (surface) gas variables. Table 10 lists the available options for gas units. Note that the units of gas Jacobians are consistent with the setting of gas_units so that if gas_units = 1, for example, then water vapour Jacobians are in units of [Kelvin / (kg/kg)].

profiles(:)%gas_units	Description
2	ppmv over moist air (default)
1	kg/kg over moist air
0 (or less)	ppmv over dry air: this is intended primarily for coefficient generation and testing, but is a valid input option.

Table 10: Options for gas units

Cloud liquid water (CLW) for MW simulations

For MW simulations it is possible to supply a cloud liquid water (CLW) profile in the **profiles(:)%clw(:)** array which is treated as a purely absorbing medium. It is important to note that this is not the same as CLW supplied to RTTOV-SCATT for scattering simulations (see section 8.7). If you wish to include CLW in the "clear-sky" simulations you must set **opts%rt_mw%clw_data** to true. By default RTTOV ignores any CLW content above 322hPa. This limit is specified in the **opts%rt_mw%clw_cloud_top** option. The Jacobian model generates non-zero Jacobians even for all levels with zero CLW content, but only those levels with pressures greater than the limit specified in **clw_cloud_top**. Historically in RTTOV the CLW optical depth calculation has been carried out on the coefficient levels. However this is not required and arguably it is better to avoid interpolating cloud profiles. In RTTOV v12.2 a new option was introduced, **opts%rt_mw%clw_calc_on_coef_lev,** which gives the option to do the calculation on coefficient levels

(true, the default) or on the input levels (false, recommended). RTTOV v12.2 also introduced a choice of liquid water permittivity parameterisations: these are selected using the new **opts%rt_mw%clw_scheme** option:

- 1 => Liebe (1989), same as previous versions of RTTOV (default)
- 2 => Rosenkranz (2015), supports frequencies up to 1000GHz, can increase run-time by ~50% (recommended)
- 3 => Turner, Kneifel, Cadeddu or TKC (2016)

These same options are now available for generating RTTOV-SCATT Mietable files. Rosenkranz (2015) is the recommended option, and is the one used to generate the latest RTTOV-SCATT Mietable files.

Limits applied to input profile data

All profile variables must lie within the "hard" limits specified in section 1.9 of **rttov_const.F90** (see Table 11). This means, for example, that all input gas profiles (water vapour, ozone, etc) must have values greater than 1E-11 ppmv at every level. This applies to *all* levels in the input profile, even those which lie below the specified surface pressure (in **profile%s2m%p**), as the optical depth calculations are carried out for every layer. Input profile variables which lie outside these hard limits will cause RTTOV to abort the simulation and report a non-zero error status.

The clear-sky gas optical depth calculations (excluding CLW for MW sensors) are based on regressions derived from line-by-line calculations run for a training set of diverse profiles. The minimum/maximum envelopes of the training profile data are contained in the coefficient files. RTTOV calculates "regression limits" by applying stretches to the envelope values (+/-10% for temperature, +/-20% for each gas). By default RTTOV compares the input profiles to these regression limits and will print warnings when the regression limits are exceeded, but the simulations are performed regardless. A new feature in RTTOV v12.2 was the output **radiance%quality** array which will indicate if the regression limits are exceeded for any quantity for a given channel (the flags will not be set if the **do_checkinput** option is false – see below). See section 7.8 for more information about the **quality** output.

RTTOV provides some options for checking input profile data:

- The printed warnings about profiles exceeding the regression limits can be suppressed by setting the **opts%config%verbose** option to false (default: true).
- Input profiles can be clipped to the regression limits when the limits are exceeded by setting **opts%config%apply_reg_limits** to true (default: false). This also prevents printed warnings being output.
- You can turn off all the internal checking of profiles against both the hard limits described above and the regression limits by setting **opts%config%do_checkinput** to false (default: true). This is not recommended unless you are using the **rttov_user_profile_checkinput** subroutine described below.

Note that when **opts%config%apply_reg_limits** is set to true the profile values are only clipped to the regression limits for the purposes of the gas absorption optical depth calculation. The source function for the radiative transfer equation is always based on the unadjusted input profiles. Also note that when warnings are printed out for gases, the values are in ppmv over dry air regardless of the setting of the **gas_units** variable. You may wish to perform your own sensitivity study to decide what setting of the **apply_reg_limits** option is best for your application.

PC-RTTOV uses its own set of regression limits, but they operate in the same way as for standard RTTOV simulations (see section 8.8).

The **rttov_user_profile_checkinput** subroutine (Annex N) can be used to test profiles against both the hard limts and the regression limits before calling RTTOV. This allows out-of-bounds profiles to be rejected without running full simulations if desired. In this case you can set **opts%config%do_checkinput** to false to turn off the internal RTTOV profile checking.

When using the RTTOV interpolator it may be the case that your input profile top is lower than the top of the RTTOV coefficient levels (usually 0.005hPa). In this case RTTOV must extrapolate the input profile values to the top coefficient level. By default constant-value extrapolation is used: if any extrapolated values lie outside the regression limits they are clipped to the limits to avoid generating warning messages. RTTOV provides an alternative option for profile extrapolation: by setting **opts%interpolation%reg_limit_extrap** to true, the profile interpolation step identifies the relative position of each interpolated profile variable at the top of the input profile with respect to the coefficient regression limits at that level. Each profile variable is then extrapolated upwards to top coefficient level by maintaining the relative position between the regression limits at every level. This is performed independently for each profile

variable. This option may be useful for obtaining physically reasonable profiles at the top of the atmosphere when the input data does not reach the top-most coefficient levels. If the interpolated value at the top input level lies beyond the regression limits, then every extrapolated value will be set to the corresponding limit value, similar to the **apply_reg_limits** functionality. The **apply_reg_limits** and **reg_limit_extrap** options may be switched on or off independently.

Variable	Minimum	Maximum
Temperature (inc. 2m T)	90 K	400 K
Water vapour (inc. 2m q)	1E-11 ppmv	60000 ppmv
O_3	1E-11 ppmv	1000 ppmv
CO_2	1E-11 ppmv	1000 ppmv
CO	1E-11 ppmv	10 ppmv
N_2O	1E-11 ppmv	10 ppmv
CH_4	1E-11 ppmv	50 ppmv
SO_2	1E-11 ppmv	1000 ppmv
Cloud liquid water (clear-sky MW only)	0 kg/kg	1 kg/kg
2m pressure	400 hPa	1100 hPa
10m wind speed	0 m/s	100 m/s
Cloud top pressure (simple cloud; only applies if cfraction > 0)	50 hPa	1100 hPa
Magnetic field Be (Zeeman only)	0.2 Gauss	0.7 Gauss

Table 11: Hard limits for input profiles variables as defined in rttov_const.F90. Input values must not lie outside the interval [minimum, maximum].

Input profile arrays	Description	Units	Mandatory?	When used	Variable for TL?
profiles(i) % nlevels	Number of pressure levels		Y		N
profiles(i) % nlayers	Number of atmospheric layers (i.e. nlevels – 1)		Y		N
profiles(i) % gas_units	Units of gas abundances *(must be the same for all profiles)*	0,1,2	Y		N
profiles(i) % p(:)	Pressure levels	hPa	Y		Y if *opts % interpolation % lgradp* true
profiles(1) % t(:)	Temperatures on levels	K	Y		Y
profiles(i) % q(:)	Water vapour conc on levels	ppmv or kg/kg	Y		Y
profiles(i) % o3(:)	Ozone conc on levels	ppmv or kg/kg	N	If ozone_data flag .true.	Y
profiles(i) % co2(:)	CO_2 conc on levels	ppmv or kg/kg	N	If co2_data flag .true.	Y
profiles(i) % n2o(:)	N_2O conc on levels	ppmv or kg/kg	N	If n2o_data flag .true.	Y
profiles(i) % co(:)	CO conc on levels	ppmv or kg/kg	N	If co_data flag .true.	Y
profiles(i) % ch4(:)	CH_4 conc on levels	ppmv or kg/kg	N	If ch4_data flag .true.	Y
profiles(i) % so2(:)	SO_2 conc on levels	ppmv or kg/kg	N	If so2_data flag .true.	Y
profiles(i) % clw(:)	Microwave cloud liquid water treated as absorbing medium; not used with RTTOV-SCATT.	kg/kg	N	MW clear-sky only, if clw_data flag .true.	Y
profiles(i) % mmr_cldaer	Logical flag to set cloud and aerosol units: true => kg/kg (cld+aer); false => $g.m^{-3}$ (cld), cm^{-3} (aer) *(must be the same for all profiles)*	T/F	N	VIS/IR cloud or aerosol with coef files	N

profiles(i) % aerosols(:,:) indices are (aerosol_type, layers)	Aerosol components/species conc on layers	kg/kg or cm^{-3}	N	VIS/IR aerosol scattering with aerosol coef file	Y
profiles(i) % cloud(:,:) indices are (cloud_type, layers)	Cloud water/ice content on layers	kg/kg or $g.m^{-3}$	N	VIS/IR cloud scattering with cloud coef file unless user inputs optical params	Y
profiles(i) % cfrac(:,:)	Cloud fractional cover on layers	0-1	N	VIS/IR cloud scattering	Y
profiles(i) % clw_scheme	CLW particle scheme: OPAC (1) or "Deff" scheme (2)	1-2	N	VIS/IR cloud scattering with coef file	N
profiles(i) % ice_scheme	Ice particle scheme: SSEC database (1) or Baran scheme (2)	1-2	N	VIS/IR cloud scattering with coef file	N
profiles(i) % idg	Ice water content effective diameter scheme	1-4	N	VIS/IR cloud scattering with coef file and ice_scheme=1	N
profiles(i) % clwde(:)	Cloud liquid water particle effective diameter	microns	N	Required with VIS/IR cloud scattering with coef file and clw_scheme=2	Y
profiles(i) % icede(:)	Ice particle effective diameter	microns	N	Optional with VIS/IR cloud scattering with coef file and ice_scheme=1	Y
profiles(i) % s2m % p	Surface pressure	hPa	Y		Y
profiles(i) % s2m % t	2m temperature	K	Y		Y
profiles(i) % s2m % q	2m water vapour	ppmv or kg/kg	N	If use_q2m flag is .true.	Y
profiles(i) % s2m % o	2m ozone	ppmv or kg/kg	N	Currently never used.	N
profiles(i) % s2m % u, profiles(i) % s2m % v	10m wind u, v components	m/s	N	Sea surface emissivity and BRDF models (except ISEM)	Y
profiles(i) % s2m % wfetc	Wind fetch (length of water over which the wind has blown)	m	N	Sea surface BRDF models	Y
profiles(i) % skin % surftype	Surface type (land = 0, sea = 1, seaice = 2)	0-2	Y	Emissivity and BRDF models and atlases	N
profiles(i) % skin % watertype	Water type (fresh = 0, ocean = 1)	0-1	N	Surface BRDF models and BRDF atlas	N
profiles(i) % skin % t	Surface skin temperature	K	Y		Y
profiles(i) % skin % salinity	Ocean salinity	Practical salinity unit	N	FASTEM 4-6, TESSEM2	Y
profiles(i) % skin % foam_fraction	Ocean foam fraction	0-1	N	FASTEM if supply_foam_fraction flag .true.	Y
profiles(i) % skin % snow_fraction	Surface snow cover fraction	0-1	N	IR emis atlases	N
profiles(i) % skin % soil_moisture	Surface soil moisture	m^3/m^3	N	Not used	N
profiles(i) % skin % fastem(1:5)	FASTEM land/sea-ice parameters (see Table 21)		N	FASTEM for land/sea-ice surface types	Y
profiles(i) % skin % specularity	Specularity (weighting for specular and Lambertian downwelling radiances).	0-1	N	If do_lambertian flag is .true.	N

profiles(i) % cfraction	Cloud fraction for simple cloud	0-1	N	Simple cloud VIS/IR only	Y
profiles(i) % ctp	Cloud top pressure for simple cloud	hPa	N	Simple cloud VIS/IR only (if cfraction > 0)	Y
profiles(i) % zenangle	Satellite zenith angle	deg	Y		N
profiles(i) % azangle	Satellite azimuth angle (0-360; measured clockwise, east=+90)	deg	N	Using FASTEM or solar option	N
profiles(i) % sunzenangle	Solar zenith angle	deg	N	Solar option, NLTE option	N
profiles(i) % sunazangle	Solar azimuth angle (0-360; measured clockwise, east=+90)	deg	N	Solar option	N
profiles(i) % latitude	Latitude (-90 to +90)	deg	Y	For radiation path geometry and emis/BRDF atlases	N
profiles(i) % longitude	Longitude (0-360)	deg	N	Emis/BRDF atlases	N
profiles(i) % elevation	Elevation	km	Y	For radiation path geometry	N
profiles(i) % Be	Earth magnetic field strength	Gauss	N	Zeeman	N
profiles(i) % cosbk	Cosine of the angle between the Earth magnetic field and wave propagation direction		N	Zeeman	N
profiles(i) % date(1:3)	Date of the profile as year (e.g. 2013), month (1-12), and day (1-31)		N	Used with solar calculations.	N
profiles(i) % time(1:3)	Time of profile as hour, minute, second.		N	Not used	N

Table 12. Profile input parameters for user profile i.

7.4.Specifying the channels to simulate

As described above, the **profiles(:)** array contains a list of the profiles for which to calculate radiances. RTTOV offers the flexibility to calculate radiances for a different set of channels for each profile in **profiles(:)**, though often in practice radiances for the same set of channels will be calculated for every profile. You should allocate a **chanprof(:)** array (derived type **rttov_chanprof**): this defines which channels are simulated for each profile. The size is the total number of channels to simulate over all profiles in each call to RTTOV. Each element in the array has two members: **chanprof(j)%prof** (the profile index), and **chanprof(j)%chan** (the channel index), where **j** runs from 1 up to the total number of radiances to calculate per call to RTTOV. Table 13 illustrates how **chanprof(:)** should be set up for three different sensors and for 2 profiles per RTTOV call: all channels for the first profile are specified, followed by all channels for the second profile, and so on. Note that for Principal Components calculations the channels you *must* simulate are determined by **opts%rt_ir%pc%ipcbnd** and **opts%rt_ir%pc%ipcreg** as described in section 8.8 and in this case you must simulate the same channels for every profile.

The RTTOV channel numbering *always begins at 1* for any coefficient file regardless of the original instrument channel numbering. This means that with IR-only and visible/IR coefficient files for the same instrument, RTTOV may use different indices to refer to the same channel (see for example SEVIRI in Table 3). If a subset of *n* possibly non-consecutive channels is read from the coefficient file (by supplying the **channels(:)** argument to **rttov_read_coefs**), then this subset of channels is identified in **chanprof(:)%chan** by the numbers 1 to *n* rather than by their original channel index in the coefficient file. For example, if coefficients for only channels 3 and 5 of a sensor with five channels are read from the coefficient file, then **chanprof(1:2)%chan** should typically be *(/1, 2/)* corresponding to the channels 3 and 5 respectively. Similarly, if the **rttov_conv_coef.exe** executable is used to create a coefficient file containing a subset of *n* possibly non-consecutive instrument channels, then these will be identified by the indices 1 to *n* when reading this new coefficient file using **rttov_read_coefs**.

If Principal Component computations are being carried out (see Section 8.8) you may request radiances to be reconstructed from the PC scores by setting **opts%rt_ir%pc%addradrec** to true. In this case the array **channels_rec(:)** must be supplied containing the indices of the channels for which reconstructed radiances are required. The reconstructed radiance channel numbers are again always counted from 1. If you passed the **channels_rec(:)** argument to **rttov_read_coefs** to specify some subset of *n* possibly non-consecutive channels for which radiances may be reconstructed, then this subset of channels is identified in the **channels_rec(:)** argument to **rttov_direct** (and

TL/AD/K) by the numbers 1 to *n* just as for **chanprof(:)%chan**. If the PC calculations are not being used or reconstructed radiances are not required the **channels_rec(:)** argument should be omitted.

Input structure	HIRS (2 profiles/call)	SSM/I (2 profiles/call)	AMSU-B (2 profiles/call)
size(chanprof)	38	14	10
size(profiles)	2	2	2
chanprof(:) % chan	1,2,3 ..,19,1,2,3..,19	1,2,3,4,5,6,7,1,2,3,4,5,6,7	1,2,3,4,5,1,2,3,4,5
chanprof(:) % prof	1,1,1,…,1,2,2,2…,2	1,1,1,1,1,1,1,2,2,2,2,2,2,2	1,1,1,1,1,2,2,2,2,2

Table 13. Examples of chanprof(:) inputs for RTTOV.

7.5. Specifying surface emissivity

This section describes how to specify surface emissivity when calling RTTOV. Section 8.4 provides more information about the emissivity models and the atlases. RTTOV allows you to provide values for the surface emissivity or alternatively RTTOV can provide values for surface emissivity. The emissivity naturally depends on the surface type (land, sea or sea-ice) which is specified in the **profiles(:)%skin%surfacetype** variable (see Annex O).

RTTOV takes as input a logical array **calcemis(:)** and an array **emissivity(:)** of type **rttov_emissivity**. Both arrays must be allocated with the same number of elements as **chanprof(:)**. For each radiance **j** to be calculated, if **calcemis(j)** is true RTTOV will supply a surface emissivity internally. If **calcemis(j)** is false then RTTOV will use the value in **emissivity(j)%emis_in** for the surface emissivity: this allows you to supply your own emissivity values to RTTOV.

Microwave emissivities

For MW instruments RTTOV has two internal emissivity models: FASTEM and TESSEM2.

FASTEM provides emissivities for both sea surfaces and for land/sea-ice surfaces. There are 6 versions of the sea surface model FASTEM available which can be chosen by setting **opts%rt_mw%fastem_version** to a value between 1 and 6. FASTEM-6 is the default and is the recommended version for channels below 200GHz. For channels above 200GHz TESSEM2 is recommended – see below. Although FASTEM-6 is recommended you may wish to carry out experiments to see if an earlier version gives better results for your application as some users have found, for example, FASTEM-3 to yield better results. The older FASTEM versions are included for users who are happily using them and do not wish to change.

The profile variables used by FASTEM for sea surface emissivities are the 10m wind speed, skin temperature, salinity, zenith angle and azimuth angle. By default FASTEM calculates a value for the ocean surface foam fraction: you can optionally supply an explicit value for the foam fraction in the profile structure by setting **opts%rt_mw%supply_foam_fraction** to true. For land and sea-ice surfaces FASTEM provides emissivities according to the parameters specified in **profiles(i)%skin%fastem(1:5)**. Section 8.4 provides more information about the different versions of the FASTEM sea surface model and the land/sea-ice emissivity parameters.

TESSEM2 is a sea-surface emissivity model (Prigent *et al*, 2016) which is recommended specifically for use with instrument channels above 200GHz (in particular the Metop-SG ICI instrument) although it can be used for any instrument. The profile variables used by TESSEM2 are the 10m wind speed, skin temperature, salinity and zenith angle (note there is no azimuthal dependence). TESSEM2 is selected for sea-surface emissivities by setting **opts%rt_mw%fastem_version** to zero. Note that for land and sea-ice surface types the FASTEM land/sea-ice model is still used where **calcemis(:)** is true.

Infrared emissivities

For IR instruments RTTOV has two internal sea surface emissivity models: ISEM and IREMIS.

The ISEM model was in previous RTTOV versions: emissivity is parameterised only in terms of zenith angle. RTTOV v12 introduces a new more physically-based model, IREMIS (described in the RTTOV v12 Science and Validation Report), which uses zenith angle, 10m wind speed and skin temperature as inputs. The IR emissivity model is selected via **opts%rt_ir%ir_sea_emis_model**: 1 selects ISEM and 2 (the default) selects IREMIS. PC-RTTOV (see section

8.8) currently uses its own emissivity model which uses zenith angle and 10m wind speed as inputs: RTTOV automatically selects this emissivity model for PC-RTTOV when **calcemis(:)** is true over sea surfaces.

Over land and sea-ice emissivities are assigned default values of 0.98 (land) and 0.99 (sea-ice) where **calcemis(:)** is true.

Tangent linear, adjoint and Jacobian emissivities

For the tangent linear model the emissivity perturbation is calculated within RTTOV when using FASTEM, TESSEM2 or IREMIS. Any input emissivity perturbation in **emissivity_tl(:)%emis_in** is ignored. ISEM depends only on zenith angle and as such is inactive in the TL, AD and K models. When using ISEM or when supplying emissivities to RTTOV you can specify input perturbations in **emissivity_tl(:)%emis_in.** The emissivity perturbations used in the TL calculations are output in **emissivity_tl(:)%emis_out**. Table 14 summarises the options for the input of surface emissivity including the tangent linear emissivity output.

In the adjoint and Jacobian models the **emissivity_ad/k(:)emis_in** output data are zero when using FASTEM, TESSEM2 or IREMIS. If you use ISEM or you supply emissivities to RTTOV then **emissivity_ad/k(:)%emis_in** contain the emissivity adjoint/Jacobian on exit.

Land surfaces

RTTOV provides IR and MW land surface emissivity atlases. These are called externally to RTTOV and the emissivities are passed to RTTOV in **emissivity(j)%emis_in** with **calcemis(:)** set to false. Each atlas provides monthly climatological emissivity values. There are three IR and two MW emissivity atlases each of which has slightly different inputs, outputs and configuration options.

UWIRemis, CAMEL 2007, and CAMEL climatology IR emissivity atlases

In the IR there are three atlases: the University of Wisconsin UWIRemis atlas (first included in RTTOV v10) described in Borbas *et al.* (2010), the CAMEL 2007 atlas (first included in RTTOV v12.1) described in the RTTOV v12 Science and Validation Report. and the CAMEL climatology atlas (new in v12.3) described in Borbas and Feltz (2019). These atlases work in very similar ways, but the CAMEL 2007 and CAMEL climatology atlases both have a resolution of 0.05° in lat/lon while the UWIRemis atlas resolution is 0.1°. The atlases can optionally include a correction for zenith angle effects (see Borbas 2014): this is specified when the atlases are initialised. By default the atlases, once loaded, can be used with any IR instrument. However if you are only using the atlas with a single instrument you can supply the corresponding coefficients structure to the atlas initialisation call. This allows some pre-calculations to be done that result in emissivities being returned much more rapidly from the atlas: it is recommended to use this option where possible, but the resulting atlas data can only be used with the instrument for which they were initialised. The PC-RTTOV land/sea coefficients were trained using the UWIRemis atlas so it is recommended (though not strictly mandatory) to use this atlas to provide land and sea-ice emissivities for PC-RTTOV.

Surface type:

- sea: returns negative values
- sea-ice: a fixed ice emissivity spectrum is used
- land: returns emissivities (where it has data) linearly combined with a snow emissivity spectrum according to the **skin%snow_fraction** profile variable (see section 8.4 for more information about the treatment of snow).

Inputs:

- mandatory: **latitude, longitude, skin%surftype**
- **skin%snow_fraction** (this modifies emissivities if the value is >0, but see section 8.4 for more information regarding the CAMEL climatology atlas)
- **zenangle**, **sunzenangle** (only required if angular correction was requested; **sunzenangle** only needs to be <85º (day) or >85º (night), the specific value does not matter for the atlas)

Outputs:

- emissivities (negative values where atlas has no data)
- optional: emissivity standard deviation (requires a flag at setup, requires more memory)
- optional: quality flag

TELSEM2 MW atlas and interpolator

This is an updated version of the TELSEM atlas and interpolator included in previous versions of RTTOV. TELSEM2 (Wang *et al*, 2016) is valid for all MW satellite instruments that RTTOV can simulate including Metop-SG ICI. When loaded the atlas data may be used with any MW instrument. TELSEM2 includes emissivities for climatological sea-ice and therefore it ignores the specified surftype and simply returns a value if it has data. The native resolution of the emissivities is 0.25° lat/lon, but optionally a different resolution may be specified to return emissivity values integrated over a larger area. For frequencies below 19GHz the atlas returns the emissivity value for 19GHz. For frequencies above 85GHz the atlas returns the emissivity at 85GHz except for certain classes of sea-ice (see the reference for more information about this atlas).

Surface type:

- sea/sea-ice/land: returns emissivity values if it has data, otherwise negative values.

Inputs:

- mandatory: **latitude, longitude, zenangle**

Outputs:

- emissivities (negative values where atlas has no data)
- optional: emissivity standard deviation for each channel
- optional: emissivity covariance matrix for all channels specified (one matrix per profile)

CNRM MW atlas

The CNRM MW atlas is described in Karbou *et al.* (2006) and Karbou *et al.* (2010). Unlike the other atlases the CNRM atlas is only compatible with a subset of satellite instruments. These are: AMSU-A, AMSU-B/MHS, SSMI/S and ATMS. In addition the CNRM atlas datasets are generated from data gathered over a specific year. The year is specified when setting up the dataset. Currently data are available for 2015 (the default if unspecified when setting up the atlas) and 2014.

Surface type:

- sea/sea-ice: returns negative values
- land: returns emissivity values if it has data, otherwise negative values.

Inputs:

- mandatory: **latitude, longitude, zenangle, skin%surftype**

Outputs:

- emissivities (negative values where atlas has no data)

The input variables used by each atlas are summarised in Table 15.

A common interface is provided to both IR and MW atlases. Three subroutines are used to access the atlases:

- **rttov_setup_emis_atlas** : this allocates the necessary arrays and reads the data for a given atlas for a given month. In some cases the atlas data are specific to a particular sensor.
- **rttov_get_emis** : this returns surface emissivity values at a given latitude/longitude for the specified channels of the given instrument. This is called when populating the **emissivity(:)%emis_in** array for input to RTTOV.
- **rttov_deallocate_emis_atlas** : deallocates memory used by the atlas and is called once the atlas is no longer required.

The interface to the atlases has changed in RTTOV v12. To initialise data for an emissivity atlas you must first declare a variable of type **rttov_emis_atlas_data** which is defined in the module **mod_rttov_emis_atlas**. This variable is passed into each of the above subroutines and it contains the atlas data for a specific month and (where relevant) for a specific instrument. This enables you to allocate data for multiple months and/or instruments simultaneously which makes the atlases more flexible.

The interfaces of these subroutines are described in Annex F. Note that if an atlas does not contain emissivity data for the given surface type or lat/lon location it will return negative values for the corresponding emissivities. You should check the returned emissivities and handle negative values accordingly, for example by setting the corresponding

elements of **calcemis(:)** to true. An example of using the atlas subroutines with a forward model call can be found in the program **src/test/example_atlas_fwd.F90**.

Note that if calculations are *only* being done for visible/near-IR channels (i.e. channels with no significant thermally emitted component, with wavelengths less than 3µm) the **calcemis** and **emissivity** arguments may be omitted in the call to RTTOV. When simulating both solar and thermally emissive channels it is recommended to set **emissivity(:)%emis_in** to zero for any channels with wavelengths less than 3µm.

	calcemis	Input ε	Forward Output ε	Tangent Linear Output ∂ε
Infrared channels	True	0	Land=0.98/sea-ice=0.99 Sea = ε_{ISEM} or ε_{IREMIS} or $\varepsilon_{PC\text{-}RTOV}$	Land/sea-ice/ISEM: no ∂ε calculated. IREMIS: ∂ε computed from ∂u, ∂v, ∂sst about ε_{IREMIS}. PC-RTTOV: ∂ε computed from ∂u, ∂v about $\varepsilon_{PC\text{-}RTTOV}$.
	False	$\varepsilon_{atlas/user}$	$\varepsilon_{atlas/user}$	No ∂ε calculated
Microwave channels	True	0	Land/sea-ice: computed from coefs in `prof % skin % fastem(1:5)` Sea = ε_{FASTEM} or $\varepsilon_{TESSEM2}$	Land/sea-ice: ∂ε about ε_{FASTEM} Sea: ∂ε, computed from ∂u, ∂v, ∂sst, ∂salinity (and optionally ∂foam_fraction) about ε_{FASTEM} or $\varepsilon_{TESSEM2}$
	False	$\varepsilon_{atlas/user}$	$\varepsilon_{atlas/user}$	No ∂ε calculated

Table 14. Input and output values of ε and ∂ε arrays for infrared and microwave channels for forward and tangent linear surface emissivity routines.

7.6. Specifying surface reflectance for solar simulations

Solar simulations are described in section 8.2. The input and output of surface reflectances is very similar to that for emissivities and again depends on the surface type (land, sea or sea-ice) which is specified in the **profiles(:)%skin%surfacetype** variable (see Annex O). RTTOV takes bi-directional reflectance function (BRDF) values as input. These are defined as the ratio of out-going radiance (towards the satellite) to incoming solar irradiance (treating the sun as a point source).

Analogously to the emissivity inputs, RTTOV takes a logical array argument **calcrefl(:)** and an array argument **reflectance(:)** of type **rttov_reflectance**, both of which must be allocated with the same number of elements as **chanprof(:)**. Where **calcrefl(:)** is true RTTOV will provide surface BRDF values as follows.

Over sea surfaces RTTOV calculates the reflectance due to sun-glint as described in Matricardi (2003). RTTOV v12.2 introduced a new option **opts%rt_ir%solar_sea_brdf_model** between 1 the JONSWAP wave spectrum (Hasselman et al, 1973), as implemented in previous RTTOV versions and 2 the Elfouhaily et al 1998 wave spectrum, new in v12.2. JONSWAP remains the default, but the Elfouhaily wave spectrum is recommended as it significantly reduces bias compared to observations. The profile variables used are the 10m wind speed, wind fetch and watertype along with the satellite and solar zenith and azimuth angles. A BRDF derived from the USGS water reflectance spectra (Clark *et al*, 2007) is added to this for visible/near-IR channels. These spectra are returned by the BRDF atlas (see below) for sea surface profiles. The resulting BRDF is appropriate for the direct surface-reflected solar beam, but for downward-scattered radiation (molecular Rayleigh scattering, aerosol, clouds) this sun-glint model is not appropriate. For visible/near-IR channels the BRDF for downward scattered radiation is taken from the USGS reflectance spectra, and for channels with both emissive and solar radiation the BRDF is calculated as (1-*emissivity*)/π which is consistent with the treatment of downwelling atmospheric emission. This does not apply to the new DOM solver for which the surface is treated as strictly Lambertian (see section 8.5).

For land/sea-ice surfaces the BRDF is calculated as (1-*emissivity*)/π for channels with emissive and solar contributions and for visible/near-IR channels the surface reflectance is set to 0.3/π or 0.8/π for land and sea-ice surface types respectively (note these latter values are necessarily crude and it is recommended to supply land surface BRDF values

instead, for example from the BRDF atlas – see below). The same BRDF value is used for the direct solar reflectance and for downward-scattered radiation.

As for emissivity you can provide your own BRDF values in **reflectance(:)%refl_in** in which case the corresponding elements of **calcrefl(:)** should be set to false. In this case the same input BRDF value is used for the direct solar reflectance and for downward-scattered radiation. Currently it is not possible to input separate BRDF values for the direct and downward scattered radiation. For sea surface BRDFs calculated by RTTOV, the output reflectance is the direct BRDF calculated by the sun-glint model. You cannot access the diffuse BRDF for downward scattered radiation, but the same fixed reflectance spectra are returned by the BRDF atlas for sea surface profiles so you can obtain the diffuse BRDF values from the atlas if required (see below).

For channels where you supply BRDF values (i.e. **calcrefl(:)** is false), you can supply input reflectance perturbations in **reflectance_tl(:)%refl_in**. If you are using the solar sea BRDF model the BRDF perturbation is computed from the relevant input **profiles_tl(:)** variables and any input reflectance perturbations in **reflectance_tl(:)%refl_in** are ignored. The reflectance perturbations used in the TL calculations are output in **reflectance_tl(:)%refl_out**.

In the adjoint and Jacobian models the **reflectance_ad/k(:)refl_in** output data are zero when using the solar sea BRDF model. If you supply BRDFs to RTTOV then **emissivity_ad/k(:)%emis_in** contain the BRDF adjoint/Jacobian on exit.

RTTOV provides a BRDF atlas (Vidot and Borbas, 2013) which is similar in many ways to the IR emissivity atlases. It provides monthly climatological land surface BRDF values based on the MODIS BRDF kernel product including climatological snow. The atlas is called outside of RTTOV and the BRDFs are input to RTTOV in **reflectance(:)%refl_in** in with the corresponding elements of **calcrefl(:)** set to false.

Surface type:
- sea: returns Lambertian BRDFs from USGS ocean or fresh water spectrum (no sunglint)
- sea-ice: returns negative values
- land: returns BRDF values if it has data (inc. climatological snow), otherwise negative values

Inputs:
- mandatory: **latitude, longitude, skin%surftype, zenangle, sunzenangle, azangle, sunazangle**
- **skin%watertype** (used for sea surface types)

Outputs:
- emissivities (negative values where atlas has no data)
- optional: bi-hemispherical (black-sky) albedo
- optional: quality flag

The interface to the BRDF atlas is similar to the emissivity atlases:
- **rttov_setup_brdf_atlas** : this allocates the necessary arrays and reads the data for a given atlas for a given month. The atlas data may be loaded for a specific sensor or they can be initialised for use with any sensor with visible channels.
- **rttov_get_brdf** : this returns surface BRDF values at a given latitude/longitude in the required channels. This should be called when populating the **reflectance(:)%refl_in** array for input to RTTOV.
- **rttov_deallocate_brdf_atlas** : deallocates atlas arrays and is called once the atlas is no longer required.

As for the emissivity atlases the interface for the BRDF has also been modified since RTTOV v11. To initialise data for the BRDF atlas you must first declare a variable of type **rttov_brdf_atlas_data** which is defined in the module **mod_rttov_brdf_atlas**. This variable is passed into each of the above subroutines and to hold the atlas data for a specific month and, optionally, a specific instrument.

The interfaces of these subroutines are described in Annex G. Note that if an atlas does not contain BRDF data for the given surface type or lat/lon location it will return negative values for the corresponding BRDFs. You should check the returned BRDFs and handle negative values accordingly, for example by setting the corresponding elements of **calcrefl(:)** to true. An example of using the atlas subroutines with a forward model call can be found in the program **src/test/example_atlas_fwd.F90**.

Note that if solar calculations are not required then the **calcrefl** and **reflectance** arguments may be omitted in the call to RTTOV.

TELSEM2 MW atlas	latitude, longitude, zenangle
CNRM MW atlas	latitude, longitude, zenangle, skin%surftype
UWIRemis and CAMEL IR emissivity atlases	latitude, longitude, skin%surftype, skin%snow_fraction zenangle, sunzenangle – *sat/sun zenith angles only required with optional angular correction; sunzenangle only needs to be <85º (day) or >85º (night)*
BRDF atlas	latitude, longitude, skin%surftype, skin%watertype zenangle, sunzenangle, azangle, sunazangle

Table 15. Profile variables required by the emissivity and BRDF atlases.

7.7.Allocation of trajectory structures.

If multiple calls to RTTOV are being made it may be more efficient to allocate some of the internal data structures before calling RTTOV and then to deallocate these structures once all calls to RTTOV have been made. This can be achieved using the optional **traj** argument to **rttov_direct**. This can be allocated using the **rttov_alloc_traj** subroutine or using the **rttov_alloc_direct/tl/ad/k** subroutines described in Annex D. The trajectory structure consists of a number of structures used internally by RTTOV. You should make a second call to **rttov_alloc_traj** to deallocate the **traj** structure once it is no longer required. It is important to note that the **traj** structure is sized according to the number of channels, profiles and levels, and according to certain input options such as the active trace gases, so if any of these values change for any calls to RTTOV a new **traj** structure must be allocated. It is also required that the **coefs** variable (of type **rttov_coefs**) be declared as a TARGET when passing the traj arguments into RTTOV.

Originally this facility was introduced for super-computers, especially for the NEC SX8, because some Fortran compilers generate complicated assembly code for memory allocation which can take a lot of execution time. Whether this capability offers performance benefit is dependent on the compiler and architecture. If you are interested in minimising run-time it is recommended to carry out a test on your system to see if this offers a benefit: it has been observed to be beneficial on some Intel/Linux-based systems. Use of the **traj** structure should never be detrimental to performance because RTTOV calls the **rttov_alloc_traj** subroutine internally if the **traj** argument is not present. Note that the **traj** structure may *not* be used in conjunction with the parallel interface to RTTOV.

7.8.Output arrays from RTTOV v12

The syntax for the call to **rttov_direct** is given again, this time with the output arguments in bold:

```
call rttov_direct(errorstatus, chanprof, opts, profiles, coefs, transmission,
radiance, radiance2, calcemis, emissivity, calcrefl, reflectance, aer_opt_param,
cld_opt_param, traj, pccomp, channels_rec)
```

errorstatus is an integer error return code. If the value of **errorstatus** is non-zero, a fatal error occurred during processing (more generally success is indicated by a return code **errorstatus_success** and failure by **errorstatus_fatal**, both of which are contained in the **rttov_const** module).

Instances **transmission** and **radiance** of the derived types **rttov_transmission** and **rttov_radiance** should be declared. Optionally an instance **radiance2** of type **rttov_radiance2** may be declared. The **radiance** structure contains the primary radiance, BT and reflectance outputs (the output reflectance quantity is defined in section 8.2) and is mandatory, while the **radiance2** structure holds secondary radiance outputs which are only calculated by the direct model (not the TL/AD/K) for non-scattering, non-solar, non-PC-RTTOV calculations. The transmission and radiance structures may be initialised using the **rttov_alloc_direct/tl/ad/k** subroutines, or the individual allocation subroutines **rttov_alloc_transmission** and **rttov_alloc_rad** (see Annex D). Annex O defines fully these output radiance and transmittance structures. Tables 17 and 18 list the output arrays and highlight those most commonly used. The table also indicates array dimensions for **rttov_direct** and gradient routines (note that **nchanprof** is the size of the **chanprof(:)** array, **nlevel** is the number of vertical levels in the profile, and **nlayer** = **nlevel - 1**). A number of the radiance outputs are defined on layers such as **rad%overcast**. These represent radiances from/to to the level bounding the bottom of each layer. All values relate to the pressure levels defined by the user *except* for the layer containing the surface where the value is the radiance calculated from/to the 2m surface pressure rather than the profile level immediately below. (If the surface pressure lies on a profile level then there is no anomaly).

Note that the **rad%overcast** values for channels with significant thermally emitted and solar reflected contributions contain *no* reflected solar contribution in accordance with the assumption that the overcast layer is perfectly emissive. The **rad%overcast** output is not calculated for visible/IR aerosol or PC simulations. Also note that in the case of visible/IR aerosol simulations, the "clear-sky" outputs (such as **rad%clear** and **rad%bt_clear**) contain the aerosol-affected radiances.

New in RTTOV v12.2 was the **radiance%quality** output array. This is a bit mask which is used to flag potential issues with radiances. If there are no issues the quality value corresponding to a given output radiance is zero. However one or more bits may be set indicating specific warnings. The bit positions are contained in the **rttov_const** module and are listed in Table 16 below. In Fortran you can test for a specific bit using the Fortran BTEST function. For example to check whether the regression limits were exceeded for channel **i** you can use:

```
IF (BTEST(radiance%quality(i), qflag_reg_limit)) THEN…
```

There is a new subroutine **rttov_print_radiance_quality** (see Annex N) which prints out a human-readable interpretation of any warning flags set in a given **radiance%quality** array element which may be useful for debugging.

The **transmission** structure contains separate transmittances for the thermally emitted and solar calculations. This is because the solar calculations are derived from different optical depth predictor calculations. For mixed thermal+solar channels the thermal and solar transmittances for the surface-satellite path will therefore not be exactly equal, and in the case of channels using Planck-weighted coefficients may be significantly different.

The **emissivity(:)%emis_out** and **reflectance(:)%refl_out** arrays contain the surface emissivities and surface BRDFs used by RTTOV.

Finally, if Principal Components calculations are being performed (section 8.8), you should allocate an instance of the **pccomp** structure (derived type **rttov_pccomp**) using the **rttov_alloc_pccomp** subroutine (Annex D). This structure will contain the computed PC scores, and (if **opts%rt_ir%pc%addradrec** is true) the reconstructed radiances. This structure is also defined in Annex O. The **pccomp** argument is not required if **opts%rt_ir%pc%addpc** is false.

Bit position	Description
qflag_reg_limits	Gas regression limits exceeded in optical depth computation.
qflag_pc_aer_reg_limits	PC-RTTOV aerosol regression limits exceeded.
qflag_mfasis_zenangle	MFASIS maximum zenith angle exceeded (by satellite or solar zenith angle).
qflag_mfasis_sumzenangle	MFASIS maximum sum of satellite and solar zenith angles exceeded.
qflag_mfasis_hiscaangle	MFASIS maximum scattering angle exceeded.
qflag_mfasis_loscaangle	MFASIS minimum scattering angle exceeded.
qflag_mfasis_geometry_bounds	MFASIS scattering angle outside LUT limits.
qflag_mfasis_opdpedia_bounds	MFASIS optical depth or effective diameter value outside LUT limits.

Table 16. Bit positions (contained in the rttov_const module) associated with RTTOV warnings. See section 7.3 for information on gas regression limits, section 8.8 for information on PC-RTTOV aerosol regression limits, and section 8.5 for information about the MFASIS flags.

rttov_radiance Radiances in units of *mW/cm-1/sr/sq.m*		
Type	**Array name**	**Contents**
real	clear(nchanprof)	Clear sky top of atmosphere radiance output for each channel
real	total(nchanprof)	Clear+cloudy top of atmosphere radiance for given cloud top pressure and fraction for each channel (simple cloud scheme) or full cloudy radiance (if opts%rt_ir%addclouds or opts%rt_ir%addaerosl is true)
real	cloudy(nchanprof)	Cloudy top of atmosphere radiance for 100% fraction for each channel at given cloud top pressure (for simple cloud scheme) or same as total (if opts%rt_ir%addclouds or opts%rt_ir%addaerosl is true).
real	overcast(nlayer,nchanprof)	Level to space overcast radiance at the level bounding the bottom of each layer. For IR channels (wavelengths above 3µm) this assumes an opaque black cloud. For solar-only channels at wavelengths less than 3µm this consists of reflected solar radiation according to assumptions described in section 8.3. This is not calculated for PC or visible/IR aerosol simulations or scattering simulations using the DOM solver..

rttov_radiance Brightness Temperatures *deg K*		
real	bt(nchanprof)	BT equivalent to total (clear+cloudy) top of atmosphere radiance output for each channel (channels with thermally emitted contribution only)
real	bt_clear(nchanprof)	BT equivalent to clear top of atmosphere radiance output for each channel (channels with thermally emitted contribution only)

Table 17. Main RTTOV output arrays. The green rows are those most commonly used. (Continued below)

rttov_radiance Bi-directional reflectance factors (BRF) *unitless*		
real	refl(nchanprof)	BRF equivalent to total (clear+cloudy) top of atmosphere radiance output for each channel (solar-affected channels only).
real	refl_clear(nchanprof)	BRF equivalent to clear top of atmosphere radiance output for each channel (solar-affected channels only).

rttov_transmission Transmittances 0-1		
real	tau_total(nchanprof)	Transmittance from surface to top of atmosphere (TOA). Only populated for channels with a significant thermally emitted contribution.
real	tau_levels(nlevel, nchanprof)	Transmittance from each user pressure level to TOA. Only populated for channels with a significant thermally emitted contribution.
real	tausun_total_path2(nchanprof)	Transmittance for combined sun-surface-satellite path. Only populated for solar-affected channels.
real	tausun_levels_path2(nlevel, nchanprof)	Transmittance from TOA to each user pressure level to TOA along combined sun-surface-satellite path. Only populated for solar-affected channels.
real	tausun_total_path1(nchanprof)	Transmittance from surface to TOA. Only populated for solar-affected channels.
real	tausun_levels_path1(nlevel, nchanprof)	Transmittance from each user pressure level to TOA. Only populated for solar-affected channels.

Output Emissivity 0-1		
real	emissivity%emis_out(nchanprof)	Emissivity vales used in RTTOV calculation (same as input emissivity values if **calcemis** is false).
Output surface BRDF		
real	reflectance%refl_out(nchanprof)	Surface BRDF values used in RTTOV calculation (same as input BRDF values if **calcrefl** is false).

Table 17 (continued). Main RTTOV output arrays. The green rows are those most commonly used. See Annex O for more details about the output transmittances.

rttov_radiance2 Radiances in units of *mW/cm-1/sr/sq.m*		
real	upclear(nchanprof)	Clear sky upwelling radiance at top of atmosphere including surface emission term, but omitting downwelling reflected radiance term.
real	dnclear(nchanprof)	Clear sky downwelling radiance at surface.
real	refldnclear(nchanprof)	Reflected clear sky downwelling radiance contribution to top of atmosphere radiance.
real	up(nlayer,nchanprof)	Summed upwelling atmospheric emission term at top of atmosphere for layers down to the level bounding the bottom of each layer.
real	down(nlayer,nchanprof)	Summed downwelling atmospheric emission term at bottom of layer for layers down to the level bounding the bottom of each layer.
real	surf(nlayer,nchanprof)	Radiance emitted by a black cloud at the level bounding the bottom of each layer; for the surface layer this is evaluated for the surface skin temperature.

Table 18. Secondary RTTOV output radiance arrays.

7.9. Calling the tangent linear (TL), adjoint (AD) and Jacobian (K) models

The tangent linear (TL), adjoint (AD) and Jacobian (K) models are typically used in assimilation and retrieval applications. They provide information about the gradient of the forward model given a particular input profile. The TL, AD and K models share some features:

- All three models take the same input atmospheric profile and surface parameters as the direct model
- All three models output the direct model radiances, BTs and reflectances
- As for the direct model all three models may be run using multiple threads via OpenMP (see section 7.10)
- Not all profile variables are "active" in the TL, AD and K models: Table 12 indicates which variables can be perturbed in the TL and hence which may be non-zero in the Jacobian and AD output.

RTTOV TL model

The RTTOV TL model (see Annex K for the interface) takes as input a **profiles_tl(:)** profile structure containing a set of perturbations to the input profile. The TL model calculates the linearisation of the direct model evaluated for the given input profile and it outputs the change in satellite seen radiances that result from the given profile perturbation for this linearisation.

The **profiles_tl(:)** array should be allocated to be the same size as the input **profiles(:)** array. As an example the subroutine **src/test/rttov_make_profile_inc.F90** is used to generate profile perturbations for the RTTOV test suite. For RTTOV-SCATT TL (section 8.7, Annex M) the **cld_profiles_tl(:)** array should be allocated to be the same size as the input **profiles(:)** and **cld_profiles(:)** arrays and it should be initialised with the cloud profile perturbations.

In RTTOV v12.2 the visible/IR aerosol/cloud explicit optical properties (method 2 described in sections 8.5 and 8.6) were enabled as active variables in the TL model. You can optionally supply perturbations to the optical properties via the **aer/cld_opt_param_tl** structure. Note that in this case the **aer/cld_opt_param_tl** argument to the tangent linear model is optional: if omitted the optical properties are treated as static variables as in earlier versions of RTTOV.

For channels where you supply emissivity values to RTTOV, you can specify an emissivity perturbation in **emissivity_tl(:)%emis_in**. This also applies when using ISEM because this is a static model depending only on zenith angle. For channels where you are using FASTEM, TESSEM2 or IREMIS, RTTOV calculates the emissivity perturbation from the relevant input **profiles_tl(:)** variables and any input emissivity perturbation is ignored. The emissivity perturbation used in the TL calculation is written to **emissivity_tl(:)%emis_out**.

Similarly, for solar calculations you can supply the BRDF perturbation in **reflectance_tl(:)%refl_in** for channels for which you input BRDF values. When using the sea surface BRDF model the reflectance perturbation is computed internally and any input perturbation is ignored. The BRDF perturbation used in the TL calculation is written to **reflectance_tl(:)%refl_out**.

The output radiance, brightness temperature and reflectance perturbations are contained in the **radiance_tl** structure. For PC-RTTOV (see section 8.8) the output perturbations in PC scores and optionally reconstructed radiances are in the **pccomp_tl** structure.

RTTOV AD and K models

The file **src/test/example_k.F90** provides en example of calling the RTTOV K model for a clear-sky simulation.

The RTTOV AD model (see Annex L for the interface) takes as input the gradient of some scalar function (e.g. a cost function) with respect to the satellite seen radiance (or BT) in **radiance_ad** and it outputs the gradient of the same scalar function with respect to the profile variables in **profiles_ad(:)**. In this case the **profiles_ad(:)** array must be allocated to be the same size as the input **profiles(:)** array.

The RTTOV K model (see Annex J for the interface) takes as input a radiance (or BT) perturbation in **radiance_k** and it outputs the gradient of each forward model radiance with respect to each input profile variable evaluated for the given input profile in **profiles_k(:)**. The gradients are scaled by the perturbations in **radiance_k**: typically the input perturbation is set to a value of 1 in radiance or BT units for each channel. For the K model, **profiles_k(:)** must be

allocated to be the same size as the **chanprof(:)** array: each element of the array contains the gradient of the forward model for the corresponding channel.

Aside from the size of the **profiles_ad/k(:)** arrays the AD and K models are very similar. The input perturbations in **radiance_ad/k** may be supplied either in radiance or BT: if the **opts%rt_all%switchrad** flag is false the input perturbations must be in **radiance_ad/k%total(:)** (in units of radiance), otherwise the input perturbations must be specified in **radiance_ad/k%bt(:)** (in Kelvin). For solar-only channels (those with wavelength < 3µm) the input perturbation is **always** in radiance regardless of the setting of **switchrad**. Note that there is never any problem in providing perturbations in both the **total(:)** and the **bt(:)** arrays: RTTOV uses the correct perturbation for each channel according to the setting of **switchrad**.

In general all AD/K input and output structures (aside from the input perturbations) should be initialised to zero before calling the AD/K models. It is important to remember to reinitialise these structures before each call when making multiple calls to the AD/K subroutines. The following structures and arrays should always be set to zero:

- **profiles_ad/k** structure (use **rttov_init_prof** subroutine)
- **transmission_ad/k** structure (use **rttov_init_transmission** subroutine)
- **radiance_ad/k** structure (use **rttov_init_rad** subroutine **before** specifying the perturbations in **total(:)** and/or **bt(:)**; for PC-RTTOV **all** members of the structure should be set to zero – see below)
- **emissivity_ad/k** structure (set both **emis_in(:)** and **emis_out(:)** member arrays to zero)
- **reflectance_ad/k** structure (set both **refl_in(:)** and **refl_out(:)** member arrays to zero)
- , see Annex D)

In RTTOV v12.2 the visible/IR aerosol/cloud explicit optical properties (method 2 described in sections 8.5 and 8.6) were enabled as active variables in the adjoint and Jacobian models. When using these arguments you should ensure the **aer/cld_opt_param_ad/k** structures are zeroed using **rttov_init_opt_param** subroutine with the **zero_only** argument. On output these structures contain the adjoint/Jacobian of the optical properties. Note that in this case the **aer/cld_opt_param_ad/k** arguments to the adjoint and Jacobian models are optional: if omitted the optical properties are treated as static variables as in earlier versions of RTTOV.

See Annex D for all **rttov_init_*** subroutine interfaces.

For channels where you supply an input emissivity or for which you use ISEM, the **emissivity_ad/k(:)%emis_in** array contains the surface emissivity AD/Jacobians on exit. For channels where you use FASTEM, TESSEM2 or IREMIS this array is zero because the emissivity is a function of input profile variables.

Similarly, for channels where you supply an input BRDF, the **reflectance_ad/k(:)%refl_in** array contains the surface BRDF AD/Jacobians on exit. For channels where you use the solar sea BRDF model this array is zero.

For RTTOV-SCATT (section 8.7, Annex M) the **rttov_scatt_ad** subroutine doubles as the adjoint and Jacobian model. If the size of the **profiles_ad(:)** array is the same size as the input **profiles(:)** array then it calculates the adjoint. Otherwise **profiles_ad(:)** must be the same size as the **chanprof(:)** array in which case **rttov_scatt_ad** calculates the Jacobian. In either case the **cld_profiles_ad(:)** array must be the same size as **profiles_ad(:)** and on exit it contains the adjoint or Jacobian of the cloud profile variables. The **cld_profiles_ad(:)** array should always be initialised to zero before calling **rttov_scatt_ad**.

When calling **rttov_scatt_ad** the behaviour is as if the **switchrad** flag is always true and the input perturbations are always in terms of brightness temperature. This is always true, even when internally computations are being done as radiances. Hence for RTTOV-SCATT the input perturbations should be set in **radiance_ad%bt(:)**.

For PC-RTTOV (see section 8.8 for details), the input perturbation must be specified in the **pccomp_ad/k** structure: the member to which the perturbation applies depends on the setting of **opts%rt_ir%pc%addradrec** and **opts%rt_all%switchrad**. If reconstructed radiances are not required (**opts%rt_ir%pc%addradrec** is false), then the input perturbation should be specified in **pccomp_ad/k%pcscores**. If reconstructed radiances are required, the input perturbation should be specified in **pccomp_ad/k%total_pccomp** if **opts%rt_all%switchrad** is false, or **pccomp_ad/k%bt_pccomp** if **opts%rt_all%switchrad** is true. Before calling the PC-RTTOV AD/K models, **all** elements of the **radiance_ad/k** structure should be initialised to zero.

The PC-RTTOV AD and K models differ slightly in the output: for the AD model the output adjoint is in the **profiles_ad(:)** array as for standard RTTOV.

For the K model, **profiles_k(:)** contains the Jacobians for the PC predictor channel set. These are **always** in terms of radiances (not brightness temperatures), regardless of the setting of **opts%rt_all%switchrad**.

If **opts%rt_ir%pc%addradrec** is false the PC Jacobians are output in the **profiles_k_pc(:)** array. This must have size equal to the number of PC scores multiplied by the number of profiles. These are gradients of PC scores with respect to the profile variables and in this case the setting of **opts%rt_all%switchrad** has no impact on the units of the Jacobians.

Alternatively if **opts%rt_ir%pc%addradrec** is true the PC Jacobians are output in the **profiles_k_rec(:)** array which must have size equal to the number of reconstructed radiance channels multiplied by the number of profiles. In this case the setting of **opts%rt_all%switchrad** determines the units of the output (radiances or brightness temperatures).

Note that only one of **profiles_k_pc(:)** or **profiles_k_rec(:)** should be supplied to **rttov_k** depending on the setting of **opts%rt_ir%pc%addradrec**.

7.10. Multi-threaded execution

If RTTOV is compiled with OpenMP (see section 5) then substantial gains in execution time can be made by calling RTTOV through the parallel interface. The interfaces to the parallel subroutines are almost identical to **rttov_direct**, **rttov_tl**, **rttov_ad** and **rttov_k**: the subroutines are named **rttov_parallel_direct**, **rttov_parallel_tl** and so on. The only difference is a final optional parameter named **nthreads** which specifies the number of threads to use. For RTTOV-SCATT there are similar **rttov_parallel_scatt**, **rttov_parallel_scatt_tl**, and **rttov_parallel_scatt_ad** interfaces.

Each thread may be assigned simulations for multiple channels (possibly across multiple profiles), but the smallest unit of computation for a single thread is a simulation for one channel for one profile. Therefore to make use of N threads, you must be simulating at least N individual channel radiances (for one or more profiles) and to obtain optimal performance you should usually be simulating many channels and/or profiles with each call to the parallel interface. For PC-RTTOV each thread is assigned at least one *profile* so you must call the parallel interface for at least N profiles to make use of N threads for PC simulations. The same applies to polarimetric instruments such as Windsat and COWVR.

If the surface emissivity and/or reflectance arguments are omitted in a call to the parallel interface for a simulation in which they are mandatory, the code will run using **calcemis** and/or **calcrefl** set to true. This is in contrast to the standard RTTOV interfaces which instead fail with an error message.

7.11. RTTOV performance

This section provides a rough guide to the relative cost of various RTTOV options and simulation types. It is difficult to be precise as execution times vary greatly depending on the compiler, compilation flags, computer architecture, the combination of RTTOV options specified at run-time, and the configuration of the input profile. Some measurements of execution time (in particular comparisons against the previous version of RTTOV) can be found in the RTTOV v12 Performance Test Logs available on the RTTOV web site.

Coefficients, trace gases, solar radiation
Simulations with coefficient files supporting more optional trace gases (related to the version of the optical depth predictors) will be slower than those with fewer optional gases. Simulations using coefficients based on v7 predictors are fastest and those using v9 predictors are slowest. MW simulations are generally faster than IR simulations (all other things being equal) as there are no optional trace gases in the MW simulations. Simulation of visible/near-IR channels (with solar radiation) is more expensive than for IR channels (no solar radiation). Simulations using coefficient files based on a larger number of levels (e.g. 101L) run slower than those with coefficients on fewer levels (e.g. 54L).

For MW simulations with CLW the run-time with the Liebe is only a little longer than without CLW. The TKC option is slightly more expensive than Liebe. However the Rosenkranz option is at least 50% more expensive than Liebe due to the relative complexity of the parameterisation. If you have more input pressure levels than coefficient levels it is generally more expensive to run the CLW calculation on the input levels, but this is nevertheless recommended.

Number of input levels, interpolation
Execution time increases with the number of levels in the input profile. Using the RTTOV interpolator increases run-time. The computational cost depends on the selected interpolation mode: the modes can be ranked from fastest to slowest as 2, 3, 1, 5, 4. For the direct model and for the TL/AD/K with the *lgradp* option set to false, the variation in cost between the modes is not very large. If *lgradp* is true (so that pressure is an active variable in the TL/AD/K) the variation in the cost betweens modes is larger. For the K model the execution cost of the interpolator vs no interpolation is much larger with *lgradp* true. Some measurements of the performance impact of interpolation vs no interpolation can be found in the RTTOV v12 Performance Test Log available on the RTTOV web site.

Surface emissivity and reflectance
Use of the internal surface emissivity and BRDF models for sea surfaces is generally more expensive than providing input values for surface emissivity and reflectance. The computational cost of ISEM is insignificant, and the cost of IREMIS is very small. For FASTEM, computation cost is modest and generally increases with the FASTEM version number (FASTEM-6 being the most expensive). The cost of TESSEM2 is approximately half that of FASTEM-6. Use of the Lambertian surface option also increases run-time slightly. The sea surface BRDF model (either option) is expensive due to its complexity, though it underwent some optimisation for v12.2. The cost of either model is comparable to that of the clear-sky radiative transfer calculation itself though the new Elfouhaily option is slightly faster than the old JONSWAP one.

Cloud/aerosol scattering
The simple cloud scheme is run "for free" with visible/IR clear-sky simulations. The cost of visible/IR scattering simulations increases with the number of layers containing scattering particles in the input profile. For aerosol simulations with Chou-scaling, the extra cost over the clear-sky case results mainly from the computation of the layer aerosol optical depths: the integration of the radiative transfer equation is the same as the clear-sky case. For cloud simulations it is both the layer optical depth calculations and also the additional cloud columns which are generated by the cloud overlap scheme which increase run-time. The radiative transfer equation is integrated for every cloud column and the number of columns can be up to twice the number of cloudy layers.

The new DOM multiple scattering solver is, broadly speaking, at least one or two orders of magnitude more expensive than Chou-scaling. However there are various factors which affect the speed of the DOM algorithm significantly (the input cloud profile, the number of DOM streams, etc). In particular, the cost of DOM simulations grows rapidly as the number of DOM streams increases, and as the number of layers containing scattering materials increases. The second of these factors particularly affects the solver for solar radiances: solar DOM simulations run significantly faster with fewer scattering layers.

Aerosol/cloud scattering simulations with explicit optical properties are slightly faster than those which use the pre-defined optical properties. For DOM simulations the effect of this becomes less noticeable as the cost of the radiative transfer calculation is larger. Using the Baran ice scheme is slightly more expensive than using the SSEC ice properties for Chou-scaling. It is significantly more expensive when using DOM and/or for solar simulations because the phase function needs to be explicitly reconstructed for the Baran scheme which is a relatively expensive calculation.

The MFASIS direct model is broadly comparable in speed to DOM with 4 streams, although as the number of cloudy layers increases DOM quickly becomes significantly more expensive while the impact on MFASIS run-times is much more modest. For the TL model and especially the AD and K models, MFASIS is much faster than DOM.

RTTOV-SCATT is, broadly speaking, a few times slower than an otherwise equivalent (clear-sky) RTTOV simulation.

PC-RTTOV/HTFRTC
For simulating a full spectrum of a hyperspectral sounder, the PC-based models are substantially faster than standard RTTOV. The PC-RTTOV K model has known performance issues, particularly with reconstructed radiances: these will be addressed in a future release. The run-time of the HTFRTC model is broadly comparable to PC-RTTOV for equivalent simulations, but as the number of variable trace gases increases, PC-RTTOV becomes relatively faster. The new simple cloud option and (in particular) the new overcast option for HTFRTC both significantly increase the run-time.

7.12. Summary of steps for running RTTOV v12

You need to ensure the following in your program which calls RTTOV:

- Create an instance of the **rttov_options** type and set any of the member flags as required. The full list of options is given in Annex O.
- Optional: you may wish to call **rttov_errorhandling** to specify the logical unit to use for error messages.
- An instance of the **rttov_coefs** type should be populated by calling **rttov_read_coefs**.
- Optional: you may wish to call **rttov_user_options_checkinput** to ensure the selected options are consistent with the coefficient file.
- Allocate the input/output structures to RTTOV with the number of channels and profiles you want to run with (e.g. by calling **rttov_alloc_direct/tl/ad/k**, see Annex D). If the emissivity and/or BRDF atlases are required these should be initialised now (see Annexes F and G). See **src/test/example_*.F90** for example code detailing the input variables required for RTTOV.
- Initialise the profile structure with your atmospheric profile. This is shown in Table 12 and listed in Annex O.
- Optional: you may wish to check the input profiles for unphysical or out-of-specification values before calling RTTOV. This can be achieved using the **rttov_user_profile_checkinput** subroutine (Annex N). If this is used, **opts%config%do_checkinput** can be set to false. You may also find the **rttov_print_profile** subroutine (Annex N) useful for debugging purposes: this prints out the contents of a single profile structure.
- Initialise the **chanprof** array with the channel and profile indexes as described in section 6.4.
- You may give a surface emissivity value in **emissivity(:)%emis_in** for each radiance calculation (for example from one of the land surface emissivity atlases), in which case you must set **calcemis(j)** to false for the desired channels. Alternatively you may let the code compute it by the use of the internal emissivity models by setting the appropriate **calcemis(j)** to true.
- For solar calculations you may supply a surface BRDF value in **reflectance(:)%refl_in** for each radiance calculation (for example from the land surface BRDF atlas), in which case you must set **calcrefl(j)** to false for the desired channels. This is recommended for land surfaces. Alternatively you may let RTTOV compute/select BRDF values internally by setting the appropriate **calcrefl(j)** to true.
- Call RTTOV (**rttov_direct**) with the input/output variables and with the coefficient structure corresponding to the instrument you want to simulate.
- When all RTTOV calls are made, then you should free memory by de-allocating the various input and output structures with the **rttov_dealloc_coef** , **rttov_alloc_direct/tl/ad/k** (see Annexes C and D). If the emissivity and or BRDF atlases were initialised, they should also be de-allocated now (see Annexes F and G).
- Most user-level RTTOV subroutines return an error status. This variable should be tested after each call: non-zero values indicate that an error occurred.
- If you want to run the cloud or aerosol options for visible/ IR sensors follow the guidance in sections 8.5/8.6.
- For microwave scattering calculations, the **rttov_scatt** routines are a level up from **rttov_direct** but they have a similar calling structure and arrays to fill (see section 8.7 and Annex M). The example program supplied **example_rttovscatt.F90** can be used as an example and similar rules apply to calling **rttov_direct**.

8. Details of specific RTTOV capabilities

8.1. Simulation of clear air radiances for infrared and microwave channels

This is the simplest mode of operation for RTTOV. The instructions given in section 7 describe the necessary steps for carrying out IR and MW clear-sky simulations. A short summary is given here: these steps are common to the majority of RTTOV simulations.

An instance of the RTTOV options structure is declared which is used to configure various aspects of the simulation. The options are described in full in Annex O. The coefficient file for the instrument of interest is then read in. For IR and MW simulations, v7, v8 or v9 predictor files may be used (see Table 4).

The atmospheric profile and surface parameters are stored in the RTTOV profile structure. Section 7.3 describes this in detail. In the case of IR sensors the optional variable trace gases available depend on the coefficient file (see Table 4). For MW sensors there are no optional trace gases, but cloud liquid water may be included as an absorbing medium. An instance of the chanprof structure is populated with the profile and channel numbers to be simulated as described in section 7.4. The input surface emissivity may be specified via the RTTOV emissivity structure or RTTOV can calculate surface emissivities internally if required. See sections 7.5 and 8.4.

Finally RTTOV is invoked by calling one of the main subroutines (or their parallel counterparts for multi-threaded execution via OpenMP): these are described fully in Annexes I, J, K and L. The outputs from RTTOV are described in section 7.8.

8.2. Simulation of clear air radiances for visible and near-infrared channels

RTTOV can also simulate clear-sky radiances for visible and near-IR channels. To include solar radiation the logical flag **opts%rt_ir%addsolar** must be set to true (see Annex O for full description of the options structure). With this flag set, the contribution of solar radiation is included in all channels with wavelengths below 5µm. Note that for channels below 3µm emission is ignored so the *only* contribution is solar radiation.

In addition to the profile variables required for IR and MW simulations, the solar zenith angle, and satellite and solar azimuth angles must also be specified in the input profile structure in **profiles(:)%sunzenangle**, **profiles(:)%azangle** and **profiles(:)%sunazangle** respectively. Note that solar radiation is only included if the solar zenith angle is less than 85º. Figure 4 illustrates the definition of azimuth angle.

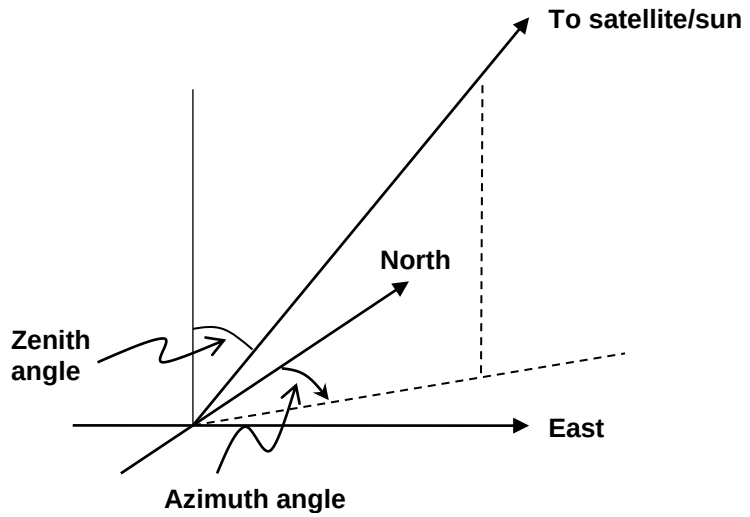

Figure 4. Azimuth angles are defined as the angle in the clockwise direction between North and the projection of the view-path or sun-surface path onto the horizontal plane at the surface.

Solar simulations can only be performed with v9 predictor coefficient files (see Table 4). These files contain a SOLAR_SPECTRUM section which includes a flag for each channel which is 0 for "thermal" channels (no significant solar contribution), 2 for "solar" channels (no significant thermally emitted contribution) and 1 for channels with both thermally emitted and solar contributions. These flags are read into the coefficients structure in **coefs%coef%ss_val_chn(:)** which has one element per channel read from the coefficient file. It is important to note that for the sake of efficiency no thermally emitted contributions are calculated for the solar channels (i.e. those with wavelengths less than 3µm). Users have noted that a solar contribution due to sun glint can be detected even in window channels in the thermal IR. By default these channels are considered thermal-only, but it is possible to include the solar contribution for them by setting the element of **coefs%coef%ss_val_chn(:)** corresponding to the channel in question to 1 after the coefficient file has been read.

The output radiance structure contains arrays for satellite-seen reflectances. These are bi-directional reflectance factors (BRFs) calculated as the ratio of the satellite seen radiance to that which would result if the surface was a perfect Lambertian reflector. BRFs are only calculated for channels with a solar contribution. Similarly, the output brightness temperature arrays will not be populated for channels with no thermally emitted contribution, and you should be careful in interpreting the output brightness temperatures for channels with both solar and emitted contributions since these may differ significantly from physical temperatures.

The SOLAR_SPECTRUM section in the RTTOV coefficient files also provides the top of atmosphere band-integrated solar irradiance at 1 AU (astronomical unit) in units of $mW/m^2/cm^{-1}$. These values are read into the **coefs%coef%ss_solar_spectrum(:)** array and are calculated using the AER solar source function software (based on Kurucz, 1992). These irradiances are used for the solar calculations: it is possible to change the values in the coefficient file or directly in the **coefs%coef%ss_solar_spectrum(:)** array after reading the coefficients if desired. If the **profile(:)%date(:)** array has been populated with a date for the profile (year, month, day) then these are used to adjust the band solar irradiances according to the Earth-sun distance. If the **date(:)** contains a date earlier than or equal to the default value (1/1/1950), no adjustment is performed. Note the date is only used to adjust the solar spectrum value for the Earth-sun distance according to the month and day of the year.

The clear-sky solar calculations comprise solar radiation reflected from the surface and, for channels with wavelengths below 2µm, a parameterisation of atmospheric Rayleigh single-scattering. Details of the solar calculations can be found in the RTTOV v11 Science and Validation Report.

For simulations including solar radiation, the surface bi-directional reflectance function (BRDF) must be specified (see section 7.6). The BRDF is defined as the ratio of reflected radiance towards the satellite to incoming solar irradiance (assuming the sun is treated as a point source). In the absence of an atmosphere the output BRF would be equal to the surface BRDF multiplied by π.

For wind-roughened sea surfaces RTTOV has an internal BRDF model as noted in section 7.6 with two choices for the wave spectrum parameterisation. To make use of this model additional profile variables should be set: the water type (fresh or salt water) **profiles(:)%watertype**, and the 10m wind and wind fetch **profiles(:)%s2m%u**, **profiles(:)%s2m%v**, **profiles(:)%s2m%wfetc**. See Annex O for details of these quantities in the profile structure and section 7.6 for more details about the sea-surface reflectances RTTOV calculates.

If **calcrefl(:)** is set to true for land or sea-ice surfaces a default BRDF of $0.3/\pi$ or $0.8/\pi$ (respectively) is used for visible and near-infrared channels: this is naturally very crude and is not generally recommended. If a land/sea-ice surface emissivity has been supplied for the channel (for example if the channel has significant emitted and solar contributions) then the BRDF is set to $(1\text{-}emissivity)/\pi$.

For solar-only channels RTTOV provides a land surface BRDF atlas (Vidot and Borbas, 2013) similar to the IR emissivity atlas. This takes as input latitude, longitude, satellite and solar zenith and azimuth angles (via the profiles structure), and provides BRDF values suitable for input to RTTOV. The BRDF atlas interface is described in detail in Annex G, and when passing these values in to RTTOV the corresponding elements of **calcrefl(:)** should be set to false.

8.3. Simple cloud

More complicated treatments of cloud and precipitation are available (see sections 8.5 and 8.7), but the simplest cloud approach described here is applied for all clear-sky visible/IR simulations. Assuming black, opaque clouds at a single level which fill the radiometer field of view the simulation of cloud affected radiances $L^{Cld}(\nu,\theta)$ is defined as:

$$L^{Cld}(\nu,\theta) = \tau_{Cld}(\nu,\theta)\,B(\nu,T_{Cld}) + \int_{\tau_{Cld}}^{1} B(\nu,T)\,d\tau \qquad (6)$$

where $\tau_{Cld}(\nu,\theta)$ is the cloud top to space transmittance and T_{Cld} the cloud top temperature specified by the cloud top pressure in the input state vector. The emissivity of the cloud top is assumed to be unity which is a tolerable assumption for optically thick water cloud at infrared radiances but not valid for optically thin cloud or any cloud at microwave frequencies. For partially cloud filled fields of view, the cloudy radiance given by equation 6 is combined with the clear sky radiance (equation 5) using the input fractional cloud cover N as follows:

$$L(\nu,\theta) = (1-N)L^{Clr}(\nu,\theta) + NL^{Cld}(\nu,\theta) \qquad (7)$$

This simple cloud scheme is activated by providing a cloud top pressure in **profiles%ctp** and a non-zero cloud fraction in **profiles%cfraction** (with 1 indicating fully overcast). The relevant radiance outputs in the radiance structure are **clear(:)** (clear radiance), **cloudy(:)** (radiance assuming 100% cloud fraction) and **total(:)** (cloudy radiance with specified cloud fraction). The cloudy radiance is calculated by interpolating the **overcast(:,:)** radiances to the cloud top pressure. Note that the simple cloud scheme is **not** applied if the full scattering calculations have been activated by setting **opts%rt_ir%addclouds** to true.

This simplistic cloud treatment has also been implemented for visible/near-infrared channels with no thermally emitted component. In this case the cloudy radiance $L^{Cld}(\nu,\theta,\theta_{sol})$, where θ_{sol} is the solar zenith angle, is calculated as the reflected component of the solar radiation from the cloud top assuming the cloud is optically thick and acts as a Lambertian reflector with a default BRDF of $0.7/\pi$ for wavelengths below 1µm and $0.6/\pi$ for wavelengths above 1µm. If the **reflectance(:)%refl_cloud_top** array contains values greater than zero, these are used in preference to the default BRDFs for the corresponding channels (see Annex O for a description of the **rttov_reflectance** structure). The cloudy radiance includes the effects of clear-sky Rayleigh scattering above the cloud top. This is a crude treatment of cloud in visible and near-infrared channels, but may be useful for qualitative applications such as simulated imagery, particularly for visible channels. Note that **refl_cloud_top** is *not* an active variable in the TL/AD/K models.

NB No solar contribution is included in the simple cloudy radiances for channels which have significant thermally emitted and solar contributions (for example, channels around 3.9µm) in accordance with the assumption that the cloud top is perfectly emissive. The simple cloud scheme is *never* applied for MW sensors.

8.4. Definition of surface emissivity

Section 7.5 describes how to use the internal emissivity models provided by RTTOV and how to provide your own input emissivities to RTTOV (including emissivities from the atlases). This section provides more details about the various emissivity models.

MW emissivity models

RTTOV incorporates the FASTEM emissivity model for MW surface emissivities and the TESSEM2 model for sea surface emissivities. There are 6 versions of the FASTEM sea surface emissivity model. FASTEM-6 is the generally recommended option for most sensors, but TESSEM2 is recommended for Metop-SG ICI (and in general for channels above 200GHz). The FASTEM options are summarised in Table 19.

The version of FASTEM to use (1 to 6) is set in the **opts%rt_mw%fastem_version** variable. TESSEM2 is selected by setting this option to zero. Any value outside the range 0-6 will result in RTTOV returning an error.

The input profile variables used by FASTEM and TESSEM2 are given in section 7.5.

For FASTEM it is also important to define the polarisation status of each channel. This is specified in the instrument coefficient file as a series of numbers in the FASTEM section and the polarisation IDs are defined in Table 20. Different polarisations are defined for cross-track scanners, conical scanners and polarimetric instruments. In general

you shouldn't need to worry about this unless you specifically want to change the polarisation of a particular microwave channel in which case you should edit the coefficient file to modify the relevant polarisation ID(s) or you can modify the values directly in **coefs%coef%fastem_polar(:)**.

FASTEM also provides emissivities for land and sea-ice surface types. This calculation uses the parameters defined in **profiles(:)%skin%fastem(1:5)**. Values for these parameters for different surface types are given in Table 21.

FASTEM-1	Fast emissivity model fitted to geometric optics model (English and Hewison, 1998).
FASTEM-2	As FASTEM-1 but with reflectivity dependence on atmospheric transmittance (Deblonde, 2000).
FASTEM-3	As FASTEM-2, but with azimuthal dependence (Liu and Weng, 2003).
FASTEM-4	Fast emissivity model as previous versions, but fitted to a new two-scale scattering model (Liu *et al*, 2011).
FASTEM-5	As FASTEM-4, but with a different foam coverage parametrisation and some fitting issues introduced with FASTEM-4 now corrected (Bormann *et al*, 2012).
FASTEM-6 ***(default)***	As FASTEM-5, but with an improved azimuthal dependence (Kazumori and English 2014 and RTTOV v12 Science and Validation Report). NB No azimuthal dependence is simulated for the 3rd and 4th Stokes channels on Windsat.

Table 19. Summary of FASTEM versions.

FASTEM Pol_ID in coef file	Definition	Examples of applicable sensors
0	Average of vertical and horizontal polarisation ie **0.5(H+V)**	SSMIS
1	Nominal vertical at nadir rotating with view angle **QV**	AMSU-A/B, MSU, MHS
2	Nominal horizontal at nadir rotating with view angle **QH**	AMSU-A, MSU, MHS
3	Vertical **V**	SSM/I, SSMIS, AMSR, Windsat
4	Horizontal **H**	SSM/I, SSMIS, AMSR, Windsat
5	+ 45 minus −45 (3rd stokes vector) **S3**	Windsat
6	Left circular - right circular (4th stokes vector) **S4**	Windsat

Table 20. Definition of polarisation status for FASTEM-2-6.

Surface type	FASTEM parameters 1:5
Typical RTTOV default for land	3. 0, 5.0, 15.0, 0.1, 0.3
Summer land surface	
Forest	1.7, 1.0, 163.0, 0.0, 0.5
Open grass	2.2, 1.3, 138.0, 0.0, 0.42
Bare soil	2.3, 1.9, 21.8, 0.0, 0.5
Winter surface type	
Forest and snow	2.9, 3.4, 27.0, 0.0, 0.0
Deep dry snow	3.0, 24.0, 60.0, 0.1, 0.15
Frozen soil	117.8, 2.0, 0.19, 0.2 ,0.35
Sea ice	
Grease ice	23.7, 7.7, 17.3, 0.0, 0.15
Baltic nilas	1.6, 3.3, 2.2, 0.0, 0.0
New ice (no snow)	2.9, 3.4, 27.0, 0.0, 0.0
New ice (snow)	2.2, 3.7, 122.0, 0.0, 0.15
Brash ice	3.0, 5.5, 183.0, 0.0, 0.0
Compact pack ice	2.0, 1700000.0, 49000000.0, 0.0, 0.0
Fast ice	1.5, 77.8, 703.0, 0.1, 0.35
Lake ice + snow	1.8, 67.1, 534.0, 0.1, 0.15
Multi-year ice	1.5, 85000.0, 4700000.0, 0.0, 0.0

Table 21. Values for FASTEM parameters for various land and sea-ice surface types. Taken from English and Hewison (1998).

IR emissivity models

The IR sea emissivity model is chosen using **opts%rt_ir%ir_sea_emis_model**. A value of 1 selects the ISEM model (Sherlock, 1999) which was included in earlier RTTOV versions and a value of 2 (the default) selects the new IREMIS model (RTTOV v12 Science and Validation Report). Section 7.5 lists the input profile variables used by each model. As noted in section 7.5 PC-RTTOV uses its own sea surface emissivity model (see also section 8.8).

For land and sea-ice surface types RTTOV uses fixed emissivity values of 0.98 (land) and 0.99 (sea-ice) where **calcemis(:)** is true.

Emissivity atlases

The IR and MW emissivity atlases are described in section 7.5. The atlases are accessed externally to RTTOV and the emissivity values are passed in **emissivity(:)%emis_in** with the corresponding elements of **calcemis(:)** set to false. The interfaces to the emissivity atlases are described in Annex F.

The CAMEL 2007 IR emissivity atlas (previously referred to as the CAMEL atlas) was new in RTTOV v12.1. This is similar in many ways to the UWIRemis atlas. One practical difference is the atlas resolution: the UWIRemis atlas data are at 0.1° resolution while the CAMEL atlas data are at 0.05° resolution (which means the CAMEL atlas requires more memory). RTTOV v12.3 introduces a third option, the CAMEL climatology atlas. As described in Borbas and Feltz (2019), this is derived from a multi-year climatology (2000-2016), provides emissivity standard deviations computed from the climatology, and includes a more sophisticated treatment of snow. For the UWIRemis and CAMEL 2007 atlases, if the profile **snow_fraction** is greater than zero, the atlas emissivity value is linearly combined with a fixed snow emissivity spectrum. In contrast, the CAMEL climatology atlas treats snow as follows:

- if the input **snow_fraction** = 1: if the atlas emissivity contains snow then the snow PC spectrum is used, otherwise the fixed snow emissivity spectrum is used.
- if 0 < **snow_fraction** < 1: if the atlas emissivity contains snow then the atlas emissivity is returned without modification, otherwise the atlas emissivity and fixed snow spectrum are linearly combined according to the **snow_fraction**.
- if **snow_fraction** = 0: if the **snow_correction** flag is true (recommended, this is an argument to **rttov_get_emis**) then if the atlas emissivity contains snow contamination the atlas will attempt to find a snow-free PC spectrum to use. If it cannot find one (or if **snow_corr** is false) it simply returns the atlas emissivity.

The PC-RTTOV land/sea coefficients were trained using the UWIRemis atlas so it is recommended (though not strictly mandatory) to use this atlas to provide land and sea-ice emissivities for PC-RTTOV.

For MW instruments, the TELSEM2 atlas and interpolator provides an updated version of the earlier TELSEM atlas which includes climatological sea-ice and is valid for channels above 200GHz (i.e. for Metop-SG ICI). The CNRM MW atlas has also been updated for RTTOV v12. This provides land surface emissivity values for AMSU-A, AMSU-B, MHS, SSMI/S and ATMS for the years 2014 and 2015.

8.5. Simulation of visible and IR cloud-affected radiances

The simple cloud calculation (equation 7, section 8.3) can be used for single layer optically thick water clouds at mid-infrared wavelengths but for more complex cloud types and/or multi-layer clouds RTTOV provides multiple scattering calculations. RTTOV v12 provides independent options for treating the thermal emission and solar source terms in visible and IR scattering simulations (note that a separate model is used for MW scattering – see section 8.7):

Discrete Ordinates Method (DOM) – thermal emission and solar source terms

The DOM solver is a pseudo-exact algorithm for solving the monochromatic radiative transfer equation. The implementation of the DOM algorithm is described in the RTTOV v12 Science and Validation Report. The number of DOM streams determines the accuracy of the simulation, but increasing the number of streams can dramatically increase the run-time. The number of streams is set in **opts%rt_ir%dom_nstreams**: this should be an even integer greater than 2 (odd values are rounded up). The default value is 8. This value should not exceed the number of Legendre coefficients for the phase functions provided as input (see below). The cloud and aerosol coefficient files contain 128 coefficients for every phase function so this is the maximum number allowed when using these pre-defined particle

types. If you supply your own optical properties (described below) there is no fixed upper limit. Experimentation suggests that there is little benefit in using more than 16 DOM streams and that 8 or 12 should be fine for most applications at least in the IR. You may wish to experiment to find the most suitable DOM parameter values for your application.

(NB "DOM streams" are not to be confused with "cloud streams" discussed elsewhere which refer to cloud columns defined by the cloud overlap scheme).

Important notes:

- By default RTTOV calculates the path of radiation through the atmosphere accounting for the curvature of the Earth (and optionally also refraction). The DOM algorithm as implemented requires a strictly plane-parallel atmospheric geometry. **Therefore if the DOM solver is being used *either* for either thermal emission *or* solar radiation calculations then the strict plane-parallel geometry is applied for *all* simulated radiances (including clear-sky radiances).** It is possible to turn on the strict plane-parallel geometry by setting **opts%rt_all%plane_parallel** to true, but it is not necessary to do so when using DOM as RTTOV does this automatically.
- When using DOM the surface is treated as a Lambertian reflector (this does ***not*** make use of the **specularity** profile variable). When solving for thermal emission the albedo is calculated as 1-*emissivity*. When solving for solar radiation the albedo is equal to the input or calculated (sunglint) BRDF multiplied by π: if this value exceeds one the albedo is capped at one.
- The DOM algorithm is relatively slow and has relatively large memory requirements. The number of DOM streams and the number of layers in the input profile have a direct impact on the speed. In addition, the more layers containing scattering particles, the slower it runs. There are two parameters which can be used to increase efficiency:
    - **opts%rt_ir%dom_accuracy**: for solar radiation the azimuthal loop within the DOM algorithm will exit early if the computed radiance increment is smaller than this fraction of the total radiance (actually if this condition has been met twice). This is identical to the DISORT accuracy parameter. The default is 0. (i.e. no truncation). This option should be used with some caution: it is recommended that you carry out sensitivity tests.
    - **opts%rt_ir%dom_opdep_threshold**: if this value is greater than zero, then the DOM solver excludes any atmospheric layers below the level at which the absorption optical depth to space exceeds this value. This typically only affects IR channels where there is significant atmospheric absorption. A value of around 10. can decrease run-time noticeably with negligible impact on radiances. The default is 0. (i.e. no layers are excluded).

  In addition, use of **opts%rt_ir%cldstr_threshold** to exclude cloud streams (columns) with small weights can have a significant impact on the run-time and memory requirements, subject to the caveats discussed below regarding this option.

The optical properties of scattering particles required for this solver are:

- absorption and scattering coefficients
- at least **opts%rt_ir%dom_nstreams** coefficients of the Legendre decomposition of the phase function
- for solar radiation the full phase function is also required

Chou-scaling – thermal emission source term only

This is the fast multiple-scattering parameterisation for IR instruments available in previous versions of RTTOV and described in the RTTOV v9 Science and Validation Report (available on the NWP SAF web site). The optical properties of scattering particles required for this solver are:

- absorption and scattering coefficients
- the "b" parameter (representing the fraction of back-scattered radiation) computed from the phase function

Single-scattering – solar source term only

The solar single-scattering capability from previous versions of RTTOV for clouds and aerosols remains available as an option, although it is not an accurate method especially in cases where scattering dominates over absorption. It is not generally recommended. The optical properties of scattering particles required for this solver are:

- absorption and scattering coefficients
- the full phase function

MFASIS – solar source term only

The MFASIS fast visible scattering parameterisation (Scheck *et al*, 2016) was introduced in RTTOV v12.2. MFASIS is described in more detail in a separate section below.

Rayleigh scattering

As described in section 8.2 RTTOV includes a single-scattering approximation for atmospheric (molecular) Rayleigh scattering. This is also applied in the case of cloud and aerosol scattering simulations. However when using the DOM solver there is no interaction between the Rayleigh single-scattered radiation and the cloud or aerosol particles. This is primarily done for reasons of efficiency: if Rayleigh scattering is included in the DOM multiple-scattering calculations then every layer contains scattering particles which would result in extremely slow calculations (as noted above). In the case of thick cloud, for example, this can result in an under-estimation of the TOA radiance in visible channels, especially for larger satellite or solar zenith angles.

The scattering models for thermal emission and solar radiation are selected independently using the following options:

Thermal emission	**opts%rt_ir%ir_scatt_model**	1=>DOM 2=>Chou-scaling (default)
Solar radiation	**opts%rt_ir%vis_scatt_model**	1=>DOM (default) 2=>single-scattering 3=>MFASIS (cloud only)

RTTOV provides two methods of specifying the optical properties of the scattering particles.

Method 1 – use pre-defined optical properties: specify abundance profiles for the pre-defined particle types listed in Table 24. For ice clouds there is a choice regarding the parameterisation of the optical properties of the ice particles.

Method 2 – provide optical properties explicitly: supply profiles of the scattering optical properties for each instrument channel directly. This provides greater flexibility as you are not limited to the pre-defined particle types, but it is a slightly more complicated way of calling RTTOV.

Note that all visible/IR scattering-related inputs are provided on **layers** rather than levels where layer **i** is bounded by input levels **i** and **i+1**.

For both methods 1 and 2

Cloud scattering simulations are enabled by setting **opts%rt_ir%addclouds** to true. The choice between methods 1 and 2 is made by setting **opts%rt_ir%user_cld_opt_param** to false for method 1 (the default) or true for method 2.

For cloud simulations a cloud fraction profile must be specified in **profiles(j)%cfrac(:)** for each profile **j**. This specifies the total fractional coverage of all cloud in each layer with 0 is no cloud and 1 is overcast: for method 1 the cloudy fraction is assumed to contain a mixture of cloud types, each with their specified concentration (see below). The remaining fraction of the layer is assumed to be clear.

When running the tangent linear (TL), adjoint (AD) or K models, you are advised to avoid specifying layers with a **cfrac** equal to 1.0. Instead a value very close to 1.0 should be used (e.g. 0.999999). In addition you are advised not to specify identical values of **cfrac** on adjacent layers. The reason for this is that the **cfrac** Jacobians are very sensitive to perturbations in these cases (the direct model is not differentiable for fully overcast layers or where identical values of **cfrac** are in adjacent layers). If this advice is not followed, RTTOV will make very small adjustments to the input **cfrac**

profile in accordance with the above advice to ensure consistency between the direct, TL, AD and K models. These adjustments are sufficiently small to not change the direct model radiances. These restrictions on the values in **cfrac** do not apply when running the direct model alone.

As detailed in the RTTOV v9 Science and Validation Report, the computation of cloud affected radiances is performed by dividing the computed atmospheric path into a number of independent cloud streams (or columns). The number of cloud streams used for the scattering calculation is computed internally by RTTOV. It is possible to reduce the number of cloud streams and save time and memory by considering only those cloud streams whose weight is larger than the variable **opts%rt_ir%cldstr_threshold**. By setting **cldstr_threshold** to a negative number (or zero), all the cloud streams will be processed (the default). This feature should be used with caution. Since the sum of the weights of all cloud streams (including the clear one) must be equal to 1, if some cloud streams are excluded, the weight of the clear stream has to be adjusted to a greater value. Consequently, if the value used for **cldstr_threshold** is too large, this can result in a disproportionate weight of the clear stream with negative implications for the accuracy of the results. You should select the value of **cldstr_threshold** in order to remove only the cloud streams with a very small weight. These cloud streams have little impact on the total radiance and their exclusion can result in a sensible reduction of the computational time required for cloud affected computations. Note however that by default **cldstr_threshold** is set to a negative number and this is the recommended setting if running the TL, AD or K models to ensure the sensitivity to **cfrac** is correctly computed in the TL, AD or Jacobian output.

Method 1 details – using pre-defined optical properties

As of RTTOV v12.3 there are two options for cloud liquid water optical properties and three options for ice cloud optical properties. These are specified in **profiles(:)%clw_scheme** and **profiles(:)%ice_scheme** respectively.

Cloud liquid water – clw_scheme = 1 (OPAC)
These are the optical properties for 5 OPAC water cloud types available in previous versions of RTTOV (see Table 24). These represent 5 different size distributions. The layer cloud liquid concentrations for one or more of the 5 types are specified in **profiles(:)%cloud(1:5,lay)** for layer **lay**.

Cloud liquid water – clw_scheme = 2 ("Deff")
These properties were new in v12.2 and are consistent with Mie cloud liquid water optical properties available with the libRadtran package (Mayer and Kylling, 2005; http://www.libradtran.org). This is a single particle type with properties parameterised in terms of particle effective diameter. Cloud liquid water concentration for layer **lay** can be specified in any of **profiles(:)%cloud(1:5,lay)**. Note that in this case all 5 columns are treated as one single particle type: RTTOV will sum up the cloud concentrations in elements 1:5. In addition you must specify the cloud liquid water effective diameter in microns in the **profiles(:)%clwde(:)** array. This value must be non-zero for all layers containing non-zero CLW concentration, otherwise the layer will be treated as clear.

Cloud ice water – ice_scheme = 1 (SSEC/Baum)
These properties (Baum *et al* 2011) were introduced in RTTOV v12.1 and are parameterised in terms of ice effective diameter. Layer ice cloud concentrations are input in **profiles(:)%cloud(6,:)**. For ice particles RTTOV provides four parameterisations of diameter in terms of temperature and ice water content (see Table 23). This is specified in **profiles(:)idg**. Alternatively you can provide explicit values for the diameter in **profiles(:)%icede(:)** (in microns). For layers containing cloud ice water the supplied **icede(:)** value will be used if non-zero, otherwise the selected parameterisation will be used.

Cloud ice water – ice_scheme = 2 or 3 (Baran 2014, Baran 2018)
The Baran ice optical property parameterisation is described in Vidot *et al* (2015). The new 2018 parameterisation is more spectrally consistent and as such is recommended over the 2014 parameterisation. As for theSSEC/Baum scheme the ice cloud concentrations are input in **profiles(:)%cloud(6,:)**. The Baran database has no explicit dependence on particle size so the **idg** and **icede(:)** profile variables are ignored. Note that when using the DOM solver the Legendre coefficients for the phase functions are calculated at run-time and as such the Baran scheme is slower than using the SSEC ice properties with the DOM solver.

All of these properties are summarised in Tables 22 and 23. These tables also give valid ranges of profile variables associated with the various options. If the input parameters exceed the given limits, RTTOV uses the limit value(s) instead in the optical property calculations. The exception is IWC for the SSEC ice database: a warning is given by **rttov_user_profile_checkinput** if the values are out of bounds, but the hard limits are not imposed by the code.

Both sets of cloud liquid water optical properties and the SSEC/Baum optical properties are stored in coefficient files whose names begin "*sccldcoef_*". The Baran ice scheme optical properties are parameterised within the code. When reading the RTTOV coefficient files (using the **rttov_read_coefs** subroutine – see section 7.2) the **opts%rt_ir%addclouds** option must be true and the cloud coefficient file is read in at the same time as the optical depth coefficient file. The cloud coefficients must be defined for the same set of channels as the optical depth coefficients: there are separate cloud scattering coefficient files available (by default) in **rtcoef_rttov12/cldaer_ir/** and **rtcoef_rttov12/cldaer_visir/** which should be used with the IR-only (v7 and v8 predictor optical depth coefficient files and hyper-spectral sounder coefficient files) and with the visible+IR coefficient files respectively.

When populating the RTTOV profiles structure, the cloud fraction profile is specified in **profiles(:)%cfrac(:)** as described above. The cloud liquid and ice water densities are specified in **profiles(:)%cloud(1:6,lay)** for layer **lay** as described above. The units for liquid and ice water content are determined by the logical flag **profiles(:)%mmr_cldaer**: if this is true (the default) the units are kg/kg, if the flag is false the units are g/m^3 as in previous RTTOV versions. Note that this setting affects units of both clouds and aerosols. All profiles passed into RTTOV in a single call must use the same units.

As of RTTOV v12.3 the cloud concentration supplied to RTTOV may be a grid box average value which is often the quantity provided by GCMs. In this case you must set the **opts%rt_ir%grid_box_avg_cloud** option to true. In previous versions of RTTOV it was necessary to supply the concentration of cloud in the cloudy fraction of the layer: this is equal to the grid box average concentration divided by the cloud fraction. In this case the **opts%rt_ir%grid_box_avg_cloud** option must be set to false (the default).

The cloud concentration can be non-zero for multiple cloud types in any layer: all cloud types present are assumed to be mixed together within the fraction of the layer specified by **cfrac(:)**, with the remaining fraction of the layer assumed to be clear. If either **cfrac(j)** or all **cloud(1:6,j)** are zero for layer **j** then there is no contribution to the simulated radiance from cloud in this layer. An example input cloud and cloud fraction profile is given in file: **rttov_test/test_example.1/cld_prof.dat**.

To compute cloudy radiances via method 1:

- Set **opts%rt_ir%addclouds** to true
- Set **opts%rt_ir%user_cld_opt_param** to false (this is the case by default)
- Ensure the cloud scattering coefficient file is read in the call to **rttov_read_coefs** (see Annex C). The naming convention for these files is *sccldcoef_msg_3_seviri.dat* where SEVIRI on MSG-3 is the sensor in this case. Remember that the cloud coefficient file must match the optical depth coefficient file (IR-only or visible+IR).
- Specify the units of cloud concentration in **profiles(:)%mmr_cldaer** (true by default for units of kg/kg).
- Populate the input **profiles(:)%cloud(:,:)** array with liquid or ice water/ice concentrations (grid box average concentration if **grid_box_avg_cloud** is true, or concentraton of cloudy fraction of layer otherwise) in appropriate units for each cloud type (the first index is the cloud type, the second is layer number).
- Populate the input **profiles(:)%cfrac(:)** array with the layer cloud fractions from 0-1.
- Specify the treatment of cloud liquid and ice particles according to the options in Tables 22 and 23.

An example program **src/test/example_cld_file_fwd.F90** has been created which demonstrates these steps for cloud scattering and can be used as a template for your own program.

OPAC CLW properties	**"Deff" CLW properties**
Set **profiles(:)%clw_scheme = 1**	Set **profiles(:)%clw_scheme = 2**
Specify vertical profiles of concentrations for one or more cloud liquid water types (see Table 24 below) in **profiles(:)%cloud(1:5,:)**	Properties are available for one CLW type. Specify the vertical profile of concentration in, say, **profiles(:)%cloud(1,:)** NB RTTOV will sum CLW concentrations in **profiles(:)%cloud(1:5,lay)** for each layer **lay**.
	Specify a vertical profile of CLW effective diameter in **profiles(:)%clwde(:)**
	Min eff. diameter: 2µm Max eff. diameter: 52µm

Table 22. Summary of cloud liquid water particle parameterisation options and associated parameter limits.

SSEC ice properties		**Baran 2014 or 2018 ice schemes**
Set **profiles(:)%ice_scheme = 1**		Set **profiles(:)%ice_scheme = 2** or **3**
Specify ice effective diameter in microns for each layer in **profiles(:)%icede(:)**	Specify parameterisation for effective diameter in Set **profiles(:)%idg** 1 = Ou and Liou (1995) 2 = Wyser (1998) (recommended) 3 = Boudala et al (2002) 4 = McFarquar et al (2003) and ensure **profiles(:)%icede(:)** is zero.	No explicit dependence on ice effective diameter: **icede(:)** and **idg** are ignored.
Min eff. diameter = 10 µm Max eff. diameter = 120 µm Min IWC = 4.984E-5 g m^{-3} Max IWC = 0.1831 g m^{-3} Only effective diameter limits are imposed in the code.		Min T = 193.157 K Max T = 273.127 K Min IWC = 6.0E-06 g m^{-3} Max IWC = 1.969466 g m^{-3}

Table 23. Summary of ice particle parameterisation options and associated parameter limits.

Column 1:	Stratus Continental	STCO
Column 2:	Stratus Maritime	STMA
Column 3:	Cumulus Continental Clean	CUCC
Column 4:	Cumulus Continental Polluted	CUCP
Column 5:	Cumulus Maritime	CUMA
Column 6:	Ice cloud (all types despite the name "CIRR")	CIRR

Table 24. Cloud types available in RTTOV v12.

Method 2 details – providing optical properties explicitly

Instead of using the pre-defined optical properties you can input your own. In this case a vertical profile is provided for each channel containing the optical property data for each layer. No cloud coefficient file us required and the **profiles(:)%cloud(:,:)** array is not used, but **profiles(:)%cfrac(:)** must still be provided. The cloud optical property profiles are input to RTTOV using the **cld_opt_param** argument (of derived type **rttov_opt_param**). This comprises arrays for each scattering parameter required by RTTOV for each layer and channel. The "nchanprof" dimension corresponds to the channels in the **chanprof(:)** array. Note that the properties specified apply specifically to the cloudy fraction in each layer: the **opts%rt_ir%grid_box_avg_cloud** flag is ignored for method 2. The members of the **rttov_opt_param** structure are:

- **abs(:,:)** : absorption coefficient (units: km^{-1}), dimensions (**nlayers, nchanprof**). Always required.
- **sca(:,:)** : scattering coefficient (units: km^{-1}), dimensions (**nlayers, nchanprof**) . Always required.
- **bpr(:,:)** : "*b* parameter" (no units), dimensions (**nlayers, nchanprof**). Only required by the Chou-scaling parameterisation for channels with thermal emission (wavelength > 3µm). This is the fraction of back-scattered radiation from each layer and is calculated from the phase function using a supplied subroutine (see below).
- **nmom** : number of Legendre coefficients specified for all phase functions. Only required by the DOM solver.
- **legcoef(:,:,:)** : coefficients of Legendre decomposition of phase function (no units), dimensions (**1:nmom+1, nlayers, nchanprof**). Only required by the DOM solver (for thermal emission or solar radiation). These can be computed using a supplied subroutine (see below). Note that the "zeroth" moment in **legcoef(1,:,:)** must always be one and **nmom** coefficients are supplied in addition to this.
- **phangle(:)** : the angles over which the phase functions are defined (units: degrees), dimension (**nphangle**). This should cover the full range of scattering angles monotonically from 0° to 180° inclusive. The angle grid does not have to be evenly-spaced.
- **pha(:,:,:)** : azimuthally-averaged phase function, dimensions (**nphangle, nlayers, nchanprof**). Required by scattering simulations for solar radiation and used for the calculation of the *b* parameter and Legendre coefficients. Phase functions should be normalised such that the integral over all scattering angles is 4π. Given phase function P defined over scattering angles Θ from 0 to π, and zenith angle θ and azimuthal angle ϕ we require

$$4\pi = \int_0^{\pi} P(\Theta)d\Theta = \int_0^{\pi} P(\theta)\int_0^{2\pi} \sin(\theta)d\phi d\theta$$

so that:

$$1 = \tfrac{1}{2}\int_0^{\pi} P(\theta)\sin(\theta)d\theta$$

A variable **cld_opt_param** of type **rttov_opt_param** should be declared and then allocated by calling **rttov_alloc_opt_param** (see Annex D). (This subroutine should also be called to deallocate the structure at the end of your program). The various parameters required by the scattering model(s) being used, as described above, should be specified. The phase functions for pre-defined particle types are defined over one of two angular arrays both of which can be found in the **rttov_const** module as **phangle_lores(1:208)** and **phangle_hires(1:498)**. Alternatively you can define your own angle grid.

A subroutine is provided to calculate the *b* parameters from the phase functions. This is relatively slow and so is not performed internally within RTTOV. You may find it beneficial to calculate the *b* parameters off-line and store them for future use if this is practical for your application. To generate the values:

- call **rttov_bpr_init** to initialise some tables to speed up the calculation
- call **rttov_bpr_calc** to calculate the *b* parameters from the phase functions: this is called once for every phase function (i.e. for each layer containing scattering particles and for each channel)
- call **rttov_bpr_dealloc** to release allocated memory

In addition the **rttov_legcoef_calc** subroutine is provided to calculate the Legendre coefficients for a phase function. This is quite fast and you only need to calculate as many coefficients as DOM streams you will specify for your simulations.

The interfaces for all these subroutines are described in Annex E.

Note that if you already have values for **bpr** and/or the Legendre coefficients and you are not running solar simulations (i.e. **opts%rt_ir%addsolar** is false) then the **phangle** and **pha** arrays (i.e. phase angles and phase functions) do *not* need to be populated for the call to RTTOV.
Finally, if solar scattering calculations are being performed, you must call the **rttov_init_opt_param** subroutine (see Annex D) to precalculate some values related to the phase angles. This is not required if **opts%rt_ir%addsolar** is false.

To compute cloudy radiances via method 2:

- Set **opts%rt_ir%addclouds** to true
- Set **opts%rt_ir%user_cld_opt_param** to true
- Declare a variable of type **rttov_opt_param**, for example **cld_opt_param**, and allocate its member arrays by calling **rttov_alloc_opt_param**
- Populate **cld_opt_param** with absorption and scattering coefficients and, for solar radiation, the phase function and phase angles
- Populate **cld_opt_param** with the *b* parameters (if using the Chou-scaling parameterisation) either from pre-calculated data or by calling **rttov_bpr_calc**
- Populate **cld_opt_param** with the Legendre coefficients (if using the DOM solver) either from pre-calculated data or by calling **rttov_legcoef_calc**
- If performing solar calculations call **rttov_init_opt_param**
- Populate the input **profiles(:)%cfrac(:)** array with cloud fraction from 0-1
- Pass the **cld_opt_param** argument into **rttov_direct** (or **_tl/ad/k**)
- Optionally pass **cld_opt_param_tl/ad/k** into **rttov_tl/ad/k** respectively (see section 7.9).
- When finished with RTTOV call **rttov_alloc_opt_par**am again to deallocate **cld_opt_param**.

An example program **src/test/example_cld_param_fwd.F90** has been created which demonstrates these steps for cloud scattering and can be used as a template for your own programs.

MFASIS fast visible scattering model

The MFASIS fast cloud scattering parameterisation (Scheck *et al*, 2016) has been incorporated into RTTOV. This is a look-up table (LUT) based model which currently enables cloud simulations for visible channels (below ~1µm).

An example program **src/test/example_cld_mfasis_fwd.F90** has been created which demonstrates a call to the MFASIS direct model and can be used as a template for your own programs.

In RTTOV, MFASIS simulations are performed in the same way as other cloud simulations through calls to the RTTOV direct/TL/AD/K models, but there are a few key differences. The MFASIS model does not use the RTTOV optical depth parameterisation and therefore does not use the RTTOV interpolator. Optional trace gases are not enabled in the MFASIS simulations. MFASIS simulations ignore the setting of **opts%config%do_checkinput** (see section 7.3): the input profile is *not* checked for invalid data. You can use the **rttov_user_profile_checkinput** subroutine, but note that this tests all profile variables, including those which are not used by MFASIS (see below). MFASIS simulations enforce the strict plane-parallel geometry (as DOM does). MFASIS simulations cannot be run with aerosols and they are not compatible with the explicit optical property inputs. When running MFASIS simulations, radiances and reflectances are only returned for channels supported in the MFASIS LUT file. Therefore you cannot run simulations for IR and visible channels with MFASIS in the same call to RTTOV.

The input profile variables used by MFASIS are given in Table 25. All other profile inputs are ignored. MFASIS assumes a Lambertian surface with albedo given by *pi* times the surface BRDF and this value is capped at one, as for DOM. The surface BRDF options are the same as for other RTTOV calls.

MFASIS is selected by setting both **opts%rt_ir%addclouds** and **opts%rt_ir%addsolar** to true and setting **opts%rt_ir%vis_scatt_model** = 3. You must set these options ***before*** the call to **rttov_read_coefs** to read in the various coefficient files. You can change, for example, the selected solar scattering model later in your code if desired.

MFASIS requires a standard RTTOV *rtcoef* file, an *sccldcoef* cloud optical properties file, and an MFASIS cloud LUT file for the sensor you wish to simulate. All three files are read in the same call to **rttov_read_coefs** (see Annex C). Due to their size the MFASIS LUTs are provided in HDF5 format. There is no problem reading ASCII *rtcoef* and *sccldcoef* files at the same time as an HDF5 MFASIS LUT. You must use the v9 predictor *rtcoef* files and the associated *sccldcoef* cloud property files.

Profile structure variables	**Notes**
gas_units	Mandatory
p(:), t(:), q(:)	Mandatory
s2m%p	Mandatory
zenangle, azangle, sunzenangle, sunazangle	Mandatory
skin%surftype	Mandatory
elevation	Mandatory
latitude	Mandatory
s2m%u, s2m%v, s2m%wfetc, skin%watertype	Used by solar sea BRDF model
mmr_cldaer	Mandatory
clw_scheme	Mandatory
clwde(:)	Required if clw_scheme = 2
ice_scheme	Mandatory
idg, icede(:)	Required if ice_scheme = 1 (see Table 23)
cfrac(:)	Mandatory
cloud(:,:)	Mandatory

Table 25: profile variables used by MFASIS.

MFASIS LUTs are trained using specific cloud liquid and ice water optical properties, as determined by the **clw_scheme** and **ice_scheme** profile variables (as described above). When running MFASIS simulations the **clw_scheme** and **ice_scheme** variables must match those in the LUT. All MFASIS LUTs are currently trained using **ice_scheme = 1** (the SSEC/Baum properties). LUTs are available based on both the OPAC and new Deff CLW optical properties, as indicated by the LUT filenames. After reading a LUT you can determine the properties used to train it from the **coefs%coef_mfasis_cld%clw_scheme** and **coefs%coef_mfasis_cld%ice_scheme** variables.

When running MFASIS you can specify any channel selection valid for the *rtcoef* file, but only channels supported by the MFASIS LUT file will be simulated. Radiances for other channels will be zero on exit.

There are various conditions which can cause possible issues with MFASIS radiances. The most common ones related to certain combinations of solar and viewing geometry which can result in questionable accuracy, or parameters which lie beyond the range of values used for training the LUTs. If relevant, warning flags are set in the **radiance%quality(:)** output array described in section 7.8. You should check whether or not the **quality(:)** array contains non-zero values (indicating potential issues) after calling RTTOV. The following warning flags may be set by MFASIS simulations (the constants referred to are defined in the **rttov_const** module):

- **qflag_mfasis_zenangle**: the sallite and solar zenith angles should each be less than **mfasis_maxzenangle**=85°.
- **qflag_mfasis_sumzenangle**: the sum of the satellite and solar zenith angles should be less than **mfasis_maxsumzenangle**=150°.
- **qflag_mfasis_loscaangle/qflag_mfasis_hiscaangle**: the scattering angle (see below) must lie in the range [**mfasis_minscaangle**, **mfasis_maxscaangle**] = [9°, 140°].
- **qflag_mfasis_geometry_bounds**: the scattering angle should lie within the range of values used for training the LUT (see below).
- **qflag_mfasis_opdpedia_bounds**: this flag indicates that a total optical depth or effective diameter value (for liquid or ice cloud) has exceeded the LUT bounds (see below). For the LUTs available on the RTTOV website this flag does not necessarily indicate a problem if the optical depths exceed the maximum values used in training as the liquid and ice water cloud optical depth maxima are large.

The scattering angle is computed as follows, with *scat_angle*=0° occurring when *zen_sat* and *zen_sun* are equal and *rel_azi*=0° (*rel_azi* is the difference between the solar and satellite azimuth angles):

$$cos(scat_angle) = cos(zen_sat) * cos(zen_sun) + sin(zen_sat) * sin(zen_sun) * cos(rel_azi)$$

Table 26 lists the minimum and maximum values used in training the MFASIS LUTs available on the RTTOV website at the time of release of v12.3. These ranges are given in the README_LUT section of the MFASIS LUT files. You can print out information about an MFASIS LUT using the **rttov_mfasis_lut_info.exe** executable (see Annex A) including the training bounds of each dimension. If a variable exceeds the bounds used in training MFASIS, the closest value is used from the LUT: there is no extrapolation beyond the bounds of the training data. Note that as the surface albedo is capped at one within RTTOV, this can never fall outside the bounds.

MFASIS variable	Minimum	Maximum
Scattering angle	9°	140°
Cloud liquid water total optical depth	0	1000
Cloud ice water total optical depth	0	300
Cloud liquid water total Deff (Deff CLW optical properties only)*	5µm	50µm
Cloud ice water total Deff	10µm	120µm
Surface albedo	0	1

Table 26. Ranges of values used in training MFASIS LUTs available on the RTTOV website.
**For LUTs trained using the OPAC optical properties, the Deff value is fixed for each of the 5 OPAC particle types. The LUT data are computed for these 5 specific Deff values so the LUT is never interpolated in this dimension and can never give out-of-bounds errors.*

8.6. Simulation of visible and IR aerosol-affected radiances

The aerosol-affected simulations are very similar to the cloud-affected simulations so you should read section 8.5 as most of the information is relevant. The same scattering models are available for aerosols and clouds (except MFASIS) and, as for clouds, there are two methods for running aerosol simulations.

For both methods 1 and 2

Aerosol scattering simulations are enabled by setting **opts%rt_ir%addaerosl** to true. The choice between methods 1 and 2 is made by setting **opts%rt_ir%user_aer_opt_param** to false for method 1 (the default) or true for method 2.

Due to the way in which the aerosol-affected radiances are calculated, the "clear-sky"outputs in the **rttov_radiance** structure (e.g. **radiance%clear, radiance%bt_clear, radiance%refl_clear**) include the effects of aerosol when aerosols are included.

Method 1 – using pre-defined optical properties

RTTOV v12 now provides two sets of files containing aerosol optical properties. The pre-existing OPAC files define thirteen types of aerosol particle. New files have been generated containing optical properties for nine species from the CAMS aerosol model. These optical properties are contained in coefficient files whose names begin "*scaercoef_*". The new CAMS files contain "*_cams*" in the filenames. When reading the RTTOV coefficient files (using the **rttov_read_coefs** subroutine – see section 7.2) the **opts%rt_ir%addaerosl** option must be true and the aerosol coefficient file is read in at the same time as the optical depth coefficient file. The aerosol coefficients must be defined for the same set of channels as the optical depth coefficients: there are separate aerosol scattering coefficient files available (by default) in **rtcoef_rttov12/cldaer_ir/** and **rtcoef_rttov12/cldaer_visir/** which should be used with the IR-only (v7 and v8 predictor optical depth coefficient files and hyper-spectral sounder coefficient files) and with the visible+IR coefficient files respectively.

When populating the RTTOV profiles structure, the aerosol concentrations are specified in **profiles(:)%aerosols(i,j)** where index **j** is the layer number and index **i** runs from 1-13 (OPAC) or 1-9 (CAMS) and represents each of the aerosol particle types listed in Table 27. The units are determined by the logical flag **profiles(:)%mmr_cldaer**: if this is true (the default) the units are kg/kg, if the flag is false the units are number density in cm^{-3} as in previous RTTOV versions. Note that this setting affects units of both clouds and aerosols. All profiles passed into RTTOV in a single call must use the same units. For most aerosol types (both OPAC and CAMS) the input is dry aerosol mass ratio. The only exception are the CAMS sea salt types for which the input is mass ratio at 80% relative humidity. This is consistent with outputs from the CAMS model.

The CAMS species are described in Bozzo *et al* (2017). For the OPAC aerosol files, particle types 1-11 are described in detail in Matricardi (2005). The new volcanic ash particle type (number 12) uses a log-normal size distribution function calculated for radii between 0.005 and 20μm with parameters derived from aircraft measurements of the 2010 Icelandic eruption (Johnson *et al*, 2012). The refractive indices are from Pollack *et al* (1973). The optical parameters for the new Asian dust particle type (number 13) are calculated using a linear combination of size distributions for the MINM, MIAM and MICM aerosol particles for radii between 0.01 and 60μm: the weights were obtained by fitting to a particle size distribution derived from sky radiometer measurements made at Dunhuang, China (Han *et al*, 2012). The refractive indices are from Volz (1972, 1973).

RTTOV v12.3 provides a new tool **rttov_make_scaercoef.exe** described in Annex N which allows you to generate custom aerosol optical property files for use with RTTOV. These can contain an arbitrary number of aerosol species: in this case when the **profiles** structure is allocated, the second dimension of the **profiles(:)%aerosols(:,:)** array is the number of aerosol species in your *scaercoef* file and you specify the species concentrations in the order in which they are defined in the file.

You can define any mix of the various aerosol components/species defined in the aerosol property file. An example input aerosol profile using the OPAC types is given in file: **rttov_test/test_example.1/aer_prof.dat**. RTTOV also provides a subroutine to calculate various climatological aerosol profiles with pre-defined mixtures of OPAC components (see the RTTOV v9 Science and Validation Report). The binary **create_aer_clim_prof.exe** (source: **src/other/create_aer_clim_prof.F90**) (see Annex N) can be used to generate climatological aerosol profiles off-line. The list of climatological compositions output by this program is also shown in Annex N. The file **data/prof_aerosl_cl.dat** contains climatological aerosol profiles generated using the standard RTTOV 101 pressure levels with T and q profiles taken from the file **data/prof.dat** for a latitude of zero, surface elevation of zero, surface level 101 and scale factor 1.0. You may wish to re-run the executable to generate your own sets of aerosol profiles for different input parameters, profiles or numbers of levels. Alternatively, the example program **src/test/example_aer_file_fwd.F90** gives an example of calling the **rttov_aer_clim_prof.F90** subroutine (see Annex N) to obtain climatological profiles at run-time.

To compute aerosol-affected radiances via method 1:

- Set **opts%rt_ir%addaerosl** to true
- Set **opts%rt_ir%user_aer_opt_param** to false (this is the case by default)
- Ensure the aerosol scattering coefficient file is read in the call to **rttov_read_coefs** (see Annex C). The naming convention for these files is *scaercoef_msg_3_seviri.dat* where SEVIRI on MSG-3 is the sensor in this case. Remember that the aerosol coefficient file must match the optical depth coefficient file (IR-only or visible+IR).
- Specify the units of aerosol concentration in **profiles(:)%mmr_cldaer** (true by default for units of kg/kg).
- Populate the input **profiles(:)%aerosol(:,:)** array with mean layer aerosol concentration in appropriate units for each aerosol type.

An example program **src/test/example_aer_file_fwd.F90** has been created which demonstrates these steps for aerosol scattering and can be used as a template for your own programs.

Column	OPAC aerosol components		CAMS aerosol species	
1	Insoluble	INSO	Black Carbon	BCAR
2	Water soluble	WASO	Dust, bin 1, 0.03-0.55 micron, ref. index: Woodward 2001	DUS1
3	Soot	SOOT	Dust, bin 2, 0.55-0.90 micron, ref. index: Woodward 2001	DUS2
4	Sea salt (acc mode)	SSAM	Dust, bin 3, 0.90-20.0 micron, ref. index: Woodward 2001	DUS3
5	Sea salt (coa mode)	SSCM	Ammonium sulphate	SULP
6	Mineral (nuc mode)	MINM	Sea salt, bin 1, 0.03-0.5 micron	SSA1
7	Mineral (acc mode)	MIAM	Sea salt, bin 2, 0.5-5.0 micron	SSA2
8	Mineral (coa mode)	MICM	Sea salt, bin 3, 5.0-20.0 micron	SSA3
9	Mineral transported	MITR	Hydrophilic organic matter	OMAT
10	Sulphated droplets	SUSO		
11	Volcanic ash	VOLA		
12	New volcanic ash	VAPO		
13	Asian dust	ASDU		

Table 27. OPAC aerosol components and CAMS aerosol species.

Method 2 – providing optical properties explicitly

This is very similar to Method 2 for cloud profiles except there is no cloud fraction. You should read the description of Method 2 above for clouds: the same parameters are required for the various scattering models. The optical property data are input to RTTOV via the **aer_opt_param** argument of derived type **rttov_opt_param**.

To compute aerosol-affected radiances via method 2:

- Set **opts%rt_ir%addaerosl** to true
- Set **opts%rt_ir%user_aer_opt_param** to true
- Declare a variable of type **rttov_opt_param**, for example **aer_opt_param**, and allocate its member arrays by calling **rttov_alloc_opt_param**
- Populate **aer_opt_param** with absorption and scattering coefficients and, for solar radiation, the phase function and phase angles
- Populate **aer_opt_param** with the *b* parameters (if using the Chou-scaling parameterisation) either from pre-calculated data or by calling **rttov_bpr_calc**
- Populate **aer_opt_param** with the Legendre coefficients (if using the DOM solver) either from pre-calculated data or by calling **rttov_legcoef_calc**
- If performing solar calculations call **rttov_init_opt_param**
- Pass the **aer_opt_param** argument into **rttov_direct** (or **_tl/ad/k**)
- Optionally pass **aer_opt_param_tl/ad/k** into **rttov_tl/ad/k** respectively (see section 7.9).
- When finished with RTTOV call **rttov_alloc_opt_param** again to deallocate **aer_opt_param**.

An example program **src/test/example_aer_param_fwd.F90** has been created which demonstrates these steps for aerosol scattering and can be used as a template for your own programs.

8.7. Simulation of microwave radiances scattered by cloud and precipitation

A separate interface, known as RTTOV-SCATT, is provided for simulating microwave radiances affected by cloud and precipitation. The scattering effects of hydrometeors at microwave frequencies are computed using the delta-Eddington approximation. Note that the cloud simulations described in section 8.5 are completely different: they are currently only applicable to visible and infrared radiances. RTTOV-SCATT is described by Bauer et al. (2006) and the cloud overlap is described in Geer et al. (2009a). Further information can also be found in the RTTOV v9, v10 and v11 Science and Validation Reports.

The RTTOV-SCATT code calls the core RTTOV for the clear air part but adds the scattering effects from water/ice in the profile. RTTOV-SCATT uses a two-independent column approximation, summarised by:

$$L_B^{Total} = (1-C)L_B^{Clear} + CL_B^{Rainy} \qquad (8)$$

Here, C is the effective cloud fraction in the vertical profile and L is radiance or brightness temperature (depending on the setting of **opts_scatt%lradiance**: it is recommended to set this flag to true so that the RT calculations are done in radiance rather than brightness temperature). The clear-air RTTOV is called from within RTTOV-SCATT and returns the radiance or brightness temperature of the clear sky column, L^{Clear}, and the profile of clear sky transmittances. RTTOV-SCATT then computes the cloudy or rainy radiance or brightness temperature, L^{Rainy}, using the clear sky transmittances provided by the core RTTOV, and lookup tables for Mie scattering properties. Finally, equation 8 is used to linearly combine the two independent columns, producing the total radiance or brightness temperature L^{Total}.

RTTOV-SCATT is called via the subroutine interface **rttov_scatt**, which is quite different from that for the core RTTOV, i.e. **rttov_direct**. The input profiles are the same as for the clear-sky RTTOV (e.g. **rttov_profile**; Table 12 and section 7.3) but additional information is required, principally hydrometeor profiles, supplied in **rttov_profile_cloud** and listed in Table 28. The **use_totalice** logical variable in **rttov_profile_cloud** should be set to false for separate ice and snow and to true for total ice. These two options are mutually exclusive.

The vertical level-layer discretisation used by RTTOV-SCATT is different to that used by the core RTTOV routines. Within the **rttov_profile_cloud** structure the constituent and hydrometeor amounts are given on 'full' pressure levels, and they apply to a domain bounded by 'half' pressure levels. Conventionally, the bottom half level is the surface (2m) pressure and the top half level is the top of the atmosphere. Full pressure levels are those supplied in **rttov_profile** (in **profiles(:)%p(:)**), but the half level pressures need to be supplied in **rttov_profile_cloud**. Figure 5 shows the arrangement of full and half levels. Note that in common with the treatment of cloud variables, for doing the radiative transfer in the cloudy column, RTTOV-SCATT treats profile variables in the **rttov_profile** structure as also representing full pressure level quantities (rather than layer quantities, as in the rest of RTTOV). Within the **rttov_profile_cloud** structure is the option to specify the rain and solid precipitation (sp) in units of [kg/kg] (the default) rather than as a flux [kg/(m^2)/s]. The input cloud concentrations must be the layer grid-box-average concentration (as opposed to the concentration within the cloudy fraction of each layer): the grid-box-average is commonly the value provided by GCMs.

Profile variable	Contents
nlevels	number of atmospheric levels, which should match that supplied in the other input profiles
use_totalice	logical flag to switch between using separate ice and snow, or total ice hydrometeor types.
mmr_snowrain	logical flag to set units for snow and rain: if true units are kg/kg (the default), if false units are (kg/(m^2)/s)
cfrac	Optional**:** if **opts_scatt%lusercfrac**=true., supply the effective cloud fraction, C, here. This is normally calculated internally in RTTOV-SCATT
ph(:)	nlevels + 1 of half-level pressures (hPa)
cc(:)	nlevels of cloud cover (0-1)
clw(:)	nlevels of cloud liquid water (kg/kg)
ciw(:)	nlevels of cloud ice water (kg/kg)
totalice(:)	nlevels of total ice (kg/kg)
rain(:)	nlevels of rain flux (units depend on *mmr_snowrain*)
sp(:)	nlevels of solid precipitation flux (units depend on *mmr_snowrain*)

*Table 28. RTTOV-SCATT profile variables for **rttov_profile_cloud***

An example of an RTTOV-SCATT forward model call is provided in **src/test/example_rttovscatt_fwd.F90**. This reads atmospheric profiles from a data file and writes the simulated brightness temperatures to an output file (this should be called using **rttov_test/run_example_rttovscatt_fwd.sh**, see section 5.3).

Another example of calling RTTOV-SCATT including a Jacobian calculation is provided in **src/mw_scatt/example_rttovscatt.F90**. The test programs **rttovscatt_test.F90** (top level, driven by **rttov_test/test_rttovscatt.sh**) and **rttovscatt_test_one.F90** both in **src/mw_scatt/** contain further examples of how to use the tangent-linear, adjoint and K functionality.

A limited number RTTOV options are available in RTTOV-SCATT via the **rttov_options_scatt** type described in Annex O. This also contains the **lusercfrac** variable which was an argument of the **rttov_scatt** subroutine in RTTOV v10. If **opts_scatt%lusercfrac** = .true., you can supply your own effective cloud fraction (see Geer et al., 2009b). By default this flag is false and the effective cloud fraction is calculated internally in RTTOV-SCATT. The optional diagnostic output array **cfrac** is available from the direct call, should you want to look at the value computed internally.

If the effective cloud fraction is smaller than a certain threshold, then the cloudy radiance is ignored and only the clear-sky radiance is returned. Prior to RTTOV v12.3 this threshold was hard-coded at 0.05, but in v12.3 it can be specified in **opts_scatt%cc_threshold** (the default remains 0.05).

Support for all-sky dynamic emissivity retrievals was added in v12.2, following the method of Baordo and Geer (2016). An optional output structure of the direct call to RTTOV-SCATT, the **rttov_scatt_emis_retrieval_type** contains the relevant parameters for the emissivity retrieval (see Annex O); as with other interface structures there are helper routines to allocate and deallocate it. To calculate these additional outputs, **opts_scatt%lradiance** must be set to true. The function **rttov_scatt_emis_retrieval** is then available to retrieve the surface emissivity (see Annex N).

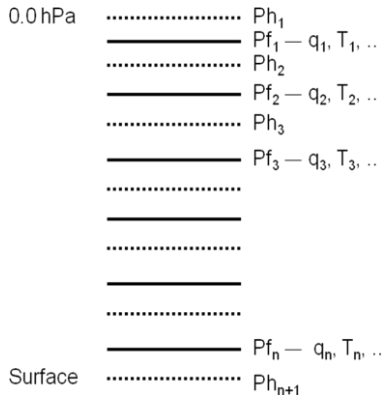

Figure 5. RTTOV-SCATT full and half levels, showing half and full level pressure (Ph,Pf) and examples of the variables specified on full levels (e.g. q, T, but also cloud and hydrometeors)

RTTOV-SCATT optical properties

RTTOV-SCATT relies on both clear-sky coefficient files (e.g. **rtcoef_noaa_15_amsua.dat**) and precomputed tables of scattering parameters (e.g. **mietable_noaa_amsua.dat**). These files are referred to as "Mietables", but they also now include optical properties for non-spherical particles. Due to their large size the Mie files for microwave sensors are not supplied in the package, but are provided on the RTTOV web site for download. The executable **rttov_ascii2bin_scattcoef.exe** may be used to convert ASCII Mie tables to binary format for faster performance. This is described in Annex A.

You may also create your own Mie coefficient files (or re-generate those supplied on the RTTOV website) using the UNIX shell script **src/mw_scatt_coef/mie_table_generation.ksh**. This script may need editing, for example to point to the location of your RTTOV binaries. The Mie table generation is based around an input file **channels.dat**, examples of which may be found in the same directory. These define the parameters for the calculations and the instruments and channels for which to generate coefficients. The existing **channels.dat** files provide useful templates. See the associated **readme.txt** in the **src/mw_scatt_coef/** directory for full details. Modifying the Mie tables may be useful in some cases but is not generally recommended.

8.8. Simulation of hyperspectral IR sounder radiances using PC-RTTOV

A principal component (PC) based version of RTTOV is available for the simulation of the full spectrum of hyperspectral IR sounders as PCs (Matricardi, 2010). The PC-based model uses polychromatic RTTOV radiances to predict the principal component scores using a linear regression scheme. To invoke the PC calculations in RTTOV v12 a logical flag **opts%rt_ir%pc%addpc** is set to true. An example of calling PC-RTTOV is given in **src/test/example_pc_fwd.F90**: this can be used as a model for your own code.

Coefficients

The PC-RTTOV coefficient filenames begin "*pccoef_*" and can be downloaded from the RTTOV web site. These must be used alongside the RTTOV optical depth (*rtcoef*) coefficient files with which they were trained. PC-RTTOV is trained used v9 predictor 101 level coefficients. At the time of release the PC coefficients were trained with the same optical depth coefficients used to train the RTTOV v11 PC coefficients. These optical depth coefficient files have "*pcrttov_compat*" in the filename. If an incompatible optical depth coefficient file is used an error will result. The PC coefficient file is read in at the same time as the optical depth coefficient file in the call to **rttov_read_coefs**.

The PC-compatible optical depth coefficient files allow for the variation of O_3, CO_2, N_2O, CO and CH_4. RTTOV v12.2 introduced new PC coefficient files which allow all of these gases to vary (optionally) in the same was as for standard (non-PC) RTTOV. PC coefficients which support all of these trace gas species have "*trace*" in the filename. Older PC-RTTOV coefficients were only trained with variable ozone and when using those PC coefficient files any CO_2, N_2O, CO and CH_4 values specified in the RTTOV input state vector are ignored and instead the fixed profiles stored in the PC coefficient file are used. This also means that the Jacobians calculated by the RTTOV K model are zero for these trace gas species. PC-RTTOV has different regression limits for gases than the RTTOV optical depth coefficients, but these are treated in the way as for standard RTTOV (see section 7.3). The PC regression limits can be found on the RTTOV coefficients download page:
https://www.nwpsaf.eu/site/software/rttov/download/coefficients/coefficient-download/#Reference_profiles_and_regression_limits

RTTOV v12.2 also introduced new PC coefficient files which support aerosol simulations using the OPAC aerosol components: these have "*aer*" in the filename. A suitable OPAC aerosol property file should be read in when the RTTOV optical depth and PC coefficient files are read by the call to **rttov_read_coefs**. The aerosol simulations are carried out as described in section 8.6, but note that you must use the Chou-scaling parameterisation (**opts%rt_ir%ir_scatt_model** = 2). The PC coefficients are trained using climatological combinations of the OPAC aerosol components: this includes only indices 1-10 from the RTTOV aerosol property files (see Table 27). RTTOV will ignore any input aerosol concentrations for the volcanic ash and Asian dust species. The minimum/maximum aerosol concentrations (regression limits) used in training the PC profiles can be found on the RTTOV coefficients download page. Note that in some layers for some species there was no variability in the aerosol concentrations (this includes the sulphates component in all layers). RTTOV automatically uses the regression limit values for those layers.

For layers where the aerosols varied in the training RTTOV will clip the input aerosol profiles to the limits if the **apply_reg_limits** option is true in a similar way as for gases (see section 7.3). If the aerosol regression limits were exceeded in one of these layers the **radiance%quality(:)** flags have the **qflag_pc_aer_reg_limits** bit set for the PC predictor channels corresponding to the relevant profile (see section 7.8).

The PC coefficients are trained with the RTTOV atmospheric refraction option turned on. Therefore refraction is accounted for regardless of the setting of **opts%rt_all%addrefrac**. It should be noted that a number of the options available in RTTOV were not enabled in the PC-RTTOV training and as such these options should not be used with PC-RTTOV. These include the inclusion of the solar term in short-wave channels and the Lambertian surface option. You can call the **rttov_user_options_checkinput** subroutine (Annex N) to check that your options are compatible with PC-RTTOV.

PC coefficient files are available which have been trained over all surface types. These coefficient files have "*landsea*" in the filename. It is strongly recommended to set **calcemis(:)** to true for sea profiles: PC-RTTOV is trained using its own sea surface emissivity model and this is used automatically for sea profiles when **calcemis(:)** is true. It is also recommended to use the UW IR land surface emissivity atlas (see section 7.5) for land and sea-ice surface types as this was used to train the PC-RTTOV coefficients. However as the new PC-RTTOV training encompasses a wide range of surface emissivities, the use of alternative physically realistic sources for surface emissivity should be acceptable. For

this reason RTTOV v12 carries out no checks on how surface emissivity is specified for PC-RTTOV: you must ensure that you do not use any sea-only PC coefficients which have just "*sea*" in the filename over land surfaces.

PC-RTTOV has been extended to carry out simulations that can include a NLTE correction (see section 8.10) to the TOA radiances (Matricardi *et al* 2016). This option is available in PC coefficient files with "*nlte*" in the filename and these files can be used to optionally include NLTE effects in the spectral region at 4.3 μm for solar zenith angles between 0 and 90 degrees for the entire viewing geometry of IASI.

As new PC-RTTOV coefficient files are created enabling more options in the simulations (for example, NLTE, additional trace gases, aerosols), the most recent coefficient file for a sensor can replace previous PC coefficient files.

The PC-RTTOV coefficient files contain the regression coefficients used to predict the PC scores and up to 400 (IASI, AIRS) or 600 (IASI-NG) eigenvectors to reconstruct radiances from the PC scores: regression coefficients are stored for the predictor sets listed in Table 29. The files contain the regression coefficients for the PC-RTTOV sea surface emissivity model (activated by setting **calcemis(:)** to true over sea as noted above). In addition they store the fixed profiles of CO_2, N_2O, CO and CH_4 used in the LBLRTM computations on which PC-RTTOV is trained for older files, or the minimum and maximum gas and (where supported) aerosol regression limits.

Coefficient file	opts%rt_ir%pc%ipcbnd	opts%rt_ir%pc%ipcreg	Max npcscores
IASI – global + trace gases + NLTE	1 => full spectrum (all channels)	1, 2, 3 or 4 => 300, 400, 500 or 600 predictors respectively	400
IASI – global + trace gases + aerosols	1 => full spectrum (all channels)	1, 2, 3 or 4 => 300, 400, 500 or 600 predictors respectively	400
AIRS – global	1 => full spectrum (all channels)	1, 2 or 3 => 200, 300 or 400 predictors respectively	400
IASI-NG – sea-only	1 => full spectrum (all channels)	1, 2, 3 or 4 => 300, 400, 500 or 600 predictors respectively	600

Table 29. Available options for band and predictor sets for PC coefficient files available at the time of the RTTOV v12.3 release.

Predictor channel sets

In order to call PC-RTTOV a specific set of channels must be specified in the **chanprof(:)%chan** array for each profile being simulated. The simulated RTTOV radiances for this set of channels comprise the predictors in the PC-RTTOV regression. The size of the predictor channel set determines the number of channels being simulated per profile. The predictor set is selected in **opts%rt_ir%pc%ipcreg**. Table 29 gives the valid values for the PC-RTTOV coefficients available at the time of the RTTOV v12.3 release. By choosing a larger predictor set one trades reduced computational efficiency for increased accuracy.

The specific channel list for a given predictor set can be obtained directly from the PC coefficient structure in **coefs%coef_pccomp%pcreg(i,j)%predictindex(:)** for band **i** (see below - usually 1) and predictor set **j**. An alternative way of obtaining the channel list is via the subroutine **rttov_get_pc_predictindex** (Annex H): an example of this can be seen in **src/test/example_pc_fwd.F90**. RTTOV will report an error if the input channel list (in **chanprof(:)%chan**) does not match the predictor channel list.

NB The predictor channel sets are unique to each new PC-RTTOV coefficient file for a given instrument so that, for example, the new NLTE-compatible IASI coefficients require a different predictor channel set to the RTTOV v11 IASI PC-RTTOV coefficients.

The variable **opts%rt_ir%pc%ipcbnd** provides a choice of carrying out calculations for limited spectral bands, but currently the PC coefficient files available on the website contain information for the whole spectrum only (band 1) and hence this variable must always be set to 1.

The number of simulated principal components can vary from 1 to 400 (IASI, AIRS) or 1 to 600 (IASI-NG). Prior to RTTOV v12.3 this was defined when calling the **rttov_alloc_pccomp** subroutine (see Annex D) to allocate the **pccomp** output structure (see below). Note that the value supplied here is the number of principal components to be calculated for all profiles in the call to RTTOV. Again, **example_pc_fwd.F90** provides a simple example. A typical choice is 200

principal components and 500 predictors for IASI and 100 principal components and 300 predictors for AIRS. Again, the number of simulated PC scores can be increased in order to improve accuracy at cost in computational efficiency.

RTTOV v12.3 provides a new option **opts%rt_ir%pc%npcscores** which can be used to specify the number of PC scores to use in the simulations. If this value is larger than zero then it determines the number of PC scores for the simulation. In this case you must ensure that the **pccomp** structure was allocated to be large enough for the number of PC scores specified via the option. This allows you to specify the size of **pccomp** for the maximum number of profiles you wish to simulate per call to RTTOV, but then to simulate fewer profiles in a given call (which may happen, for example, in the last batch of profiles from a larger set): previously RTTOV would not allow this. The default behaviour of v12.3 replicates v12.2 and earlier (**npcscores** defaults to -1 so that the number of PC scores is determined by the size of **pccomp**).

PC-RTTOV outputs

The computed PC scores are stored in the **rttov_pccomp** structure (see Annex O) in the **pcscores(:)** array. It is possible to reconstruct radiances from the PC scores by setting **opts%rt_ir%pc%addradrec** to true. In this case the total number of reconstructed radiances for all profiles (or the maximum number of profiles being passed per call to RTTOV) must be passed in the call to **rttov_alloc_pccomp** (Annex D) and the reconstructed channel list **channels_rec(:)** must be supplied to **rttov_direct** (or to the TL, AD or K model; Annexes I, J, K, L). The reconstructed radiances are also stored in the **pccomp** structure in the **total_pccomp(:)** array, and the corresponding brightness temperatures are in **bt_pccomp(:)**. The **channels_rec(:)** input argument to **rttov_direct** is mandatory if **opts%rt_ir%pc%addradrec** is true.

It is technically possible to run PC-RTTOV using coefficient files containing a subset of channels (created using **rttov_conv_coef.exe**, see Annex A) or to read in a subset of channels in the call to **rttov_read_coefs** (see Annex C). However it is important to note that the subset of extracted channels must include the set of predictor channels being used and also any channels for which reconstructed radiances are required. In addition, the channels specified in **chanprof(:)%chan** and **channels_rec(:)** are always indexed starting from 1 into the set of channels read into RTTOV. This means you must keep track of the remapped channel numbers. For this reason it is not advisable to run PC-RTTOV with channel subsets unless you are confident in what you are doing.

8.9. Simulation of hyperspectral IR sounder radiances using HTFRTC

HTFRTC is a Principal Components-based fast radiative transfer model (Havemann *et al*, 2018). The HTFRTC model was introduced into RTTOV in v12.1 as an external library. In RTTOV v12.2 the implementation was improved significantly to be easier to compile, and more flexible and efficient to run. In RTTOV v12.3 the capability was further developed to enable all RTTOV optional variable gases, to enable all RTTOV gas units options, to enable input/output of surface emissivities, to compute overcast radiances and simple cloudy radiances (equivalent to those computed by RTTOV - see section 8.3), and to support some additional RTTOV options. The implementation currently enables direct and K-model clear-sky simulations over all surface types for hyperspectral IR sounders.

The core HTFRTC code is now included within the RTTOV package and is compiled with RTTOV which means that HTFRTC does not have to be downloaded and compiled separately. The HTFRTC coefficient files are in NetCDF format and so RTTOV must be compiled against the NetCDF v4 library (see section 5.2).

HTFRTC simulations are enabled by setting the **opts%htfrtc_opts%htfrtc** option to true. When running HTFRTC simulations the usual RTTOV *rtcoef* coefficient files are not required. HTFRTC simulations require two input files: one is a static coefficient file (*htfrtc_coef_static.nc*) that is required for all HTFRTC simulations, and the other is specific to the sensor being simulated (e.g. *htfrtc_coef_sensor_metop_iasi.nc*). Different static files are required for RTTOV v12.2 and v12.3. HTFRTC coefficients are available on the RTTOV coefficient downloads page (see section 5.1). Both coefficient files are read by calling the **rttov_read_coefs_htfrtc** subroutine which is used instead of **rttov_read_coefs** for HTFRTC simulations (see Annex C). The coefficient data are read into the **rttov_coefs** structure just as for other RTTOV simulations. HTFRTC computes reconstructed radiances (if requested) for *all* channels read from the coefficient file. By default all instrument channels are read in, but you can specify a subset of channels via the optional **channels_rec** argument to **rttov_read_coefs_htfrtc**.

HTFRTC makes use of only a limited number of RTTOV options and profile variables. The number of PC scores required is specified in **opts%htftrtc_opts%n_pc_in**. The maximum number allowed by the HTFRTC input files is 300. If you want reconstructed radiances these are enabled by setting **opts%htftrtc_opts%reconstruct** to true. As noted above radiances are calculated for all channels read from the coefficients file (i.e. the **channels_rec** argument to **rttov_direct** and **rttov_k** is ignored). In RTTOV v12.3, all RTTOV optional trace gases are supported and ozone is now optional. If a gas is not supplied, the RTTOV background profile is used. For any gas which is supplied you must set the corresponding "gas_data" option to true, just as for RTTOV: for example, if you supply an ozone profile you must set **opts%rt_ir%ozone_data** option to true. The only other RTTOV options used by HTFRTC are **opts%rt_all%use_q2m**, **opts%rt_all%switchrad, opts%rt_all%do_lambertian** and **opts%rt_all%plane_parallel**: all other options are ignored. In the case of the **do_lambertian** option, HTFRTC uses a fixed angle for the downwelling radiance and makes use of the **specularity** skin profile variable in the same way as RTTOV.

In RTTOV v12.3, HTFRTC can optionally compute simple cloudy radiances for a single grey cloud at a given pressure using the profile **ctp** and **cfraction** variables. This is enabled by setting **opts%htfrtc_opts%simple_cloud** to true. Similarly, the new capability to output overcast radiances for each layer (equivalent to the RTTOV overcast radiance output) is enabled by setting **opts%htfrtc_opts%overcast** to true. These options can be used together or alone, but note that they increase the run-time. If you wish to generate these outputs, you must also set the options to true before you call **rttov_alloc_pccomp** to ensure the relevant output arrays are allocated (though you can subsequently turn them off for standard "clear-sky" simulations if required) - see below.

The profile variables used by HTFRTC are listed in Table 30. Profiles can be input on any set of pressure levels: profiles do not need to have the same pressure levels, but all profiles must be on the same number of levels (as for RTTOV).

Profile structure variables	**Notes**
gas_units	Mandatory - these affect all gas input units just as for RTTOV.
p(:), t(:), q(:)	Mandatory
o3(:), co2(:), n2o(:), co(:), so2(:)	Optional - set corresponding **opts%rt_ir%*_data** to true
s2m%p, s2m%t	Mandatory
s2m%q	Used if **opts%rt_all%use_q2m** is true
s2m%u, s2m%v	Used by emissivity calculation, mandatory for sea profiles
skin%t	Mandatory
skin%surftype	Mandatory
skin%specularity	Used if **opts%rt_all%do_lambertian** is true.
zenangle	Mandatory
elevation	Mandatory
ctp, cfraction	Used if **opts%htfrtc_opts%simple_cloud** is true.

Table 30: profile variables used by HTFRTC.

A number of mandatory arguments to **rttov_direct** and **rttov_k** are not used by HTFRTC. These arguments must be passed into the subroutine calls, but the structures do not need to be allocated. In particular, the **chanprof(:)** argument is not used: HTFRTC computes PC scores (and optionally reconstructed radiances) for all input profiles. In order to prevent compilation issues you can allocate a **chanprof(:)** array of size 0 and pass this into RTTOV.

The HTFRTC outputs are contained in the **pccomp** structure (see Annex O). When allocating this structure (by calling **rttov_alloc_pccomp** – see Annex D) you must specify the number of PC scores (as for PC-RTTOV) and, if required, the number of reconstructed radiances (which is the same as the number of channels read from the coefficients, as described above). HTFRTC does not write to the **radiance** or **transmission** data structures so there is no need to allocate these structures as for standard RTTOV calls. The same applies to the equivalent K model variables for **rttov_k** calls. In RTTOV v12.3 the **pccomp** structure now contains a number of additional output arrays for HTFRTC which are intended to mirror the **radiance** structure outputs. In particular there are "clear" and "total" outputs for PC scores, radiances and BTs which contain the clear-sky and simple-cloudy simulated outputs if the **simple_cloud** option is enabled or otherwise are identical. In addition the 100% cloudy PC scores and radiances are output for the simple cloud scheme, and overcast PC scores and radiances if the **overcast** option is true. The **pcscores** output remains for backward compatibility, and this is always equal to **clear_pcscores**. See Annex O for the full description of the **pccomp** structure.

As of RTTOV v12.3 HTFRTC provides more flexibility for surface emissivities. HTFRTC can still provide surface emissivities using the PC-RTTOV model for sea surfaces and the fixed RTTOV values for land and sea-ice (0.98 and

0.99 respectively). The **emissivity** (and **emissivity_k**) inputs are now optional: if supplied then they will return the emissivities used/calculated by HTFRTC in **emissivity(:)%emis_out** and the emissivity Jacobians calculated by HTFRTC in **emissivity_k(:)%emis_in**. You can also provide input emissivities to HTFRTC by passing the **calcemis** argument and providing input emissivities in **emissivity(:)%emis_in**. This works in a similar manner to RTTOV, but it is important to note that the input/output emissivities for HTFRTC are specified for the reconstructed radiance channels. This means that these inputs/outputs are only available if **opts%htftrtc_opts%reconstruct** is true and the size of the **calcemis**, **emissivity** and **emissivity_k** arrays is the number of reconstructed channels multiplied by the number of profiles. The **emissivity** argument is mandatory if **calcemis** is supplied, and **calcemis** must be false for all reconstructed radiance channels corresponding to a given profile for the input emissivities to be used - otherwise, if any member of **calcemis** for a profile is true, then HTFRTC computes the emissivities internally for that profile. Input emissivities are linearly interpolated internally from the reconstructed radiance channel wavenumbers to the HTFRTC centroid frequencies for use the the RT calculation.

It is also possible to use the RTTOV emissivity atlases with HTFRTC: the atlas interface is the same as if calling it for standard RTTOV simulations, including optionally initialising for use with a single instrument. The only difference is that the **chanprof** argument to the **rttov_get_emis** subroutine is not used because emissivities are returned for all reconstructed radiance channels in the initialised HTFRTC coefficients in the **rttov_coefs** structure for each profile.

When running Jacobian simulations using **rttov_k** you should initialise the **pccomp_k** structure to zero (for example using the **rttov_init_pccomp** subroutine – see Annex D). HTFRTC uses an input perturbation of 1. in every channel. The Jacobians are calculated in terms of PC score if **opts%htftrtc_opts%reconstruct** is false. Otherwise they are calculated in terms of radiance if **opts%rt_all%switchrad** is false or in terms of BT if **opts%rt_all%switchrad** is true (this is similar to PC-RTTOV).

NB It is important to note that you must supply both **profiles_k_pc** and **profiles_k_rec** arguments to **rttov_k** when computing Jacobians for reconstructed radiances as HTFRTC *always* calculated the Jacobians for PC scores.

HTFRTC may be called via the RTTOV parallel interface in exactly the same way as for RTTOV.

Example code demonstrating a forward model HTFRTC call can be seen in **src/test/example_htfrtc_fwd.F90** and this also includes code using the emissivity atlas. This is similar to the other example code, and can be run from within the **rttov_test/** directory using the **run_example_htfrtc_fwd.F90** script in the same way that the other example scripts are run.

8.10. Inclusion of non-local thermodynamic equilibrium effects

RTTOV v12 is able to estimate NLTE effects above altitudes of ~40 km in the CO_2 v3 band (around 4.3 μm). Here, local thermodynamic equilibrium breaks down due to the absorbtion of the strong solar radiation field. The NLTE model has been updated since RTTOV v11 and at the time of release coefficients for the new model are only available for IASI. NLTE coefficients for other hyperspectral sounders can be requested via the NWP SAF helpdesk.

To invoke the NLTE correction it is necessary to:

- Use an optical depth (*rtcoef*) coefficient file that contains the NLTE coefficients for the regression.
- Set **opts%rt_ir%do_nlte_correction** = .true.

As noted in section 8.8, PC-RTTOV coefficients are now available which can include the NLTE correction. To invoke the NLTE correction for PC-RTTOV it is necessary to:

- Use an optical depth (*rtcoef*) coefficient file that contains the NLTE coefficients for the regression and is compatible with PC-RTTOV (see section 8.8). This allows the addition of a NLTE correction to the PC-RTTOV predictors.
- Use a PC-RTTOV coefficient file that has been trained including NLTE effects.
- Set **opts%rt_ir%do_nlte_correction** = .true.

For both classical RTTOV and PC-RTTOV the correction is valid (and is applied to channels) between 2200 cm^{-1} – 2400 cm^{-1} which corresponds to:

- IASI channels 6221 – 7021 (801 channels)

The radiance correction scheme is documented in Matricardi *et al* (2016) and consists of eight predictors. These predictors consist of various combinations of the solar zenith angle, θ^{sol} the sensor zenith angle, θ^{sat} and the average kinetic temperature in two broad atmospheric layers above ~51 hPa (i.e. the average temperature between 0.005 hPa and ~0.2 hPa, *T1* and the average temperature between ~0.3 hPa and ~51 hPa, *T2*). The predictors *p* are shown in Table 31.

Predictor number	Predictor
1	constant
2	$\cos(\theta^{sol})$
3	$(\cos(\theta^{sol}))^{0.5}$
4	$\cos(\theta^{sol})\ \sec(\theta^{sat})$
5	$(\cos(\theta^{sol})\ \sec(\theta^{sat}))^2$
6	$\cos(\theta^{sol})\ T1$
7	$\cos(\theta^{sol})\ T2$
8	$\sec(\theta^{sat})\ T1$
9	$\sec(\theta^{sat})\ T2$

Table 31: the predictors used in the PC-RTTOV NLTE algorithm

The radiance correction, ΔR_{ch}^{NLTE}, is added to the LTE TOA radiance to give the NLTE TOA radiance, thus,

$$R_{ch}^{NLTE} = R_{ch}^{LTE} + \Delta R_{ch}^{NLTE}, \qquad (9)$$

where ΔR_{ch}^{NLTE} is written as:

$$\Delta R_{ch}^{NLTE} = \sum_{j=1}^{9} c_{ch,j}\ p_j \qquad (10)$$

Here $c_{ch,j}$ are the regression coefficients.

You are responsible for ensuring that input values for the solar zenith angle are in range (i.e. >0° and < 90°). For any particular profile, RTTOV will not calculate the correction if the supplied solar zenith angle lies out of range and will extrapolate the correction if the supplied satellite zenith angle is out of range (as long as the solar zenith angle is valid).

Note that the NLTE correction is distinct from the solar simulation capability and as such it is *not* necessary to set **opts%rt_ir%addsolar** to true for the NLTE correction to be applied. The NLTE correction may be used in conjunction with IR scattering calculations, but you should be aware that the NLTE correction is added to the final LTE cloudy/aerosol-affected radiances and so the NLTE radiation does not interact with the scattering particles.

A more detailed discussion of the science and the impact of including the new NLTE correction is available in the RTTOV v 12 Science and Validation Report.

8.11. Option to treat surface as a Lambertian reflector

The reflection of downward radiation over snow or multi-year sea-ice is better characterised by assuming a Lambertian approximation rather than the specular reflection which is the default in RTTOV. This is particularly relevant for microwave sounders as sea-ice and snow have high reflectances at these frequencies. True Lambertian reflection requires an integral over a range of angles to cover the hemisphere which would prove difficult (and costly) in the RTTOV framework. Fortunately Matzler (2005) has developed an approximation as a function of optical depth providing a fixed angle of ~55º to be used for the downward radiation. The option of Lambertian reflection is now possible in the MW and the IR. To invoke Lambertian reflection set the **opts%rt_all%do_lambertian** flag to true. By default it is false. Significant changes (up to 10K) will be seen in microwave window channels with the largest differences for nadir views. More details are provided in the RTTOV v11 Science and Validation Report.

RTTOV v12.3 includes a new option (**opts%rt_all%lambertian_fixed_angle**) to enable use of the parameterisation (Guedj *et al*, 2010) of the downwelling angle in terms of the total atmospheric optical depth: if false (the default), the fixed angle of 55º is used, otherwise if true the parameterisation for the angle is used.

In addition RTTOV v12.3 introduces a new profile variable: **profiles(:)%skin%specularity**. Most surfaces are not truly Lambertian or truly specular, but lie somewhere between the two. This variable is active when the **do_lambertian** option is true, and specifies the weighting applied when linearly combining the downwelling specular and Lambertian radiances. The valid range of values is 0 (fully Lambertian) to 1 (fully specular - the same as if **do_lambertian** is false).

NB For profiles/channels where an internal sea surface emissivity model is being used (i.e. FASTEM or TESSEM2 in the MW and ISEM or IREMIS in the IR) the Lambertian option is not valid and so is not applied for these channels. When activated the Lambertian option is applied for sea surfaces where **calcemis(:)** is false and for land and sea-ice surfaces regardless of **calcemis(:)**.

8.12. Zeeman effect for SSMIS and AMSU-A

For microwave sensors that have high peaking weighting functions in the mesosphere such as SSMIS, channels close to lines of molecular oxygen may be significantly affected by the redistribution of line intensity through Zeeman splitting as described in the RTTOV v10 Science and Validation Report. The absorption for the affected channels will depend on the strength and orientation of the magnetic field. You must specify two input variables for the geomagnetic field in the **rttov_profiles** structure, these being the magnitude, **Be**, of the field and the cosine, **cosbk**, of the angle between the field vector and the viewing path considered. For SSMIS, values will be available with the satellite data stream, and will therefore already match the geographical location and orientation of the viewing path. For AMSU-A, this is not the case, but the values may be obtained from a pre-computed look-up table. For instance, the **rttov_zutility** module provided in the **src/other/** directory may be used to provide values (see Annex N). For a normal run where the Zeeman effect is not computed the variables can be set to any value, including zero, but if the Zeeman effect is to be calculated, **Be** should lie in the range 0.2-0.7 gauss as this covers the range of values over which the Zeeman coefficients were trained. In particular, **Be** must not be set to zero when calling RTTOV with a Zeeman coefficient file: if the Zeeman effect is not important for an application a non-Zeeman coefficient file should be used instead.

A 'Zeeman' coefficient file will have the Zeeman flag set to unity in the 'Fast Model Variables' section and "*zeeman*" in the filename. To include the Zeeman effect for a given sensor, you must run RTTOV v12 with a Zeeman coefficient file.

As with all RTTOV simulations your input profile pressure levels should reach sufficiently high that they span the weighting functions of the channels being simulated (i.e. sufficiently high that there is insignificant absorption above the top-most input pressure level). If this condition is not met there can be significant errors resulting from the treatment of emission from the region of atmosphere above the top input level. After interpolating the calculated optical depth profile onto the input pressure levels, RTTOV by default sets the optical depth of the top-most level to zero. This behaviour is intended to mitigate the case where there is significant absorption above the top level: in this case the top layer is effectively stretched to reach the space boundary and the effects of emission and absorption from the region of atmosphere above the top level are included in the integration of the radiative transfer equation. However without an accurate representation of the temperature of the atmosphere above the top level the emission term will be in error to some degree. Note that this feature has negligible impact if the top input profile level is sufficiently high because in that case the interpolated optical depth at the top level will be close to zero anyway. This feature is enabled by setting **opts%interpolation%spacetop** to true which is the case by default. If this flag is set to false the interpolated optical depth at the top level is not modified: when the radiative transfer equation is integrated, the absorption due to the atmosphere above the top input level is accounted for, but emission from this region of atmosphere is omitted entirely. In general there is no need or benefit to set this flag to false. Note that the **spacetop** flag affects all simulations, not only Zeeman-affected simulations.

For SSMIS channels 19-22 which are affected by Zeeman splitting, the brightness temperature in channel 20 may be altered by as much as 10 K – the change in column absorption will shift the channel weighting function, but the effect of this will actually depend on the temperature profile. When you run RTTOV with a non-Zeeman coefficient file, the mixed gas prediction scheme will be based on the usual set of predictors. However, when a Zeeman coefficient file is used, the mixed gas scheme will incorporate additional predictors used for the high peaking channels. In the optical depth calculation for channels 1-18 and 23-24 (non-Zeeman), contributions from the additional predictors will be nullified by zero coefficients. In contrast, for channels 19-22 (Zeeman), it is only the contributions from the new predictors that contribute.

For AMSU-A, only channel 14 is affected. This channel, while dominated by oxygen absorption, sounds lower down in the atmosphere than the Zeeman channels of SSMIS, and it is also located further from the oxygen line centres. The

impact is therefore much smaller (~0.5K). If you run with a non-Zeeman coefficient file, all channels will use the usual set of mixed gas predictors and the Zeeman effect will not be represented in channel 14. If a Zeeman coefficient file is used, then a small set of additional predictors will be included. These will contribute for channel 14 but will be nullified for the other channels by zero coefficients.

8.13. Simulation of SSU radiances

For SSU, which uses pressure modulated gas cells to define the channels, RTTOV v12 allows you to take some account of inadvertent cell pressure changes that may have occurred over the lifetime of the instrument.

SSU coefficients were updated for RTTOV v10 using more recent molecular spectroscopy (LBLRTMv12.0), more variable gases (now O_3 as well as H_2O, CO_2) and more stratospheric levels (now 51). RTTOV v11 provided an additional set of coefficients with the label '*pmcshift*' in the file name which are based on the same nominal set of cell pressures, and differ only in having a new 'PRESSURE_MODULATED_CELL' section.

When simulating SSU with these pressure modulated cell (PMC) shift coefficients, you must now provide a preferred set of gas cell pressures (in hPa), one for each channel. These may be different from those in the nominal set and should be assigned to the channel array **coefs%coef%pmc_ppmc**. This should be done after the coefficient file has been read using the **rttov_read_coefs** subroutine. It is also mandatory to supply an input CO_2 profile when using the PMC shift coefficients and this also feeds into the cell pressure scheme.

Note that the SSU coefficient files do not support the new IREMIS sea surface emissivity model so you must set **opts%rt_ir%ir_sea_emis_model** to 1 to select ISEM otherwise RTTOV will fail with an error. In practice this makes no difference to the simulations as SSU is not sensitive to the surface.

8.14. Simulations with variable SO_2

RTTOV v12 introduces new optical depth coefficient files which allow for variable SO_2. SO_2 is treated in the same way as other gases. You should set the **opts%rt_ir%so2_data** flag to true to indicate you are supplying SO_2 profiles and the input profile data are specified in **profiles(:)%so2(:)**. The SO_2 optical depth prediction has been trained using a selection of profiles which cover both "clean" (low-SO_2) and "volcanic" (high-SO_2) atmospheres. If you do not supply an SO_2 profile with an SO_2-enabled coefficient file then RTTOV uses a "clean" background SO_2 profile.

The SO_2-enabled coefficients show slightly larger errors (as measured by comparisons to the line-by-line model used to train RTTOV) in the spectral regions where SO_2 is active for "clean" SO_2 profiles. For this reason coefficients are available for all trace gases with and without variable SO_2. SO_2-enabled coefficients have been generated for hyperspectral sounders and are identified by "*_so2*" in the filename: they can be downloaded from the RTTOV coefficients web page. More detailed information about the new coefficients is available in the RTTOV v12 Science and Validation Report.

9. Limitations of RTTOV v12

There are a number of scientific limitations of RTTOV v12 you should be aware of. The main ones are listed here:

- RTTOV v12 only simulates top of atmosphere radiances from a nadir or off-nadir view which intersects with the Earth's surface (i.e. no limb paths or upward viewing paths).
- RTTOV v12 only allows for water vapour, ozone, carbon dioxide, nitrous oxide, methane, carbon monoxide and sulphur dioxide to be variable gases with all others included in the mixed gases transmittance calculation.
- RTTOV v12 can only simulate radiances for instruments for which a coefficient file has been generated. The instruments currently supported are listed in Table 3. Only sensors with channels at wavelengths greater than 0.4 microns can be simulated with RTTOV v12.
- The accuracy of simulations for very broad channels (e.g. SEVIRI channel 4 at 3.9 microns) is poor with significant biases noted (~1-2K) (see e.g. Brunel and Turner, 2003). This is the case for all versions of RTTOV. A work around is to use Planck weighted coefficient files (which are now standard for all sensors where this is a problem) resulting in much lower biases. Whether coefficients are Planck-weighted can be determined by examining the PLANCK_WEIGHTED section in the coefficient file (if it is not present, there are no Planck-weighted channels).
- PC-RTTOV computations are limited by the configuration of the coefficient training. More information is given in section 8.8.
- HTFRTC: TL and AD models are not implemented.

10. Reporting and known bugs for RTTOV v12

Bug reports or other comments/feedback can be submitted via the NWP SAF helpdesk: http://nwpsaf.eu/site/helpdesk/. Select RTTOV as the "department" and include the following information:

- RTTOV version number (i.e. 12.3)
- Platform and operating system you are running the code on (e.g. Linux PC, IBM, Cray)
- Compiler used (e.g. *gfortran*, *ifort*, *pgf90*, etc) and compilation flags
- Classification of report as: serious, cosmetic or improvement
- Report of problem including any input /output files the SAF can use to reproduce the problem

Once the problem has been analysed it will be posted on the RTTOV web site (http://nwpsaf.eu/site/software/rttov/rttov-v12/code-updates/) with a description of the fix if appropriate. There is also a RTTOV v12 email list where major bugs are announced which you can subscribe to by contacting the help desk. When you register to download RTTOV you will be automatically included on this list unless you indicate otherwise.

The known issues in RTTOV v12.3 are given below. Any further problems and corrections will be provided via the RTTOV v12 web page as they become known:
http://nwpsaf.eu/site/software/rttov/rttov-v12/code-updates/

i. On the NEC architecture for RTTOV v10 some subroutines could not be compiled with optimisations. See the **nec-meteofrance** file in the **build/arch/** directory. The internal RTTOV interpolator was found to run relatively slowly on the NEC when compared to other platforms. Due to unavailability of a NEC platform RTTOV v12 has not been tested on this architecture, but similar issues may be expected.

ii. RTTOV v12 compiled with NAG Fortran v6.0 (build 1032) fails when running the test suite. This appears to be a compiler bug related to reading nested structures in a namelist. Since the example programs and tests outside of the **rttov_test.pl** test harness run OK it is believed that RTTOV is compatible with this compiler, but thorough testing has not been undertaken. NAG v6.1 does not have this problem.

11. Frequently asked questions

This section has now been put on the RTTOV v12 web site to allow updating:
http://nwpsaf.eu/site/software/rttov/documentation/faq/

12. Glossary

AD	RTTOV Adjoint model
AMSU	Advanced Microwave Sounding Unit
ATMS	Advanced Technology Microwave Sounder
ATOVS	Advanced TIROS Operational Vertical Sounder
B parameter/BPR	Back-scattering parameter for IR scattering simulations
BRDF	Bi-directional Reflectance Distribution Function
BRF	Bi-directional Reflectance Factor
BT	Brightness Temperature
CAMS	Copernicus Atmosphere Monitoring Service
CLW	Cloud Liquid Water
CNRM	Centre National de Recherches Météorologiques
DOM	Discrete Ordinates Method
ECMWF	European Centre for Medium-Range Weather Forecasts
EUMETSAT	European Organisation for the Exploitation of Meteorological Satellites
FASTEM	MW surface emissivity model
GCM	Global Circulation Model
HDF5	Hierarchical Data Format version 5
HTFRTC	Havemann-Taylor/HyperspecTral Fast Radiative Transfer Code, a PC-based fast RT model
IR	Infrared
IREMIS	New physically-based IR sea surface emissivity model
ISEM	Older IR sea surface emissivity model, depends only on zenith angle
IWC	Ice Water Content
K	RTTOV Jacobian model
LBLRTM	Line-By-Line Radiative Transfer Model used to generate RTTOV coefficients for visible/IR sensors
Liebe-89 MPM	Line-by-line model used to generate RTTOV coefficients for MW sensors
MFASIS	a Method for FAst Satellite Image Simulation, a fast visible cloudy radiance solver
MODIS	Moderate resolution Imaging Spectroradiometer
MHS	Microwave Humidity Sounder
MW	Microwave
NIR	Near-Infrared (see VIS below)
NLTE	Non Local Thermodynamic Equilibrium
NWP	Numerical Weather Prediction
OPAC	Optical Properties of Aerosols and Clouds
OpenMP	Application Programming Interface supporting multi-platform shared-memory parallel programming
PC	Principal Components
PMC	Pressure Modulated Cell
PW	Planck-Weighted
RTTOV	Radiative Transfer for TOVS
RTTOV-SCATT	RTTOV interface for MW cloud and hydrometeor scattering simulations
TELSEM2	A Tool to Estimate Land Surface Emissivities from Microwaves to Millimetres
TESSEM2	A Tool to Estimate Sea Surface Emissivities from Microwaves to Millimetres
TL	RTTOV Tangent Linear model
SAF	Satellite Applications Facility
SEVIRI	Spinning Enhanced Visible and Infrared Imager
SSMI/S	Special Sensor Microwave Imager/Sounder
SSU	Stratospheric Sounding Unit
TIROS	Television Infrared Observation Satellite
TOVS	TIROS Operational Vertical Sounder
VIS	Visible (here used synonymously with VIS/NIR)

13.References

Aires F., C. Prigent, F. Bernado, C. Jiménez, R. Saunders, and P. Brunel, 2010. A tool to estimate Land Surface Emissivities at Microwaves frequencies (TELSEM) for use in numerical weather preduction. *Q.J.R. Meteorol. Soc.* **137,** 690-699. doi: 10.1002/qj.803.

Baordo, F. and Geer, A.J., 2016. Assimilation of SSMIS humidity-sounding channels in all-sky conditions over land using a dynamic emissivity retrieval. *Q.J.R. Meteorol. Soc.*, **142**, 2854-2866.

Bauer P., E. Moreau, F. Chevallier, and U. O'Keeffe 2006 Multiple-scattering microwave radiative transfer for data assimilation applications. *Q.J.R. Meteorol. Soc.* **132**, 1259-1281.

Baum, B. A., P. Yang, A. J. Heymsfield, C. Schmitt, Y. Xie, A. Bansemer, Y. X. Hu, and Z. Zhang, 2011. Improvements to shortwave bulk scattering and absorption models for the remote sensing of ice clouds. *J. Appl. Meteor. Clim.*, **50**, 1037-1056.

Bormann, N, A. Geer, S. English, 2012. Evaluation of the microwave ocean surface emissivity model FASTEM-5 in the IFS. *ECMWF Technical Memorandum 667.*

Borbas, E. E. and B. C. Ruston, 2010. The RTTOV UWiremis IR land surface emissivity module. *NWP SAF report.* https://www.nwpsaf.eu/publications/vs_reports/nwpsaf-mo-vs-042.pdf

Borbas, E, 2014. The RTTOV UWiremis module Investigation into the angular dependence of IR surface emissivity. *NWP SAF report.* https://www.nwpsaf.eu/publications/vs_reports/nwpsaf-mo-vs-050.pdf

Borbas E. and Feltz M., 2019: Updating the CAMEL surface emissivity atlas for RTTOV. *NWP SAF report NWPSAF-MO-VS-058.* https://www.nwpsaf.eu/publications/vs_reports/nwpsaf-mo-vs-058.pdf

Boudala, F.S., Isaac, G.A., Fu, Q., and Cober, S.G., 2002: Parameterization of effective ice particle size for high latitude clouds. *Int. J. Climatol*., **22**, 1267-1284**.**

Bozzo, A., Remy, S., Benedetti, A., Flemming, J., Bechtold, P., Rodwell, M.J., Morcrette, J.J., 2017: Implementation of a CAMS-based aerosol climatology in the IFS. *ECMWF Technical Memorandum 801*

Brunel, P. and S. Turner 2003 On the use of Planck-weighted transmittances in RTTOV presented at the 13th International TOVS Study Conference, Ste Adele, Canada 29 Oct – 4 Nov 2003. http://cimss.ssec.wisc.edu/itwg/itsc/itsc13/thursday/brunel_poster.pdf

Chen, Y., Y. Han, P. van Delst, F. Weng, 2013: Assessment of Shortwave Infrared Sea Surface Reflection and Nonlocal Thermodynamic Equilibrium Effects in the Community Radiative Transfer Model Using IASI Data. *J. Atmos. Oceanic Technol*., 30, 2152–2160. doi: http://dx.doi.org/10.1175/JTECH-D-12-00267.1

Clark, R.N., Swayze, G.A., Wise, R., Livo, E., Hoefen, T., Kokaly, R., Sutley, S.J., 2007, USGS digital spectral library splib06a: U.S. Geological Survey, Digital Data Series 231, http://speclab.cr.usgs.gov/spectral.lib06.

Deblonde, G., 2000. Evaluation of FASTEM and FASTEM2, *NWP SAF report.* NWPSAF-MO-VS 1. Available here: https://www.nwpsaf.eu/site/download/documentation/rtm/papers/evalfastems.pdf https://www.nwpsaf.eu/site/download/documentation/rtm/papers/evalfastemsfigs.pdf

Elfouhaily, T., Chapron, B., Katsaros, K., and Vandemark, D., 1997: A unified directional spectrum for long and short wind-driven waves, *J. Geophys. Res.*, **102**(C7), 15781-15796, doi:10.1029/97JC00467.

English, S.J. and T.J. Hewison, 1998: A fast generic microwave emissivity model, *Proceedings of SPIE, 3503, Microwave Remote Sounding of the Environment*, Eds. T Hayasaka, D.L. Wu, Y. Jin and J. Jiang, 288-300

Eyre J. R. 1991 A fast radiative transfer model for satellite sounding systems. *ECMWF Technical Memorandum 176*

Geer A.J., P. Bauer and C. W. O'Dell, 2009a: A revised cloud overlap scheme for fast microwave radiative transfer in rain and cloud, *J. App. Met. Clim.*, **48**, 2257–2270

Geer, A.J., R.M. Forbes and P. Bauer, 2009b: Cloud and precipitation overlap in simplified scattering radiative transfer, *EUMETSAT/ECMWF Fellowship Programme Research Report no. 18.* http://www.ecmwf.int/en/research/publications

Guedj S., Karbou F., Rabier F., and Bouchard A., 2010: Toward a Better Modeling of Surface Emissivity to Improve AMSU Data Assimilation Over Antarctica. *IEEE Trans. Geosci. Remote Sensing* **48** 4 https://doi.org/10.1109/TGRS.2009.2036254

Han, H.-J., B.-J. Sohn, H.-L. Huang, E. Weisz, R. Saunders, and T. Takamura, 2012. An improved radiance simulation for hyperspectral infrared remote sensing of Asian dust, *J. Geophys. Res.*, **117**, D09211, doi:10.1029/2012JD017466.

Hasselmann, K. et al. (1973) Measurements of wind-wave growth and swell during the Joint North Sea Wave Project (JONSWAP) *Dtsch.Hydrogr.Z.*, **12**, 95 pp.

Havemann, S., J.-C. Thelen, J. P. Taylor and R. C. Harlow, 2018. The Havemann-Taylor Fast Radiative Transfer Code (HT-FRTC): a multipurpose code based on Principal Components. *JQSRT*, **220,** 180-192. https://doi.org/10.1016/j.jqsrt.2018.09.008

Hocking, J., 2014: Interpolation methods in the RTTOV fast radiative transfer model. Met Office Forecasting Research Technical Report 590. Available here: http://www.metoffice.gov.uk/binaries/content/assets/mohippo/pdf/i/k/frtr590.pdf

Johnson, B., K. Turnbull, P. Brown, R. Burgess, J. Dorsey, A. J. Baran, H. Webster, J. Haywood, R. Cotton, Z. Ulanowski, E. Hesse, A. Woolley, and P. Rosenberg, 2012: In situ observations of volcanic ash clouds from the FAAM aircraft during the eruption of Eyjafjallajökull in 2010, *J. Geophys. Res.*, **117**, D00U24, doi:10.1029/2011JD016760.

Karbou, F., E.Gérard, and F. Rabier, 2006: Microwave land emissivity and skin temperature for AMSU-A and -B assimilation over land, *Q.J.R. Meteorol. Soc.* **132**, No. 620, Part A, pp. 2333-2355(23), doi :10.1256/qj.05.216

Karbou, F., E. Gérard, and F. Rabier, 2010: Global 4DVAR assimilation and forecast experiments using AMSU observations over land. Part I: Impacts of various land surface emissivity parameterizations. *Wea. Forecasting*, **25,** 5–19.

Kazumori, M. and S.J. English, 2015: Use of the ocean surface wind direction signal in microwave radiance assimilation, *Q.J.R. Meteorol. Soc.* **141**, No. 689, Part B, pp 1354–1375, doi: 10.1002/qj.2445

Liebe, H.J., 1989: An atmospheric millimeter-wave propagation model. *Int. J. of Infrared and Millimeter Waves*, **10**, 6.

Liu Q. and F. Weng, 2003. Retrieval of sea surface wind vector from simulated satellite microwave polarimetric measurements. *Radio Sci.* **38**, 8078, doi: 10.1029/2002RS002729.

Liu, Q., F. Weng, and S. English, 2011: An Improved Fast Microwave Water Emissivity Model, *IEEE Geosci. Remote Sensing*, **49** 1238 – 1250. doi: 10.1109/TGRS.2010.2064779.

Matricardi, M., F. Chevallier and S. Tjemkes 2001 An improved general fast radiative transfer model for the assimilation of radiance observations. *ECMWF Technical Memorandum 345*. http://www.ecmwf.int/en/research/publications

Matricardi, M. 2003 RTIASI-4, a new version of the ECMWF fast radiative transfer model for the infrared atmospheric sounding interferometer. *ECMWF Technical Memorandum 425*. http://www.ecmwf.int/en/research/publications

Matricardi, M., 2005 The inclusion of aerosols and clouds in RTIASI, the ECMWF fast radiative transfer model for the Infrared Atmospheric Sounding Interferometer. *ECMWF Technical Memorandum 474*.

Matricardi, M. 2008 The generation of RTTOV regression coefficients for IASI and AIRS using a new profile training set and a new line-by-line database. *ECMWF Technical Memorandum 564.*

Matricardi, M. 2010 A principal component based version of the RTTOV fast radiative transfer model. *ECMWF Technical Memorandum 617.*

Matricardi M., López Puertas M., and Funke B. 2018: Modelling of nonlocal thermodynamic equilibrium effects in the principal component based version of the RTTOV fast radiative transfer model. *JGR Atmos.* **123**, 11, *5741-5761* https://doi.org/10.1029/2018JD028657

Matzler, C. 2005 On the determination of surface emissivity from Satellite observations. *Geoscience and Remote Sensing Letters*, **2,** *160-163.*

Mayer, B. and Kylling, A., 2005. Technical note: The libRadtran software package for radiative transfer calclations - description and examples of use, *Atmos. Chem. Phys.*, **5**, 1855-1877.

McFarquar, G.M., Iacobellis, S. & Somerville, R.C.J., 2003 : SCM simulations of tropical ice clouds using observationally based parameterizations of microphysics. *J. Clim.*, **16,** 1643-1664.

Ou, S. & Liou, K.-N., 1995: Ice microphysics and climatic temperature feedback. *Atmos. Res.*, **35,** *127-138*.

Pollack, J. B., O. B. Toon, and B. N. Khare (1973), Optical properties of some terrestrial rocks and minerals, *Icarus, 19, 372–389*, doi:10.1016/0019-1035(73)90115-2.

Prigent, C., Aires, F., Wang, D., Fox, S. and Harlow, C., 2016: Sea surface emissivity parameterization from microwaves to millimeter waves. *Q.J.R. Meteorol. Soc.* Accepted Author Manuscript. doi:10.1002/qj.2953

Rochon, Y., L. Garand, D.S. Turner and S. Polavarapu. 2007: Jacobian mapping between vertical co-ordinate systems in data assimilation. *Q.J.R. Meteorol. Soc*. **133** *1547-1558.*

Rosenkranz, P.W., 2015: A Model for the Complex Dielectric Constant of Supercooled Liquid Water at Microwave Frequencies. *IEEE Trans. on Geosci. and Remote Sensing*, **53**, 3, *1387-1393*.

Saunders R.W., M. Matricardi and P. Brunel 1999 An Improved Fast Radiative Transfer Model for Assimilation of Satellite Radiance Observations. *Q.J.R. Meteorol. Soc.* **125**, *1407-1425*.

Saunders, R., Hocking, J., Turner, E., Rayer, P., Rundle, D., Brunel, P., Vidot, J., Roquet, P., Matricardi, M., Geer, A., Bormann, N., and Lupu, C., 2018: An update on the RTTOV fast radiative transfer model (currently at version 12), Geosci. Model Dev., 11, 2717-2737, https://doi.org/10.5194/gmd-11-2717-2018

Scheck, L., P. Frèrebeau, R. Buras-Schnell, B. Mayer, 2016: A fast radiative transfer method for the simulation of visible satellite imagery. *JQSRT*, **175**, *54-67*.

Sherlock, V. 1999 ISEM-6: Infrared Surface Emissivity Model for RTTOV-6. *NWP SAF report.* https://www.nwpsaf.eu/site/download/documentation/rtm/papers/isem6.pdf

Turner, D.D., Kneifel, S., and Cadeddu, M.P. 2016: An Improved Liquid Water Absorption Model at Microwave Frequencies for Supercooled Liquid Water Clouds. *J. Atmos. Oceanic Tech.*, **33**, *33-44*, doi: 10.1175/JTECH-D-15-0074.1

Vidot, J. and E. Borbas, 2013: Land surface VIS/NIR BRDF atlas for RTTOV-11: Model and Validation against SEVIRI Land SAF Albedo product. *Q.J.R. Meteorol. Soc.* **140**, 2186–2196, doi: 10.1002/qj.2288

Vidot, J., A. J. Baran, and P. Brunel, 2015: A new ice cloud parameterization for infrared radiative transfer simulation of cloudy radiances: Evaluation and optimization with IIR observations and ice cloud profile retrieval products. *J. Geophys. Res. Atmos.*, **120**, 6937–6951. doi: 10.1002/2015JD023462.

Volz, F. E. (1972), Infrared refractive index of atmospheric aerosol substances, *Appl. Opt.*, **11,** 755–759, doi:10.1364/AO.11.000755.

Volz, F. E. (1973), Infrared optical constants of ammonium sulfate, Sahara dust, volcanic pumice, and flyash, *Appl. Opt.*, **12,** 564–568, doi:10.1364/ AO.12.000564.

Wang, D., C. Prigent, L. Kilic, S. Fox, R. C. Harlow, C. Jimenez, F. Aires, C. Grassotti, and F. Karbou, 2017. Surface emissivity at microwaves to millimeter waves over Polar Regions: parameterization and evaluation with aircraft experiments. *J. Atmos. and Oceanic Tech.* https://doi.org/10.1175/JTECH-D-16-0188.1

Wu, X. and W.L. Smith, Emissivity of rough sea surface for 8-13\mum: modeling and validation, *Appl. Opt.* **36**, 1-11

Wyser, K., 1998: The effective radius in ice clouds. *J. Clim.*, **11,** 1793-1802.

14.Annexes
Annex A - Coefficient information and conversion tools

1. RTTOV_COEF_INFO.EXE

The program **rttov_coef_info.exe** (located in the **bin/** directory) can be used to display information about any given *rtcoef_* coefficient file. This is particularly useful for determining the contents of binary or HDF5 coefficient files.

The usage is as follows:

```
$ rttov_coef_info.exe --coef ... --format FORMATTED|UNFORMATTED|HDF5 --verbose
```

RTTOV will usually determine format of the coefficient file automatically so the `--format` argument is not generally required. The `--verbose` option prints out additional per-channel information so the amount of output is greatly increased for hyperspectral sounders.

Argument	Description
--coef	Input coefficient file.
--format	Format of coefficient file (optional)
--verbose	Include additional per-channel information (optional).

2. RTTOV_MFASIS LUT_INFO.EXE

The program **rttov_mfasis_lut_info.exe** (located in the **bin/** directory) can be used to display information about a given HDF5 MFASIS LUT file. This is particularly useful for determining the cloud liquid and ice water schemes used for training the LUT and for finding out the range of values for which the LUT is trained in each dimension (see section 8.5).

The usage is as follows:

```
$ rttov_coef_info.exe –mfasis_lut ... --verbose
```

The executable only reads files in HDF5 format. The `--verbose` option additionally prints out the “README_LUT” section of the file which may contain additional information.

Argument	Description
--mfasis_lut	Input MFASIS LUT file.
--verbose	Include additional per-channel information (optional).

3. RTTOV_CONV_COEF_.EXE

The program **rttov_conv_coef.exe** (located in the **bin/** directory of the RTTOV build) is used to convert coefficient files between ASCII, Fortran unformatted (binary) and HDF5 formats and to create coefficient files for subsets of channels. A help message can be displayed as follows:

```
$ rttov_conv_coef.exe --help
```

The usage is as follows:

```
$ rttov_conv_coef.exe \
     --format-in FORMATTED|UNFORMATTED|HDF5 \
     --format-out FORMATTED|UNFORMATTED|HDF5 \
     --channels 1 2 3 4 5 ... \
     --coef-in  ... --scaer-in  ... --sccld-in  ... --pccoef-in  ... \
     --coef-out ... --scaer-out ... --sccld-out ... --pccoef-out ... \
     --hdf5-reals32 --force-single --all-in-one --compress
```

Argument	Description
--format-in FORMATTED | UNFORMATTED | HDF5	Format of input coefficient file(s). FORMATTED=ASCII; UNFORMATTED=binary, HDF5 only applicable if RTTOV compiled with HDF5 capability (optional).
--format-out FORMATTED | UNFORMATTED | HDF5	Format of output coefficient files(s).
--channels 1 2 3 4 5 ...	List of channels to extract (optional)
--coef-in/--coef-out	Input/output RTTOV coefficient file (output file optional).
--scaer-in/--scaer-out	Input/output aerosol scattering coefficient file (optional).
--sccld-in/--sccld-out	Input/output cloud scattering coefficient file (optional).
--pccoef-in/--pccoef-out	Input/output Principal Components coefficient file (optional).
--hdf5-reals32	If present store ALL reals in HDF5 32 bits, default is the RTTOV real Kind. Only applicable to HDF5 output (optional).
--force-single	If present store real 2D and larger arrays in HDF5 32 bits, default is the RTTOV default real kind. Only applicable to HDF5 output (optional).
--all-in-one	If present write all coefs (optical depth, scattering, PC) to a single output file. Only applicable to HDF5 output (optional).
--compress	If present use HDF5 internal GZIP compression, only useful for hi-res sounders. Only applicable to HDF5 output (optional).

Most arguments are optional, though both `--coef-in` and `--format-out` must be specified at least. RTTOV will try to determine the input file format if this is not specified. Note that in all cases an optical depth coefficient file (*rtcoef_**) will be created: if `--coef-out` is not specified this file is written to the directory containing the input coefficient file.

As described in section 7.4, when you extract some subset of *n* channels to a new coefficient file the channels will then be identified by the indices 1 to *n* in RTTOV and not by the original channel numbers. If you are carrying out PC calculations the channels that RTTOV must simulate are prescribed by the PC predictor selection chosen as described in section 8.8. In this case, if you wish to create a smaller coefficient file for use with these simulations, it is strongly recommended that you extract specifically the set of channels used as predictors for the PC calculations you require: if you extract some other subset of channels (which must in any case be a superset of the necessary PC predictor channels) it will become complicated to manage the channel numbering correctly.

Example 1 – convert an HDF5 file to binary format:

```
$ rttov_conv_coef.exe --format-out unformatted --coef-in rtcoef_eos_2_airs.H5
     --coef-out rtcoef_eos_2_airs.bin
```

Example 2 – extract a subset of channels to a compressed HDF5 file:

```
$ rttov_conv_coef.exe --format-out hdf5 --coef-in rtcoef_eos_2_airs.H5 \
     --coef-out rtcoef_eos_2_airs_subset.H5 --compress --channels 10 20 30 ...
```

4. RTTOV_ASCII2BIN_SCATTCOEF.EXE

The program **rttov_ascii2bin_scattcoef.exe** (located in the **bin/** directory) can be used to convert ASCII Mie table files to binary format.

The usage is as follows:

```
$ rttov_ascii2bin_scattcoef.exe --coef-in ...  --coef-out ...
```

Argument	Description
--coef-in	Input ASCII RTTOV-SCATT Mie table file.
--coef-out	Output binary format Mie table file.

5. RTTOV11_CONV_COEF_11TO12.EXE

The program **rttov11_conv_coef_11to12.exe** (located in the **bin/** directory) can be used to convert v10/v11-format coefficient files to v12 format. This works with ASCII and HDF5 files and the output file is in the same format as the input file.

The NLTE coefficients in v11 files are not compatible with the v12 NLTE model so these are not copied to the output file. The output v12-format file will not include coefficients for the IREMIS sea surface emissivity model so you must select ISEM as the emissivity model for IR sensors when running RTTOV (you can manually copy the IREMIS section from an original v12 coefficient file for the same sensor to the converted file).

The usage is as follows:

```
$ rttov11_conv_coef_11to12.exe --coef-in ...  --coef-out ...
```

Argument	Description
--coef-in	Input ASCII or HDF5 v10/v11 RTTOV coefficient file.
--coef-out	Output v12-compatible RTTOV coefficient file.

6. RTTOV11_CONV_COEF_12TO11.EXE

The program **rttov11_conv_coef_12to11.exe** (located in the **bin/** directory) can be used to convert v12-format coefficient files to v10/v11 format. This works with ASCII and HDF5 files and the output file is in the same format as the input file.

The NLTE coefficients in v12 files are not compatible with the v11 NLTE model so these are not copied to the output file. Similarly the IREMIS emissivity model is not implemented in v11 so these coefficients are not copied either.

The usage is as follows:

```
$ rttov11_conv_coef_12to11.exe --coef-in ...  --coef-out ...
```

Argument	Description
--coef-in	Input ASCII or HDF5 v12 RTTOV coefficient file.
--coef-out	Output v10/v11-compatible RTTOV coefficient file.

Annex B – RTTOV_ERRORHANDLING interface

```
call rttov_errorhandling (err_unit)
```

rttov_errorhandling may optionally be called at any time to set the Fortran file unit number to which output error messages are written. The default value is the one given in the **rttov_const** module (currently 0). On most platforms the standard error is 0, but for HP it is 7. You should set the value according to your system. If no call is made, it is the same as calling the routine with the default values.

Type	In/Out	Variable	Description
Integer	Intent(in)	err_unit	Logical error unit

Annex C – Coefficient allocation and deallocation subroutines

1. RTTOV_READ_COEFS interface

```
call rttov_read_coefs (
        err,
        coefs,
        opts,
        channels,
        channels_rec,
        form_coef,
        form_scaer,
        form_sccld,
        form_pccoef,
        file_coef,
        file_scaer,
        file_sccld,
        file_pccoef,
        file_id_coef,
        file_id_scaer,
        file_id_sccld,
        file_id_pccoef,
        instrument,
        path)
```

This subroutine is used to read the coefficient file(s). Only the arguments relevant to the required coefficient files are necessary. The format arguments may be one of "**formatted**", "**unformatted**" or "**hdf5**" (the latter only if the code was compiled with HDF5 capability). The routine will attempt to determine the format of input coefficient files automatically if this is not supplied. Note that the input files (optical depth, cloud, aerosol, MFASIS, PC) **do not need to be** in the same format. This means, for example, you can read ASCII optical depth coefficient and cloud property files and an HDF5 MFASIS LUT in the same call.

You can supply the path and filename of coefficient files or the logical unit of the file if it has already been opened. These are the recommended methods for reading coefficient files. If neither of these arguments are supplied, the **instrument** argument is mandatory so that the routine can use it to construct the coefficient filename(s). The satellite and instrument IDs are listed in Tables 2 and 3. By default the coefficient file(s) must be in the current directory, but you can specify an alternative directory via the **path** argument.

Important notes:

If the **channels(:)** argument is supplied to extract data for *n* channels from the coefficient file(s), then within RTTOV these channels are referred to using the indices *1...n* rather than the original channel numbers. In particular, these new indices should be used when populating the **chanprof(:)%chan** array.

The settings in the options structure must be consistent with the coefficients being read. When reading visible/IR cloud coefficient files **opts%rt_ir%addclouds** must be true and **opts%rt_ir%user_cld_opt_param** must be false. Similarly when reading aerosol coefficient files **opts%rt_ir%addclouds** must be true and **opts%rt_ir%user_cld_opt_param** must be false. When reading PC-RTTOV coefficient files **opts%rt_ir%pc%addpc** must be true and **opts%rt_ir%pc%addradrec** must be true if you want to calculate reconstruct radiances..

When reading PC-RTTOV coefficient files, if you specify the **channels(:)** argument then this channel list must be a super-set of the channel list which forms the predictor set for the selected PC regression configuration.

When calling RTTOV-SCATT you **must read all channels** from the coefficient file (i.e. do not use the **channels(:)** argument). The RTTOV-SCATT Mietable file is read using the **rttov_read_scattcoefs** subroutine (see below).

Type	In/Out	Variable	Description
Integer	Intent(out)	err	Return code
Type(rttov_coefs)	Intent(out)	coefs	RTTOV coefficients structure.
Type(rttov_options)	Intent(in)	opts	RTTOV options structure.
Integer	Intent(in), optional	channels(:)	List of channels to extract.
Integer	Intent(in), optional	channels_rec(:)	List of channels for which to calculate radiances from PC scores (only applicable if both **opts%rt_ir%pc%addpc** and **opts%rt_ir%pc%addradrec** are true).
Character	Intent(in), optional	form_coef	Format of RTTOV coefficient file: should be either "**unformatted**" (binary), "**formatted**" (ASCII) or "**hdf5**" (only applicable if RTTOV has been compiled with HDF5 functionality).
Character	Intent(in), optional	form_scaer	Format of visible/IR aerosol scattering coefficient file.
Character	Intent(in), optional	form_sccld	Format of visible/IR cloud scattering coefficient file.
Character	Intent(in), optional	form_pccoef	Format of Principal Components coefficient file.
Character	Intent(in), optional	file_coef	Name of RTTOV coefficient file.
Character	Intent(in), optional	file_scaer	Name of visible/IR aerosol scattering coefficient file.
Character	Intent(in), optional	file_sccld	Name of visible/IR cloud scattering coefficient file.
Character	Intent(in), optional	file_pccoef	Name of Principal Components coefficient file.
Integer	Intent(in), optional	file_id_coef	Logical unit of pre-opened RTTOV coefficient file.
Integer	Intent(in), optional	file_id_scaer	Logical unit of visible/IR aerosol scattering coefficient file.
Integer	Intent(in), optional	file_id_sccld	Logical unit of visible/IR cloud scattering coefficient file.
Integer	Intent(in), optional	file_id_pccoef	Logical unit of Principal Components coefficient file.
Integer	Intent(in), optional	instrument(3)	platform ID, satellite ID, instrument ID (see Tables 2/3). If no filename is supplied, the instrument argument is used to construct the coefficient file name(s).
Character	Intent(in), optional	path	Used with **instrument** argument, directory containing coefficient files (if omitted coefficient files are assumed to be in the current directory).

2. RTTOV_READ_COEFS_HTFRTC interface

```
call rttov_read_coefs_htfrtc (err, coefs, fname_coef, fname_sensor,
                              channels_rec)
```

This subroutine is used to read the input files for the HTFRTC model. HTFRTC simulations require two input files: the static file (specified in fname_coef) is required for all simulations. In addition a sensor-specific file is required (specified in fname_sensor). The channels for which reconstructed radiances are required are specified when the HTFRTC files are read in: if channels_rec is omitted then reconstructed radiances are calculated for all sensor channels. Otherwise you can specify a channel list here. Note that you cannot specify a channel list when calling RTTOV/HTFRTC (unlike for PC-RTTOV).

Type	In/Out	Variable	Description
Integer	Intent(out)	err	Return code
Type(rttov_coefs)	Intent(inout)	coefs	RTTOV coefficients structure.
Character	Intent(in)	fname_coef	Path to HTFRTC static coefficient file.
Character	Intent(in)	fname_sensor	Path to HTFRTC sensor-specific coefficient file.
Integer	Intent(in), optional	channels_rec(:)	List of channels for which to calculate radiances from PC scores. If omitted HTFRTC simulations will calculate reconstructed radiances for all sensor channels if **opts%htfrtc%reconstructed** is true.

3. RTTOV_DEALLOC_COEFS interface

```
call rttov_dealloc_coefs (err, coefs)
```

rttov_dealloc_coefs is called to de-allocate the memory for the **coefs** structure.

Type	In/Out	Variable	Description
Integer	Intent(out)	err	Return code
Type(rttov_coefs)	Intent(inout)	coefs	RTTOV coefficients structure.

4. RTTOV_READ_SCATTCOEFFS interface

```
call rttov_read_scattcoeffs (err, opts_scatt, coef_rttov, coef_scatt,
                             file_id, file_coef, path)
```

This subroutine is called to read the RTTOV-SCATT Mietable coefficients file into the **rttov_scatt_coef** structure. The filename (including full path) of the Mietable file may be specified in **file_coef**. Alternatively it is possible to open the Mietable file before calling this subroutine and to pass the logical unit into **rttov_read_scattcoeffs**. If neither optional argument is specified the filename of the Mietable file is constructed from the name of the platform and instrument in the RTTOV coefficient file. In this case the file is assumed to be in the current directory unless you specify an alternative directory in the **path** argument.

This subroutine can read both ASCII Mietable files (which have file extension ".dat") and binary formatted Mietable files created with **rttov_ascii2bin_scattcoef.exe** described in Annex A (which have file extension ".bin").

Type	In/Out	Variable	Description
Integer	Intent(out)	err	Return code
Type(rttov_options_scatt)	Intent(in)	opts_scatt	RTTOV-SCATT options structure.
Type(rttov_coefs)	Intent(in)	coef_rttov	RTTOV coefficients structure.
Type(rttov_scatt_coef)	Intent(inout)	coef_scatt	RTTOV Mietable coefficients structure.
Integer	Intent(in), optional	file_id	Logical unit of pre-opened RTTOV-SCATT Mietable file.
Character	Intent(in), optional	file_coef	Name of RTTOV-SCATT Mietable file.
Character	Intent(in), optional	path	If the **file_id** and **file_coef** arguments are not supplied you can specify the directory containing the Mietable file via this argument, otherwise RTTOV assumes the file is in the current directory.

5. RTTOV_DEALLOC_SCATTCOEFFS interface

```
call rttov_dealloc_scattcoeffs (coef_scatt)
```

rttov_dealloc_scattcoeffs is called to de-allocate the memory for the RTTOV-SCATT Mietable structure.

Type	In/Out	Variable	Description
Type(rttov_scatt_coef)	Intent(inout)	coef_scatt	RTTOV Mietable coefficients structure.

Annex D – RTTOV allocation/deallocation and initialisation subroutines

RTTOV provides four subroutines which may be used for allocating all input arrays and structures required by the direct, TL, AD and K models. Alternatively it is possible to call individual subroutines to allocate each input and output structure individually. Additionally there are subroutines which can be called to initialise the structures which can be useful to ensure data from a previous RTTOV call does not interfere with a subsequent call.

1. RTTOV_ALLOC_DIRECT interface

```
call rttov_alloc_direct (err, asw, nprofiles, nchanprof, nlevels, chanprof,
                         opts, profiles, coefs, transmission, radiance,
                         radiance2, calcemis, emissivity, calcrefl, reflectance,
                         aer_maxnmom, aer_nphangle, aer_opt_param, cld_maxnmom,
                         cld_nphangle, cld_opt_param, traj, npcscores,
                         nchannels_rec, pccomp, channels_rec, init)
```

rttov_alloc_direct may be called to allocate or de-allocate any or all input arrays and structures for the RTTOV direct model (with the exception of the cloudy profile structure for RTTOV-SCATT). It may be convenient to call a single allocation subroutine rather than multiple subroutines. The order of the arguments closely mirrors the arguments for **rttov_direct**.

It is important to note that array arguments passed to this subroutine (e.g. chanprof, profiles, calcemis) must be declared as pointers. The arrays themselves are (de)allocated by the call to this subroutine as well as any members within the derived types.

Note also that due to the large number of optional arguments it is common that named arguments must be supplied.

Type	In/Out	Variable	Description
Integer	Intent(out)	err	Return code
Integer	Intent(in)	asw	Switch (1=allocate; 0=deallocate)
Integer	Intent(in)	nprofiles	Number of profiles per call to RTTOV
Integer	Intent(in)	nchanprof	Number of channels simulated per call to RTTOV (size of chanprof array)
Integer	Intent(in)	nlevels	Number of profile levels
Type(rttov_chanprof)	**Pointer, optional**	chanprof(:)	chanprof array pointer: (de)allocated after call
Type(rttov_options)	Intent(in)	opts	RTTOV options structure
Type(rttov_profile)	**Pointer, optional**	profiles(:)	Profiles structure array pointer to be (de)allocated: the profiles array itself will be (de)allocated as well as the member arrays.
Type(rttov_coefs)	Intent(in)	coefs	RTTOV coefficient structure
Type(rttov_transmission)	Intent(inout), optional	transmission	Transmission structure to be (de)allocated
Type(rttov_radiance)	Intent(inout), optional	radiance	Radiance structure to be (de)allocated
Type(rttov_radiance2)	Intent(inout), optional	radiance2	Secondary radiance structure to be (de)allocated
Logical	**Pointer, optional**	calcemis(:)	calcemis array pointer: (de)allocated after call
Type(rttov_emissivity)	**Pointer, optional**	emissivity(:)	Surface emissivity array pointer: (de)allocated after call
Logical	**Pointer, optional**	calcrefl(:)	calcrefl array pointer: (de)allocated after call
Type(rttov_reflectance)	**Pointer, optional**	reflectance(:)	Surface reflectance array pointer: (de)allocated after call
Integer	Intent(in), optional	aer_maxnmom	Maximum number of coefficients in Legendre expansion of aerosol phase functions.
Integer	Intent(in), optional	aer_nphangle	Number of phase angles over which aerosol phase functions are to be defined.
Type(rttov_opt_param)	Intent(inout), optional	aer_opt_param	Aerosol optical parameters structure to be (de)allocated.
Integer	Intent(in), optional	cld_maxnmom	Maximum number of coefficients in Legendre expansion of cloud phase functions.
Integer	Intent(in), optional	cld_nphangle	Number of phase angles over which cloud phase functions are to be defined.
Type(rttov_opt_param)	Intent(inout), optional	cld_opt_param	Cloud optical parameters structure to be (de)allocated.
Type(rttov_traj)	Intent(inout), optional	traj	Trajectory structure
Integer	Intent(in), optional	npcscores	Number of principal components to calculate for each profile multiplied by the number of profiles.
Integer	Intent(in), optional	nchannels_rec	The number of channels for which reconstructed radiances are required multiplied by the number of profiles. Only needed if **opts%rt_ir%pc%addpc** and **opts%rt_ir%pc%addradrec** are true.
Type(rttov_pccomp)	Intent(inout), optional	pccomp	pccomp structure to be (de)allocated
Integer	**Pointer, optional**	channels_rec(:)	Array pointer for list of channel numbers for which to reconstruct radiances
Logical	Intent(in), optional	init	Initialise the newly allocated structures

2. RTTOV_ALLOC_TL interface

```
call rttov_alloc_tl (err, asw, nprofiles, nchanprof, nlevels, chanprof, opts,
                     profiles, profiles_tl, coefs, transmission,
                     transmission_tl, radiance, radiance_tl, calcemis,
                     emissivity, emissivity_tl, calcrefl, reflectance,
                     reflectance_tl, aer_maxnmom, aer_nphangle, aer_opt_param,
                     aer_opt_param_tl, cld_maxnmom, cld_nphangle, cld_opt_param,
                     cld_opt_param_tl, traj, traj_tl, npcscores, nchannels_rec,
                     pccomp, pccomp_tl, channels_rec, init)
```

rttov_alloc_tl may be called to allocate or de-allocate any or all input arrays and structures for the RTTOV TL model (with the exception of the cloudy profile structure for RTTOV-SCATT). It may be convenient to call a single allocation subroutine rather than multiple subroutines. The order of the arguments closely mirrors the arguments for **rttov_tl**.

It is important to note that array arguments passed to this subroutine (e.g. chanprof, profiles, calcemis) must be declared as pointers. The arrays themselves are (de)allocated by the call to this subroutine as well as any members within the derived types.

Note also that due to the large number of optional arguments it is common that named arguments must be supplied.

Most of the arguments are common to **rttov_alloc_direct**: only the additional arguments are listed in the following table.

Type	In/Out	Variable	Description
Type(rttov_profile)	**Pointer, optional**	profiles_tl(:)	Profiles TL structure array pointer to be (de)allocated: the profiles_tl array itself will be (de)allocated as well as the member arrays.
Type(rttov_transmission)	Intent(inout), optional	transmission_tl	Transmission TL structure to be (de)allocated
Type(rttov_radiance)	Intent(inout), optional	radiance_tl	Radiance TL structure to be (de)allocated
Type(rttov_emissivity)	**Pointer, optional**	emissivity_tl(:)	Surface emissivity TL array pointer: (de)allocated after call
Type(rttov_reflectance)	**Pointer, optional**	reflectance_tl(:)	Surface reflectance TL array pointer: (de)allocated after call
Type(rttov_opt_param)	Intent(inout), optional	aer_opt_param_tl	Aerosol optical parameter TL structure to be (de)allocated.
Type(rttov_opt_param)	Intent(inout), optional	cld_opt_param_tl	Cloud optical parameter TL structure to be (de)allocated.
Type(rttov_traj)	Intent(inout), optional	traj_tl	Trajectory TL structure
Type(rttov_pccomp)	Intent(inout), optional	pccomp_tl	pccomp TL structure to be (de)allocated

3. RTTOV_ALLOC_AD interface

```
call rttov_alloc_ad (err, asw, nprofiles, nchanprof, nlevels, chanprof, opts,
                     profiles, profiles_ad, coefs, transmission,
                     transmission_ad, radiance, radiance_ad, calcemis,
                     emissivity, emissivity_ad, calcrefl, reflectance,
                     reflectance_ad, aer_maxnmom, aer_nphangle, aer_opt_param,
                     aer_opt_param_ad, cld_maxnmom, cld_nphangle, cld_opt_param,
                     cld_opt_param_ad, traj, traj_ad, npcscores, nchannels_rec,
                     pccomp, pccomp_ad, channels_rec, init)
```

rttov_alloc_ad may be called to allocate or de-allocate any or all input arrays and structures for the RTTOV AD model (with the exception of the cloudy profile structure for RTTOV-SCATT). It may be convenient to call a single allocation subroutine rather than multiple subroutines. The order of the arguments closely mirrors the arguments for **rttov_ad**.

It is important to note that array arguments passed to this subroutine (e.g. chanprof, profiles, calcemis) must be declared as pointers. The arrays themselves are (de)allocated by the call to this subroutine as well as any members within the derived types.

Note also that due to the large number of optional arguments it is common that named arguments must be supplied.

Most of the arguments are common to **rttov_alloc_direct**: only the additional arguments are listed in the following table.

Type	In/Out	Variable	Description
Type(rttov_profile)	**Pointer, optional**	profiles_ad(:)	Profiles AD structure array pointer to be (de)allocated: the profiles_ad array itself will be (de)allocated as well as the member arrays.
Type(rttov_transmission)	Intent(inout), optional	transmission_ad	Transmission AD structure to be (de)allocated
Type(rttov_radiance)	Intent(inout), optional	radiance_ad	Radiance AD structure to be (de)allocated
Type(rttov_emissivity)	**Pointer, optional**	emissivity_ad(:)	Surface emissivity AD array pointer: (de)allocated after call
Type(rttov_reflectance)	**Pointer, optional**	reflectance_ad(:)	Surface reflectance AD array pointer: (de)allocated after call
Type(rttov_opt_param)	Intent(inout), optional	aer_opt_param_ad	Aerosol optical parameter AD structure to be (de)allocated.
Type(rttov_opt_param)	Intent(inout), optional	cld_opt_param_ad	Cloud optical parameter AD structure to be (de)allocated.
Type(rttov_traj)	Intent(inout), optional	traj_ad	Trajectory AD structure
Type(rttov_pccomp)	Intent(inout), optional	pccomp_ad	pccomp AD structure to be (de)allocated

4. RTTOV_ALLOC_K interface

```
call rttov_alloc_k (err, asw, nprofiles, nchanprof, nlevels, chanprof, opts,
                    profiles, profiles_k, coefs, transmission,
                    transmission_k, radiance, radiance_k, calcemis,
                    emissivity, emissivity_k, calcrefl, reflectance,
                    reflectance_k, aer_maxnmom, aer_nphangle, aer_opt_param,
                    aer_opt_param_k, cld_maxnmom, cld_nphangle, cld_opt_param,
                    cld_opt_param_k, traj, traj_k, npcscores, nchannels_rec,
                    pccomp, pccomp_k, profiles_k_pc, profiles_k_rec,
                    channels_rec, init)
```

rttov_alloc_k may be called to allocate or de-allocate any or all input arrays and structures for the RTTOV K model (with the exception of the cloudy profile structure for RTTOV-SCATT). It may be convenient to call a single allocation subroutine rather than multiple subroutines. The order of the arguments closely mirrors the arguments for **rttov_k**.

It is important to note that array arguments passed to this subroutine (e.g. chanprof, profiles, calcemis) must be declared as pointers. The arrays themselves are (de)allocated by the call to this subroutine as well as any members within the derived types.

Note also that due to the large number of optional arguments it is common that named arguments must be supplied.

Most of the arguments are common to **rttov_alloc_direct**: only the additional arguments are listed in the following table.

Type	In/Out	Variable	Description
Type(rttov_profile)	**Pointer, optional**	profiles_k(:)	Profiles K structure array pointer to be (de)allocated: the profiles_k array itself will be (de)allocated as well as the member arrays.
Type(rttov_transmission)	Intent(inout), optional	transmission_k	Transmission K structure to be (de)allocated
Type(rttov_radiance)	Intent(inout), optional	radiance_k	Radiance K structure to be (de)allocated
Type(rttov_emissivity)	**Pointer, optional**	emissivity_k(:)	Surface emissivity K array pointer: (de)allocated after call
Type(rttov_reflectance)	**Pointer, optional**	reflectance_k(:)	Surface reflectance K array pointer: (de)allocated after call
Type(rttov_opt_param)	Intent(inout), optional	aer_opt_param_k	Aerosol optical parameter K structure to be (de)allocated.
Type(rttov_opt_param)	Intent(inout), optional	cld_opt_param_k	Cloud optical parameter K structure to be (de)allocated.
Type(rttov_traj)	Intent(inout), optional	traj_k	Trajectory K structure
Type(rttov_pccomp)	Intent(inout), optional	pccomp_k	pccomp K structure to be (de)allocated
Type(rttov_profile)	**Pointer, optional**	profiles_k_pc(:)	PC score Jacobian structure array pointer to be (de)allocated: the profiles_k_pc array itself will be (de)allocated as well as the member arrays.
Type(rttov_profile)	**Pointer, optional**	profiles_k_rec(:)	PC reconstructued radiance Jacobian structure array pointer to be (de)allocated: the profiles_rec_k array itself will be (de)allocated as well as the member arrays.

5. RTTOV_ALLOC_PROF interface

```
call rttov_alloc_prof (err, nprofiles, profiles, nlevels,
                       opts, asw, coefs, init)
```

rttov_alloc_prof is called to allocate or de-allocate the memory for the **profiles** structure. When allocating **profiles_k**(:) for the K model remember that the size of the array is **nchanprof**. The **profiles**(:) array must have been allocated before calling this subroutine.

Type	In/Out	Variable	Description
Integer	Intent(out)	err	Return code
Integer	Intent(in)	nprofiles	Number of profiles per call to RTTOV
Type(rttov_profile)	Intent(inout)	profiles(nprofiles)	Array of profiles structures to be (de)allocated
Integer	Intent(in)	nlevels	Number of profile levels
Type(rttov_options)	Intent(in)	opts	RTTOV options structure
Integer	Intent(in)	asw	Switch (1=allocate; 0=deallocate)
Type(rttov_coefs)	Intent(in), optional	coefs	RTTOV coefficient structure. This is mandatory if either **opts%rt_ir%addclouds** or **opts%rt_ir%addaerosl** are true, otherwise it may be omitted.
Logical	Intent(in), optional	init	Additionally initialise profiles structure

6. RTTOV_INIT_PROF interface

```
call rttov_init_prof (profiles, p)
```

rttov_init_prof is used to initialise a previously allocated profiles structure. This is particularly useful when calling the adjoint or K models as the **profiles_ad/profiles_k** structures must be initialised before each call.

Type	In/Out	Variable	Description
Type(rttov_profile)	Intent(inout)	profiles(nprof)	Profiles structure to be initialised
Real	Intent(in), optional	p(nlevels)	Optional pressure profile with which to initialise profiles(:)%p(:) (this argument is not commonly used)

7. RTTOV_ALLOC_RAD interface

```
call rttov_alloc_rad (err, nchanprof, radiance, nlevels, asw, radiance2, init)
```

rttov_alloc_rad is called to allocate or de-allocate the memory for the **radiance** structure and optionally for the **radiance2** structure.

Type	In/Out	Variable	Description
Integer	Intent(out)	err	Return code
Integer	Intent(in)	nchanprof	Total number of channels being simulated
Type(rttov_radiance)	Intent(inout)	radiance	Radiance structure to be (de)allocated
Integer	Intent(in)	nlevels	Number of profile levels
Integer	Intent(in)	asw	Switch (1=allocate; 0=deallocate)
Type(rttov_radiance2)	Intent(inout), optional	radiance2	Secondary radiance structure to be created
Logical	Intent(in), optional	init	Additionally initialise radiance structure(s)

8. RTTOV_INIT_RAD interface

```
call rttov_init_rad (radiance, radiance2)
```

rttov_init_rad is used to initialise a previously allocated **radiance** structure, and optionally also a **radiance2** structure.

Type	In/Out	Variable	Description
Type(rttov_radiance)	Intent(inout)	radiance	Radiance structure to be initialised
Type(rttov_radiance2)	Intent(inout), optional	radiance2	Secondary radiance structure to be initialised

9. RTTOV_ALLOC_TRANSMISSION interface

```
call rttov_alloc_transmission (err, transmission, nlevels, nchanprof, asw, init)
```

rttov_alloc_transmission is called to allocate or de-allocate the memory for the **transmission** structure.

Type	In/Out	Variable	Description
Integer	Intent(out)	err	Return code
Type(rttov_transmission)	Intent(inout)	transmission	Transmission structure to be (de)allocated
Integer	Intent(in)	nlevels	Number of profile levels
Integer	Intent(in)	nchanprof	Total number of channels being simulated
Integer	Intent(in)	asw	Switch (1=allocate; 0=deallocate)
Logical	Intent(in), optional	init	Additionally initialise transmission structure

10. RTTOV_INIT_TRANSMISSION interface

```
call rttov_init_transmission (transmission)
```

rttov_init_transmission is used to initialise a previously allocated **transmission** structure.

Type	In/Out	Variable	Description
Type(rttov_transmission)	Intent(inout)	transmission	Transmission structure to be initialised

11. RTTOV_ALLOC_PCCOMP interface

```
call rttov_alloc_pccomp (err, pccomp, npcscores, asw, init, nchannels_rec, &
                         opts, nlevels)
```

rttov_alloc_pccomp is called to allocate or de-allocate the memory for the **pccomp** structure. This is only required for PC-RTTOV or HTFRTC simulations.

Type	In/Out	Variable	Description
Integer	Intent(out)	err	Return code
Type(rttov_pccomp)	Intent(inout)	pccomp	pccomp structure to be (de)allocated
Integer	Intent(in)	npcscores	Number of principal components to calculate for each profile multiplied by the number of profiles.
Integer	Intent(in)	asw	Switch (1=allocate; 0=deallocate)
Logical	Intent(in), optional	init	Additionally initialise pccomp structure
Integer	Intent(in), optional	nchannels_rec	The number of channels for which reconstructed radiances are required multiplied by the number of profiles. Only needed if **opts%rt_ir%pc%addradrec** (PC-RTTOV) or **opts%htfrtc_opts%reconstruct** (HTFRTC) is true.
Type(rttov_options)	Intent(in), optional	opts	RTTOV options structure, required for HTFRTC simulations in which case **opts%htfrtc_opts%htfrtc** must be true, and the **overcast** and/or **simple_cloud** options must be true if they will be requested in the simulations.
Integer	Intent(in), optional	nlevels	Number of levels in profile structure, required for HTFRTC only if **opts%htfrtc_opts%overcast** is true.

12. RTTOV_INIT_PCCOMP interface

```
call rttov_init_pccomp (pccomp)
```

rttov_init_pccomp is used to initialise a previously allocated **pccomp** structure.

Type	In/Out	Variable	Description
Type(rttov_pccomp)	Intent(inout)	pccomp	pccomp structure to be initialised

13. RTTOV_ALLOC_OPT_PARAM interface

```
call rttov_alloc_opt_param (err, opts, opt_param, nchanprof,
                            nlevels, nmom, nphangle, asw, init)
```

rttov_alloc_opt_param is called to allocate or de-allocate the memory for the **rttov_opt_param** structure (for aerosol or cloud optical parameters).

Type	In/Out	Variable	Description
Integer	Intent(out)	err	Return code
Type(rttov_options)	Intent(in)	opts	RTTOV options structure
Type(rttov_opt_param)	Intent(inout)	opt_param	Optical parameters structure to be (de)allocated
Integer	Intent(in)	nchanprof	Total number of channels being simulated
Integer	Intent(in)	nlevels	Number of profile levels
Integer	Intent(in)	nmom	Maximum number of coefficients in Legendre expansions of phase functions. Can be zero if DOM scattering solver is not used.
Integer	Intent(in)	nphangle	Number of phase angles over which phase functions are to be defined. Phase functions are required for solar scattering simulations. They may also be required for non-solar simulations to calculate the Legendre coefficients or the "bpr" parameters.
Integer	Intent(in)	asw	Switch (1=allocate; 0=deallocate)
Logical	Intent(in), optional	init	If true, zero the newly allocated structure.

14. RTTOV_INIT_OPT_PARAM interface

```
call rttov_init_opt_param (err, opts, opt_param)
```

rttov_init_opt_param is called to initialise phase angle variables in the **rttov_opt_param** structure (for aerosol or cloud optical parameters). NB This is only required if **opts%rt_ir%addsolar** is true.

It can also be used to initialise an optical property structure to zero using the **zero_only** argument: this is useful when calling the AD/K models.

Type	In/Out	Variable	Description
Integer	Intent(out)	err	Return code
Type(rttov_options)	Intent(in), optional	opts	RTTOV options structure (mandatory if the **zero_only** argument is omitted or is false).
Type(rttov_opt_param)	Intent(inout)	opt_param	Optical parameters structure to be initialised.
Logical	Intent(in)	zero_only	If true just initialise the structure to zero: use this to zero AD/K optical property structures. Set this to false (or omit it) for the direct model structure.

15. RTTOV_ALLOC_TRAJ interface

```
call rttov_alloc_traj (err, nprofiles, nchanprof, opts, nlevels, coefs,
                       asw, traj, traj_tl, traj_ad, traj_k)
```

When calling RTTOV a number of temporary structures are required. These are stored in an encompassing "trajectory" structure. To avoid allocating and deallocating these data structures inside every call to RTTOV this subroutine may be used to create trajectory structures so they can then be passed as arguments to every call to the RTTOV core routines (direct, TL, AD or K). Whether this is useful depends on the architecture/compiler: it has been seen to improve performance in some cases on Linux platforms so you may wish to consider this. In no case should it be detrimental to performance. Note that the trajectory structures *cannot* be used with the RTTOV parallel interfaces.

NB When passing the directory structure to RTTOV, the `coefs` variable MUST be declated with the TARGET attribute.

Once all radiances have been computed, you should call **rttov_alloc_traj** again with **asw** set to zero to deallocate the trajectory structure(s).

Type	In/Out	Variable	Description
Integer	Intent(out)	err	Return code.
Integer	Intent(in)	nprofiles	Number of profiles per call to RTTOV.
Integer	Intent(in)	nchanprof	Total number of channels being simulated
Type(rttov_options)	Intent(in)	opts	RTTOV options structure
Integer	Intent(in)	nlevels	Number of profile levels
Type(rttov_coefs)	Intent(in)	coefs	RTTOV coefficient structure.
Integer	Intent(in)	asw	Switch (1=allocate; 0=deallocate)
Type(rttov_traj)	Intent(inout), optional	traj, traj_tl, traj_ad, traj_k	Trajectory structures: a structure for each core RTTOV routine can be (de)allocated with a single call to **rttov_alloc_traj**.

16. RTTOV_ALLOC_SCATT_PROF interface

```
call rttov_alloc_scatt_prof (err, nprof, cld_profiles, nlev, use_totalice, asw,
                             init, mmr_snowrain)
```

rttov_alloc_scatt_prof is called to allocate or de-allocate the memory for the **cld_profiles** structure which contains cloud profile information for RTTOV-SCATT (the MW scattering model).

Type	In/Out	Variable	Description
Integer	Intent(out)	err	Return code.
Integer	Intent(in)	nprof	Number of profiles per call to RTTOV.
Type(rttov_profile_cloud)	Intent(inout)	cld_profiles(nprof)	Array of RTTOV-SCATT cloud profile structures to be (de)allocated.
Type(rttov_options)	Intent(in)	nlev	Number of levels in input profiles.
Logical	Intent(in)	use_totalice	Choose separate ciw and snow, or totalice
Integer	Intent(in)	asw	Switch (1=allocate; 0=deallocate)
Logical	Intent(in)	init	Additionally initialise cld_profiles structure
Logical	Intent(in), optional	mmr_snowrain	Units of snow and rain input units: False => kg/m^2/s; True => kg/kg (default)

17. RTTOV_INIT_SCATT_PROF interface

```
call rttov_init_scatt_prof (cld_profiles)
```

rttov_init_scatt_prof is used to initialise the **cld_profiles** structure. This is particularly useful when calling the adjoint or K models as the **cld_profiles_ad/cld_profiles_k** structures must be initialised before each call.

Type	In/Out	Variable	Description
Type(rttov_profile_cloud)	Intent(inout)	cld_profiles(nprof)	RTTOV-SCATT cloud profile structure to be initialised.

18. RTTOV_ALLOC_EMIS_RET_TERMS interface

```
call rttov_alloc_emis_ret_terms (err, nchanprof, emis_retrieval_terms, asw)
```

rttov_alloc_emis_ret_terms is called to allocate or de-allocate the memory for the emissivity retrieval terms structure which contains radiance and transmittance outputs from RTTOV-SCATT which can be used for all-sky emissivity retrievals. This is an optional argument to RTTOV-SCATT.

Type	In/Out	Variable	Description
Integer	Intent(out)	err	Return code.
Integer	Intent(in)	nchanprof	Total number of channels being simulated
Type(rttov_scatt_emis_retrieval_type)	Intent(inout)	emis_retrieval_terms	Emissivity retrieval terms structure to be (de)allocated.
Integer	Intent(in)	asw	Switch (1=allocate; 0=deallocate)

19. RTTOV_SCATT_SETUPINDEX interface

```
call rttov_scatt_setupindex (nprofiles, n_chan, coef, nchannels, chanprof,
                             frequencies, lchannel_subset)
```

Before calling RTTOV-SCATT you should make a call to **rttov_scatt_setupindex** which initialises the **chanprof** and the **frequencies** input arrays to RTTOV-SCATT. The **frequencies** array contains the indices into the Mietable coefficients for each channel to be simulated.

The **n_chan** argument must be the number of channels in the RTTOV coefficient file. The **chanprof(:)** array should have size equal to the total number of channels to simulate over all profiles. If you want to simulate every channel for every profile then the size of **chanprof(:)** is **nprofiles*n_chan** and the **lchannel_subset(:,:)** argument is not required. However, if you do not require certain channels to be simulated (either for every profile or for some profiles) then you can indicate the channels to be simulated for each profile by setting the corresponding elements of **lchannel_subset(:,:)** to .TRUE. In this case the **chanprof(:)** array must have size equal to the number of .TRUE. elements in **lchannel_subset(:,:)**.

Type	In/Out	Variable	Description
Integer	Intent(in)	nprofiles	Number of profiles.
Integer	Intent(in)	n_chan	Number of channels in coefficient structure.
Type(rttov_coefs)	Intent(in)	coef_rttov	RTTOV coefficients structure.
Integer	Intent(in)	nchannels	Total number of channels being simulated.
Type(rttov_chanprof)	Intent(out)	chanprof(nchannels)	RTTOV chanprof structure.
Integer	Intent(out)	frequencies(nchannels)	Frequency number for each channel being simulated.
Logical	Intent(in), optional	lchannel_subset(nprofiles, n_chan)	Logical array to specify which channels to simulate for which profiles.

Annex E – Optical parameter calculation subroutines

1. RTTOV_BPR_INIT interface

```
call rttov_bpr_init (err, phangle)
```

rttov_bpr_init is called before calculating *b* parameters from phase functions using **rttov_bpr_calc**: this subroutine prepares some tables to speed up the calculations.

Type	In/Out	Variable	Description
Integer	Intent(out)	err	Return code
Real	Intent(in)	phangle(:)	Array of angles over which phase functions are defined.

2. RTTOV_BPR_CALC interface

```
call rttov_bpr_calc (err, pha, phangle, bpr, nthreads)
```

rttov_bpr_calc is used to calculate a single *b* parameter given a phase function. Note that this is a relatively slow calculation and as such is intended to be called "off-line" rather than within performance-critical code. If you compile RTTOV with OpenMP this subroutine can exploit multiple cores/CPUs: choose the number of threads with the **nthreads** argument.

Type	In/Out	Variable	Description
Integer	Intent(out)	err	Return code
Real	Intent(in)	pha (:)	Phase function.
Real	Intent(in)	phangle(:)	Array of angles over which phase function *pha* is defined.
Real	Intent(out)	bpr	Calculated *b* parameter.
Integer	Intent(in), optional	nthreads	Use multiple threads, requires RTTOV to have been compiled with OpenMP, default = 1.

3. RTTOV_BPR_DEALLOC interface

```
call rttov_bpr_dealloc (err)
```

rttov_bpr_dealloc is called after *b* parameters have been calculated to deallocate memory.

Type	In/Out	Variable	Description
Integer	Intent(out)	err	Return code

4. RTTOV_LEGCOEF_CALC interface

```
call rttov_legcoef_calc (err, pha, phangle, nmom, legcoef, ngauss, q, w)
```

rttov_legcoef_calc is used to calculate the Legendre expansion of a given phase function. The **nmom** parameter need only equal the largest value of **opts%rt_ir%dom_nstreams** with which RTTOV will be called. The coefficients are calculated using a 1000 point Gaussian quadrature by default: you can specify an alternative quadrature size or, for greater efficiency (if making many calls to this subroutine), you can pre-compute a quadrature and the associated weights and pass these in.

Type	In/Out	Variable	Description
Integer	Intent(out)	err	Return code
Real	Intent(in)	pha (:)	Phase function.
Real	Intent(in)	phangle(:)	Array of angles over which phase function *pha* is defined.
Integer	Intent(in)	nmom	Number of Legendre coefficients to calculate (excluding first coefficient which is always unity).
Real	Intent(inout)	legcoef(:)	Calculated Legendre coefficients, array must be at least *nmom+1* in size.
Integer	Intent(in), optional	ngauss	Choose size of Gaussian quadrature to use in calculation (the default is 1000 points).
Real	Intent(in), optional	q(:)	You can supply pre-computed quadrature points which may be faster if calling this subroutine many times.
Real	Intent(in), optional	w(SIZE(q))	Weights corresponding to quadrature q.

Annex F – Emissivity atlas subroutines

This section describes the interface to the atlas subroutines. More information about using the atlases is provided in section 7.5. As of RTTOV v12 it is possible to use the TELSEM2 atlas without compiling RTTOV with any external library dependencies because the TELSEM2 atlas data are contained in ASCII files. The other atlas data are provided in HDF5 files and as such RTTOV must be compiled with HDF5 support (see section 5.2).

1. RTTOV_SETUP_EMIS_ATLAS interface

```
call rttov_setup_emis_atlas (err, opts, imonth, atlas_type, atlas,
                             atlas_id, path, coefs, ir_atlas_read_std,
                             ir_atlas_ang_corr, year)
```

The atlases provide monthly climatologies so you must provide the month (**imonth**) for which to load atlas data. By default the atlas files are assumed to be in the current directory, but you can optionally specify the **path** to the directory containing the atlas data files. The type of atlas (MW or IR) is selected by the **atlas_type** argument (1=>MW, 2=>IR).

The **atlas** argument is used to hold the loaded atlas data. In contrast to previous RTTOV versions you can initialise data for multiple atlases, months and instruments simultaneously: the only limitation is the available memory. The atlas argument is of derived type **rttov_emis_atlas_data** which is defined in the module **mod_emis_atlas_data**. You can declare as many instances of this type as you require and load the data from any atlas for any month and instrument into each one by calling this subroutine for each **atlas** variable.

The **atlas_id** argument is used to select between the available atlases: if omitted the default IR or MW atlas is selected. The valid IDs are as follows:

IR instruments

- UWIREmis – atlas_id = 1 (default)
- CAMEL 2007 atlas – atlas_id = 2
- CAMEL climatology atlas – atlas_id = 3

MW instruments

- TELSEM2 – atlas_id = 1 (default)
- CNRM MW atlas – atlas_id = 2

Once initialised the TELSEM2 atlas data can be used with any MW instrument: in this case the optional **coefs** argument is ignored. The CNRM MW atlas is always initialised for a specific instrument and the initialised **atlas** data can only be used with that particular instrument (AMSU-A, AMSU-B, MHS, SSMI/S or ATMS). In this case the **coefs** argument is mandatory and must contain the coefficients for a compatible sensor.

For the IR atlases the **coefs** argument is optional. If omitted the loaded atlas data can be used with any IR sensor. If the **coefs** argument is supplied the atlas data are loaded for use with that specific instrument. This makes it much faster to obtain emissivities from the atlas, but the loaded atlas data can only be used with that specific instrument. If you are only using the atlas for one IR instrument it is recommended to supply the **coefs** argument.

In addition the IR atlases can optionally include a zenith angle correction which is activated when reading the data by setting **ir_atlas_ang_corr** to true. The IR atlases can also optionally return emissivity data: this must be requested at setup by setting **ir_atlas_read_std** to true, but note that this increases the atlas memory requirements significantly. Note that if you do not require the standard deviations you do not need to download the standard deviation atlas data files.

The IR atlases can be used for HTFRTC in the same way as for standard RTTOV calls. The differences are that the **emissivity(:)**, **emis_std(:)** and **emis_flag(:)** arrays have size equal to the number of reconstructed radiance channels multiplied by the number of profiles, and the **chanprof** argument to **rttov_get_emis** is not used (see section 8.9).

Type	In/Out	Variable	Description
Integer	Intent(out)	err	Return code
Type(rttov_options)	Intent(in)	opts	RTTOV options structure
Integer	Intent(in)	imonth	Month (1-12) of atlas data to be loaded.
Integer	Intent(in)	atlas_type	Specifies whether to load a MW or IR atlas: 1 => MW, 2 => IR.
Type(rttov_emis_atlas_data) (from mod_rttov_emis_atlas)	Intent(inout)	atlas	Structure to hold the loaded atlas data.
Integer	Intent(in), optional	atlas_id	ID of atlas to use (see above); defaults to 1 for both IR and MW.
Character	Intent(in), optional	path	Path of directory containing emissivity atlas data files. Defaults to the current directory.
Type(rttov_coefs)	Intent(in), optional	coefs	RTTOV instrument coefficients structure. This is mandatory for the CNRM MW atlas and is ignored by TELSEM2. For the UW and CAMEL IR atlases, if this argument is passed they will be initialised specifically for use with this instrument: the atlases are then faster to access.
Logical	Intent(in), optional	ir_atlas_read_std	If true, initialise IR atlas error dataset. If the errors (standard deviations) are not required, this should be set to false: the atlas look-ups will be faster and the atlas will require much less memory. Default is false. Not applicable to MW atlases.
Logical	Intent(in), optional	ir_atlas_ang _corr	If true, initialise IR atlas so that zenith angle correction will be applied. This requires the additional angular correction data files. Default is false. Not applicable to MW atlases.
Integer	Intent(in), optional	year	This only applies to the CNRM MW atlas: data are available for multiple years as indicated in the atlas filenames. This argument selects the year (default=2015).

2. RTTOV_GET_EMIS interface

The **rttov_get_emis** subroutine takes the **profiles(:)** array as input: Table 15 lists the profile variables used by each atlas. Although there is only one routine to access the IR and MW atlases, each atlas has a number of options which are unique to it. Therefore the **rttov_get_emis** subroutine will be described separately for the IR and MW atlases, discussing just those arguments relevant in each case. If an argument is supplied which is not applicable to the given atlas, a warning message is printed and the argument is ignored. The output **emissivity(:)** array can be used as input to **rttov_direct**, **rttov_tl**, **rttov_ad** and **rttov_k** via **emissivity(:)%emis_in** with the corresponding elements of **calcemis(:)** set to false. However it is very important to check the emissivities that are returned to ensure they are positive values: the atlases may return negative values if they do not have data for the given surface type or lat/lon location. You may wish to set **calcemis(:)** to true where the returned emissivities are negative, for example.

In the discussions below, **nchanprof** is the size of the **chanprof(:)** array, **nprof** is the size of the **profiles(:)** array, and **nchan** is the largest number of channels computed for any given profile.

Returning IR emissivities (UWIRemis, CAMEL 2007 and CAMEL climatology atlases):

```
call rttov_get_emis (err, opts, chanprof, profiles, coefs, atlas,
                     emissivity, emis_std, emis_flag, snow_correction)
```

If the zenith angle correction was selected when calling **rttov_setup_emis_atlas** then the **zenangle** and **sunzenangle** profile members are required: **sunzenangle** is used to determine whether the day or night bias correction should be applied so it only needs to be less than 85º for day or greater than 85º for night (the exact value is not important). It returns emissivity values for both land and sea-ice surface types, and, over land, returns a linear combination of the land surface emissivity and a snow emissivity weighted according to the snow fraction. If the snow fraction is zero, just the land surface emissivities are returned. See section 8.4 for more information about the treatment of snow in the CAMEL climatology atlas. The IR atlases can optionally return the emissivity standard deviations: the CAMEL 2007 atlas can be used with the UWIRemis standard deviation dataset, while the CAMEL climatology atlas has a new standard deviation dataset derived from the multi-year climatology at higher spatial resolution. In addition, each surface point in the atlas has an associated flag which may be returned. You may wish to use this flag as a form of quality control (see the IR atlas documentation Borbas *et al.*, 2010 and the RTTOV v12 Science and Validation Report). Note that there is only one flag per profile (ie per location). However the **emis_flag(:)** output array is of size **nchanprof** to be consistent with the **emissivity(:)** array. If the **coefs** argument was passed to **rttov_setup_emis_atlas** you must ensure the **coefs** argument here is compatible with the **atlas** argument.

Type	In/Out	Variable	Description
Integer	Intent(out)	err	Return code
Type(rttov_options)	Intent(in)	opts	RTTOV options structure
Type(rttov_chanprof)	Intent(in)	chanprof(:)	Chanprof structure.
Type(rttov_profile)	Intent(in)	profiles(:)	Profiles structure.
Type(rttov_coefs)	Intent(in)	coefs	RTTOV instrument coefficient structure.
Type(rttov_emis_atlas_data) (from mod_rttov_emis_atlas)	Intent(in)	atlas	Atlas data previously loaded via **rttov_emis_atlas_setup**.
Real	Intent(out)	emissivity(nchanprof)	Emissivity values.
Real	Intent(out), optional	emis_std(nchanprof)	Emissivity errors (standard deviations).
Integer	Intent(out), optional	emis_flag(nchanprof)	Emissivity atlas flags.
Logical	Intent(in), optional	snow_correction	CAMEL climatology atlas only: if true then atlas attempts to find snow-free spectra when profile **snow_fraction** is zero, default true (recommended).

Returning MW emissivities (TELSEM2):

```
call rttov_get_emis (err, opts, chanprof, profiles, coefs, atlas,
                     emissivity, emis_std, emis_cov, resolution)
```

TELSEM2 consists of both an atlas and an interpolator which carries out interpolation of emissivity values in frequency and in space. The atlas has a nominal spatial resolution of 0.25°. If the **resolution** argument is supplied and is larger than 0.25, the emissivity values returned are integrated over the atlas grid according to the specified resolution. As with the IR atlases, the TELSEM MW atlas can optionally return estimated emissivity errors. It can also optionally return emissivity covariance matrices: this provides emissivity covariances among all channels for each profile.

Type	In/Out	Variable	Description
Integer	Intent(out)	err	Return code
Type(rttov_options)	Intent(in)	opts	RTTOV options structure
Type(rttov_chanprof)	Intent(in)	chanprof(:)	Chanprof structure.
Type(rttov_profile)	Intent(in)	profiles(:)	Profiles structure.
Type(rttov_coefs)	Intent(in)	coefs	RTTOV instrument coefficient structure.
Type(rttov_emis_atlas_data) (from mod_rttov_emis_atlas)	Intent(in)	atlas	Atlas data previously loaded via **rttov_emis_atlas_setup**.
Real	Intent(out)	emissivity(nchanprof)	Emissivity values.
Real	Intent(out), optional	emis_std(nchanprof)	Emissivity errors (standard deviations).
Real	Intent(out), optional	emis_cov(nprof, nchan, nchan)	Emissivity covariances.
Real	Intent(in), optional	resolution	Return emissivities at user-defined resolution. Units are degrees latitude/longitude. The default (i.e. nominal atlas) resolution is 0.25°.

Returning MW emissivities (CNRM MW atlas):

```
call rttov_get_emis (err, opts, chanprof, profiles, coefs, atlas, emissivity)
```

The CNRM atlas has emissivity datasets **only** for specific instruments and so does not need to carry out interpolation in frequency to instrument channels. It does not provide an estimate of emissivity error. The **coefs** argument must be compatible with the **atlas** data.

Type	In/Out	Variable	Description
Integer	Intent(out)	err	Return code
Type(rttov_options)	Intent(in)	opts	RTTOV options structure
Type(rttov_chanprof)	Intent(in)	chanprof(:)	Chanprof structure.
Type(rttov_profile)	Intent(in)	profiles(:)	Profiles structure.
Type(rttov_coefs)	Intent(in)	coefs	RTTOV instrument coefficient structure.
Type(rttov_emis_atlas_data) (from mod_rttov_emis_atlas)	Intent(in)	atlas	Atlas data previously loaded via **rttov_emis_atlas_setup**.
Real	Intent(out)	emissivity(nchanprof)	Emissivity values.

3. RTTOV_DEALLOCATE_EMIS_ATLAS interface

```
call rttov_deallocate_emis_atlas (atlas)
```

rttov_deallocate_emis_atlas is called to de-allocate the memory for an emissivity atlas data structure. This should be called for every loaded **atlas** structure.

Type	In/Out	Variable	Description
Type(rttov_emis_atlas_data) (from mod_rttov_emis_atlas)	Intent(inout)	atlas	Atlas data structure to deallocate.

Annex G – BRDF atlas subroutines

This section describes the interface to the atlas subroutines. More information about using the atlas is provided in section 7.6. The atlas data are provided in HDF5 files and as such RTTOV must be compiled with HDF5 support (see section 5.2).

1. RTTOV_SETUP_BRDF_ATLAS interface

```
call rttov_setup_brdf_atlas (err, opts, imonth, atlas, atlas_id, path, coefs)
```

The atlas provides monthly climatology so you must provide the month (**imonth**) for which to load atlas data. By default the atlas files are assumed to be in the current directory, but you can optionally specify the **path** to the directory containing the atlas data files.

The **atlas** argument is used to hold the loaded atlas data. In contrast to previous RTTOV versions you can initialise data for multiple months and instruments simultaneously: the only limitation is the available memory. The atlas argument is of derived type **rttov_brdf_atlas_data** which is defined in the module **mod_brdf_atlas_data**.

The **atlas_id** argument is used to select between the available atlases: currently there is only one BRDF atlas (atlas_id=1) so this argument can be omitted.

The **coefs** argument is optional. If omitted the loaded atlas data can be used with any visible/IR sensor. If the **coefs** argument is supplied the atlas data are loaded for use with that specific instrument. This makes it much faster to obtain BRDFs from the atlas, but the loaded atlas data can only be used with that specific instrument. If you are only using the atlas for one visible/IR instrument it is recommended to supply the **coefs** argument.

Type	In/Out	Variable	Description
Integer	Intent(out)	err	Return code
Type(rttov_options)	Intent(in)	opts	RTTOV options structure
Integer	Intent(in)	imonth	Month (1-12) of atlas data to be loaded.
Type(rttov_brdf_atlas_data) (from mod_rttov_brdf_atlas)	Intent(inout)	atlas	Structure to hold the loaded atlas data.
Integer	Intent(in), optional	atlas_id	ID of atlas to use. Currently there is only one atlas, so this argument can be omitted.
Character	Intent(in), optional	path	Path of directory containing BRDF atlas data files. Defaults to the current directory.
Type(rttov_coefs)	Intent(in), optional	coefs	RTTOV instrument coefficients structure. If this argument is passed the atlas will be initialised specifically for use with this instrument: the atlas is then faster to access.

2. RTTOV_GET_BRDF interface

The **rttov_get_brdf** subroutine takes the **profiles(:)** array as input: the routine uses the **latitude**, **longitude, zenangle, azangle, sunzenangle, sunazangle, skin%surftype** and **skin%watertype** members of each profile. The output **brdf(:)** array can be used as input to **rttov_direct**, **rttov_tl**, **rttov_ad** and **rttov_k** via **reflectance(:)%refl_in** with the corresponding elements of **calcrefl(:)** set to false. However it is very important to check the BRDFs that are returned to ensure they are positive values: the atlas may return negative values if it does not have data for the given surface type or lat/lon location. You may wish to set **calcrefl(:)** to true where the returned BRDFs are negative, for example. In the discussions below, **nchanprof** is the size of the **chanprof(:)** array.

```
call rttov_get_brdf (err, opts, chanprof, profiles, coefs, atlas,
                     brdf, brdf_flag, bh_albedo)
```

The BRDF atlas returns BRDF values for over land and also over sea, but it does not take sun glint into account (set **calcrefl** to true to use RTTOV's sea surface reflectance model instead). The BRDF atlas can optionally return the bi-hemispherical (black-sky) albedo. In addition, each surface point in the atlas has an associated flag which may be returned. You may wish to use this flag as a form of quality control (see the BRDF atlas documentation in the RTTOV v11 Science and Validation Report). Note that there is only one flag per profile (ie per location). However, the **brdf_flag(:)** output array is of size **nchanprof** to be consistent with the **brdf(:)** array. If the **coefs** argument was passed to **rttov_setup_brdf_atlas** you must ensure the **coefs** argument here is compatible with the **atlas** argument.

Type	In/Out	Variable	Description
Integer	Intent(out)	err	Return code
Type(rttov_options)	Intent(in)	opts	RTTOV options structure
Type(rttov_chanprof)	Intent(in)	chanprof(:)	Chanprof structure.
Type(rttov_profile)	Intent(in)	profiles(:)	Profiles structure.
Type(rttov_coefs)	Intent(in)	coefs	RTTOV instrument coefficient structure.
Type(rttov_brdf_atlas_data) (from mod_rttov_brdf_atlas)	Intent(in)	atlas	Atlas data previously loaded via **rttov_brdf_atlas_setup**.
Real	Intent(out)	brdf (nchanprof)	BRDF values.
Integer	Intent(out), optional	brdf_flag(nchanprof)	BRDF atlas flags.
Real	Intent(out), optional	bh_albedo(nchanprof)	Bi-hemispherical (black-sky) albedo.

3. RTTOV_DEALLOCATE_BRDF_ATLAS interface

```
call rttov_deallocate_brdf_atlas (atlas)
```

rttov_deallocate_brdf_atlas is called to de-allocate the memory for a BRDF atlas data structure. This should be called for every loaded **atlas** structure.

Type	In/Out	Variable	Description
Type(rttov_brdf_atlas_data) (from mod_rttov_brdf_atlas)	Intent(inout)	atlas	Atlas data structure to deallocate.

Annex H – RTTOV_GET_PC_PREDICTINDEX interface

```
call rttov_get_pc_predictindex(err, opts, predictindex, form_pccoef,
                               file_pccoef, file_id_pccoef, instrument)
```

This subroutine can be found in the **src/coef_io/** directory. It may be used to obtain the indices for the specified set of Principal Components regression channels, which depends on the setting of **opts%rt_ir%pc%ipcbnd** and **opts%rt_ir%pc%ipcreg**. Note that the regression channel set is available through the **coef_pccomp** structure after the PC coefficient file has been read (by calling **rttov_read_coefs** – see Annex C). This is demonstrated in **example_pc_fwd.F90**. The **rttov_get_pc_predictindex** routine may be used outside of RTTOV.

Before calling the subroutine, **opts%rt_ir%pc%ipcbnd** must be set to a value between 1 and 3 (IASI clear-sky) or to 1 (IASI cloudy, AIRS) and **opts%rt_ir%pc%ipcreg** must be set to a value between 1 and 3 for AIRS or 1 and 4 for IASI, corresponding to the predictor channel set required (see Section 8.8). Either the filename, the logical unit of a pre-opened coefficient file, or the instrument ID triplet should be supplied.

Type	In/Out	Variable	Description
Integer	Intent(out)	err	Return code.
Type(rttov_options)	Intent(in)	opts	RTTOV options structure
Integer, Pointer	Intent(out)	predictindex(:)	The output channel list. This array is allocated with the appropriate dimension within the routine.
Character	Intent(in), optional	form_pccoef	Format of PC coefficient file: should be either "**unformatted**" (binary) or "**formatted**" (ASCII) or "**hdf5**".
Character	Intent(in), optional	file_pccoef	Filename of PC coefficient file.
Integer	Intent(in), optional	file_id_pccoef	Logical unit of pre-opened PC coefficient file.
Integer	Intent(in), optional	instrument(3)	platform id; satellite id, instrument id (see Tables 2/3). If no filename is supplied, the instrument argument is used to construct the coefficient file name.

An example of how this routine might be used is included in comments in **example_pc_fwd.F90**. The executable **rttov_test_get_pc_predictindex** can be used to test the above subroutine:

```
$ rttov_test_get_pc_predictindex.exe --pccoef-in ...  --ipcbnd ... --ipcreg ...
```

where the arguments are the PC coefficient file name and the indices of the PC band and regression set respectively.

Annex I – RTTOV_DIRECT interface

```
call rttov_direct (errorstatus, chanprof, opts, profiles, coefs, transmission,
                   radiance, radiance2, calcemis, emissivity, calcrefl,
                   reflectance, aer_opt_param, cld_opt_param, traj, pccomp,
                   channels_rec)
```

rttov_direct should be called for each instrument required. Each call carries out computations for the profiles in the array **profiles(:)**. The **transmission** structure contains the calculated atmospheric transmittances and the **radiance** (and optionally **radiance2**) contain the calculated radiances, brightness temperatures and reflectances (see Annex O for information on the output structures).

The **chanprof(:)** array must be set up with channel and profile indexes as illustrated in Table 13. For Principal Components calculations (see section 8.8) the **chanprof(:)** array must be set up for the PC predictor channels. Therefore **chanprof(:)%chan** should be populated with the channel list defined by **opts%rt_ir%pc%ipcbnd** and **opts%rt_ir%pc%ipcreg** for each profile being simulated in the call to **rttov_direct**.

For most simulations the **calcemis(:)** and **emissivity(:)** arguments must be supplied. These are used to pass surface emissivities into and out from RTTOV and to specify whether RTTOV should calculate emissivities internally. See sections 7.5 and 8.4 for more information about surface emissivity.

For solar simulations the **calcrefl(:)** and **reflectance(:)** arguments must be supplied. These relate to the surface BRDF and are analogous to the corresponding emissivity arrays. See sections 7.6 and 8.2 for more information about surface reflectance.

Note that if no solar calculations are required the **calcrefl** and **reflectance** arguments may be omitted. Likewise, if carrying out simulations **only** for channels with no significant thermally emitted contribution (wavelengths < 3µm) the **calcemis** and **emissivity** arguments may be omitted.

For IR scattering calculations for which you wish to supply the scattering optical parameters directly, the **aer_opt_param** (for aerosols) and/or the **cld_opt_param** (for clouds) arguments should be supplied as described in sections 8.5 and 8.6.

The "trajectory" structure **traj** may be (de)allocated outside of RTTOV and passed in to improve performance on some architectures as described in section 7.7.

The files **src/test/example_fwd.F90** and **src/test/example_pc_fwd.F90** provide examples of running **rttov_direct** for standard RTTOV and PC-RTTOV. The files **src/test/example_aer_file_fwd.F90** and **src/test/example_cld_file_fwd.F90** provide examples of running **rttov_direct** for IR aerosol and cloud scattering using coefficient files. The files **src/test/example_aer_param_fwd.F90** and **src/test/example_cld_param_fwd.F90** provide examples of running **rttov_direct** for IR aerosol and cloud scattering where the scattering optical parameters are supplied to RTTOV explicitly via **aer_opt_param/cld_opt_param**.

In the following table **nchanprof** is the size of the **chanprof(:)** array (i.e. the total number of radiances to compute), and **nprof** is the size of the **profiles(:)** array (i.e. the number of profiles to process in each call to RTTOV). **nchannelsrec** is the number of reconstructed radiances required per profile for Principal Components calculations.

The **rttov_parallel_direct** subroutine has the same arguments as **rttov_direct** plus an optional final argument **nthreads** to specify the number of threads. Note that the trajectory argument cannot be used by the parallel routine.

There are three additional optional arguments to **rttov_direct** (traj_sta, traj_dyn and lbl_check) which are not intended for use in typical calls.

Type	In/Out	Variable	Description
Integer	Intent(out)	errorstatus	Return code.
Type(rttov_chanprof)	Intent(in)	chanprof(nchanprof)	Chanprof structure.
Type(rttov_options)	Intent(in)	opts	RTTOV options structure
Type(rttov_profile)	Intent(in)	profiles(nprof)	Profiles structure.
Type(rttov_coefs)	Intent(in)	coefs	RTTOV coefficient 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).
Logical	Intent(in), optional	calcemis(nchanprof)	.true. if RTTOV should calculate/select surface emissivity internally or .false. if user is supplying an emissivity value (e.g. from an atlas).
Type(rttov_emissivity)	Intent(inout), optional	emissivity(nchanprof)	Input/output emissivities.
Logical	Intent(in), optional	calcrefl(nchanprof)	.true. if RTTOV should calculate/select surface BRDF internally or .false. if user is supplying a BRDF value (e.g. from the atlas).
Type(rttov_reflectance)	Intent(in), optional	reflectance(nchanprof)	Input/output reflectances.
Type(rttov_opt_param)	Intent(in), optional	aer_opt_param	Aerosol optical parameter input profiles if **opts%rt_ir%addaerosl** and **opts%rt_ir%user_aer_opt_param** are true.
Type(rttov_opt_param)	Intent(in), optional	cld_opt_param	Cloud optical parameter input profiles if **opts%rt_ir%addclouds** and **opts%rt_ir%user_cld_opt_param** are 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_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%rt_ir%pc%addradrec** is .true.

Annex J – RTTOV_K interface

```
call rttov_k (errorstatus, chanprof, opts, profiles, profiles_k, coefs,
              transmission, transmission_k, radiance, radiance_k,
              calcemis, emissivity, emissivity_k, calcrefl, reflectance,
              reflectance_k, aer_opt_param, aer_opt_param_k, cld_opt_param,
              cld_opt_param_k, traj, traj_k, pccomp, pccomp_k, profiles_k_pc,
              profiles_k_rec, channels_rec)
```

The file `src/test/example_k.F90` provides an example of calling the RTTOV K model. **rttov_k** should be called for each instrument required. Each call carries out computations for the profiles in the array **profiles(:)**. **rttov_k** calculates the Jacobian for the RTTOV direct model, the output being written to **profiles_k**. As such, the **profiles_k** array should have the same size as the **chanprof** array (i.e. the total number of radiances being computed). The **emissivity_k(:)%emis_in** and **reflectance_k(:)%refl_in** arrays contain the Jacobians for the surface emissivity and BRDF respectively.

See section 7.9 for more information about the RTTOV K model. The file **src/test/example_k.F90** provides a simple example of calling the RTTOV K model for clear-sky simulations.

Note that in RTTOV v9 the **radiance_k** argument was optional, but in RTTOV v12 this argument is mandatory. To obtain the same behaviour as the default in RTTOV v9 you should set either **radiance_k%total(:)** or **radiance_k%bt(:)** to 1.0, depending on the setting of **opts%rt_all%switchrad**.

All K arguments should be initialised to zero before calling **rttov_k**. The only exception is the array (or arrays) in which the input perturbations are specified in **radiance_k** (or **pccomp_k** for PC-RTTOV).

Note that the emissivity and reflectance arguments are optional for **rttov_k** in exactly the same way as for **rttov_direct**.

In the following table, **nchanprof** is the size of the **chanprof(:)** array, and **nprof** is the size of the **profiles(:)** array. **npcscores** is the number of PC scores requested per profile, and **nchannelsrec** is the number of reconstructed radiances required per profile for Principal Components calculations.

The **rttov_parallel_k** subroutine has the same arguments as **rttov_k** plus an optional final argument **nthreads** to specify the number of threads. Note that the trajectory argument cannot be used by the parallel routine.

Type	In/Out	Variable	Description
Integer	Intent(out)	errorstatus	Return code.
Type(rttov_chanprof)	Intent(in)	chanprof(nchanprof)	Chanprof structure.
Type(rttov_options)	Intent(in)	opts	RTTOV options structure
Type(rttov_profile)	Intent(in)	profiles(nprof)	Profiles structure.
Type(rttov_profile)	Intent(inout)	profiles_k(nchanprof)	Jacobian on profile variables.
Type(rttov_coefs)	Intent(in)	coefs	RTTOV coefficient structure.
Type(rttov_transmission)	Intent(inout)	transmission	Output transmittances (0-1).
Type(rttov_transmission)	Intent(inout)	transmission_k	Jacobian of transmittances.
Type(rttov_radiance)	Intent(inout)	radiance	Direct model output radiances (mW/cm^{-1}/sr/m^2, degK, BRF/unitless).
Type(rttov_radiance)	Intent(inout)	radiance_k	Input radiance perturbations (see text).
Logical	Intent(in), optional	calcemis(nchanprof)	True if RTTOV should calculate/select surface emissivity internally or false if user is supplying an emissivity value.
Type(rttov_emissivity)	Intent(inout), optional	emissivity(nchanprof)	Input/output emissivities.
Type(rttov_emissivity)	Intent(inout), optional	emissivity_k(nchanprof)	Jacobian on surface emissivity.
Logical	Intent(in), optional	calcrefl(nchanprof)	True if RTTOV should calculate/select surface BRDF internally or false if user is supplying a BRDF value.
Type(rttov_reflectance)	Intent(inout), optional	reflectance(nchanprof)	Input/output BRDFs.
Type(rttov_reflectance)	Intent(inout), optional	reflectance_k(nchanprof)	Jacobian on surface BRDF.
Type(rttov_opt_param)	Intent(in), optional	aer_opt_param	Aerosol optical parameter input profiles if **opts%rt_ir%addaerosl** and **opts%rt_ir%user_aer_opt_param** are true.
Type(rttov_opt_param)	Intent(inout), optional	aer_opt_param_k	Jacobian of aerosol optical properties. Optional even if **aer_opt_param** argument is present.
Type(rttov_opt_param)	Intent(in), optional	cld_opt_param	Cloud optical parameter input profiles if **opts%rt_ir%addclouds** and **opts%rt_ir%user_cld_opt_param** are true.
Type(rttov_opt_param)	Intent(inout), optional	cld_opt_param_k	Jacobian of cloud optical properties. Optional even if **cld_opt_param** argument is present.
Type(rttov_traj)	Intent(inout), optional	traj	Trajectory structure to hold temporary data.
Type(rttov_traj)	Intent(inout), optional	traj_k	Trajectory structure to hold temporary data.
Type(rttov_pccomp)	Intent(inout). optional	pccomp	Structure to hold output PC scores and reconstructed radiances.
Type(rttov_pccomp)	Intent(inout). optional	pccomp_k	Input PC or reconstructed radiance perturbations (see text).
Type(rttov_profile)	Intent(inout). optional	profiles_k_pc(npcscores* nprofiles)	Jacobian on principal component scores. Only required if **opts%rt_ir%pc%addradrec** is .false.
Type(rttov_profile)	Intent(inout). optional	profiles_k_rec(nchannelsrec* nprofiles)	Jacobian on profile variables for reconstructed radiance channels. Only required if **opts%rt_ir%pc%addradrec** is .true.
Integer	Intent(in), optional	channels_rec(nchannelsrec)	Channels for which to compute reconstructed radiances if **opts%rt_ir%pc%addradrec** is .true.

Annex K – RTTOV_TL interface

```
call rttov_tl (errorstatus, chanprof, opts, profiles, profiles_tl, coefs,
               transmission, transmission_tl, radiance, radiance_tl,
               calcemis, emissivity, emissivity_tl, calcrefl, reflectance,
               reflectance_tl, aer_opt_param, aer_opt_param_tl, cld_opt_param,
               cld_opt_param_tl, traj, traj_tl, pccomp, pccomp_tl, channels_rec)
```

rttov_tl should be called for each instrument required. Each call carries out computations for the profiles in the array **profiles(:)**. **rttov_tl** calculates the tangent linear of the RTTOV direct model given a profile perturbation in **profiles_tl(:)**.

See section 7.9 for more information about the RTTOV TL model.

Note that the emissivity and reflectance arguments are optional for **rttov_tl** in exactly the same way as for **rttov_direct**.

In the following table, **nchanprof** is the size of the **chanprof(:)** array, and **nprof** is the size of the **profiles(:)** array. **nchannelsrec** is the number of reconstructed radiances required per profile for Principal Components calculations.

The **rttov_parallel_tl** subroutine has the same arguments as **rttov_tl** plus an optional final argument **nthreads** to specify the number of threads. Note that the trajectory argument cannot be used by the parallel routine.

Type	In/Out	Variable	Description
Integer	Intent(out)	errorstatus	Return code.
Type(rttov_chanprof)	Intent(in)	chanprof(nchanprof)	Chanprof structure.
Type(rttov_options)	Intent(in)	opts	RTTOV options structure
Type(rttov_profile)	Intent(in)	profiles(nprof)	Profiles structure.
Type(rttov_profile)	Intent(inout)	profiles_tl(nprof)	Input profile variable increments.
Type(rttov_coefs)	Intent(in)	coefs	RTTOV coefficient structure.
Type(rttov_transmission)	Intent(inout)	transmission	Output transmittances (0-1).
Type(rttov_transmission)	Intent(inout)	transmission_tl	TL of transmittances.
Type(rttov_radiance)	Intent(inout)	radiance	Direct model output radiances ($mW/cm^{-1}/sr/m^2$, degK, BRF/unitless).
Type(rttov_radiance)	Intent(inout)	radiance_tl	TL of radiances.
Logical	Intent(in), optional	calcemis(nchanprof)	.true. if RTTOV should calculate/select surface emissivity internally or .false. if user is supplying an emissivity value (e.g. from an atlas).
Type(rttov_emissivity)	Intent(inout), optional	emissivity(nchanprof)	Input/output emissivities.
Type(rttov_emissivity)	Intent(inout), optional	emissivity_tl(nchanprof)	Input/output surface emissivity TL.
Logical	Intent(in), optional	calcrefl(nchanprof)	.true. if RTTOV should calculate/select surface BRDF internally or .false. if user is supplying a BRDF value (e.g. from the atlas).
Type(rttov_reflectance)	Intent(inout), optional	reflectance(nchanprof)	Input/output BRDFs.
Type(rttov_reflectance)	Intent(inout), optional	reflectance_tl(nchanprof)	Input/output surface BRDF TL.
Type(rttov_opt_param)	Intent(in), optional	aer_opt_param	Aerosol optical parameter input profiles if **opts%rt_ir%addaerosl** and **opts%rt_ir%user_aer_opt_param** are true.
Type(rttov_opt_param)	Intent(inout), optional	aer_opt_param_tl	Input aerosol optical property increments. Optional even if **aer_opt_param** argument is present.
Type(rttov_opt_param)	Intent(in), optional	cld_opt_param	Cloud optical parameter input profiles if **opts%rt_ir%addclouds** and **opts%rt_ir%user_cld_opt_param** are true.
Type(rttov_opt_param)	Intent(inout), optional	cld_opt_param_tl	Input cloud optical property increments. Optional even if **cld_opt_param** argument is present.
Type(rttov_traj)	Intent(inout), optional	traj	Trajectory structure to hold temporary data.
Type(rttov_traj)	Intent(inout), optional	traj_tl	Trajectory structure to hold temporary data.
Type(rttov_pccomp)	Intent(inout), optional	pccomp	Structure to hold output PC scores and reconstructed radiances.
Type(rttov_pccomp)	Intent(inout), optional	pccomp_tl	TL of principal components.
Integer	Intent(in), optional	channels_rec(nchannelsrec)	Channels for which to compute reconstructed radiances if **opts%rt_ir%pc%addradrec** is .true.

Annex L – RTTOV_AD interface

```
call rttov_ad (errorstatus, chanprof, opts, profiles, profiles_ad, coefs,
               transmission, transmission_ad, radiance, radiance_ad,
               calcemis, emissivity, emissivity_ad, calcrefl, reflectance,
               reflectance_ad, aer_opt_param, aer_opt_param_ad, cld_opt_param,
               cld_opt_param_ad, traj, traj_ad, pccomp, pccomp_ad, channels_rec)
```

rttov_ad should be called for each instrument required. Each call carries out computations for the profiles in the array **profiles(:)**. **rttov_ad** calculates the adjoint of the RTTOV direct model, the output being written to **profiles_ad**. The **emissivity_ad(:)%emis_in** and **reflectance_ad(:)%refl_in** arrays contain the adjoint for the surface emissivity and surface BRDF respectively.

All AD arguments should be initialised to zero before calling **rttov_ad**. The only exception is the array (or arrays) in which the input perturbations are specified in **radiance_ad**.

See section 7.9 for more information about the RTTOV AD model.

Note that the emissivity and reflectance arguments are optional for **rttov_ad** in exactly the same way as for **rttov_direct**.

In the following table, **nchanprof** is the size of the **chanprof(:)** array, and **nprof** is the size of the **profiles(:)** array. **nchannelsrec** is the number of reconstructed radiances required per profile for Principal Components calculations.

The **rttov_parallel_ad** subroutine has the same arguments as **rttov_ad** plus an optional final argument **nthreads** to specify the number of threads. Note that the trajectory argument cannot be used by the parallel routine.

Type	In/Out	Variable	Description
Integer	Intent(out)	errorstatus	Return code.
Type(rttov_chanprof)	Intent(in)	chanprof(nchanprof)	Chanprof structure.
Type(rttov_options)	Intent(in)	opts	RTTOV options structure
Type(rttov_profile)	Intent(in)	profiles(nprof)	Profiles structure.
Type(rttov_profile)	Intent(inout)	profiles_ad(nprof)	AD on profile variables.
Type(rttov_coefs)	Intent(in)	coefs	RTTOV coefficient structure.
Type(rttov_transmission)	Intent(inout)	transmission	Output transmittances (0-1).
Type(rttov_transmission)	Intent(inout)	transmission_ad	AD of transmittances.
Type(rttov_radiance)	Intent(inout)	radiance	Direct model output radiances ($mW/cm^{-1}/sr/m^2$, degK, BRF/unitless).
Type(rttov_radiance)	Intent(inout)	radiance_ad	Input radiance perturbations (see text).
Logical	Intent(in), optional	calcemis(nchanprof)	.true. if RTTOV should calculate/select surface emissivity internally or .false. if user is supplying an emissivity value (e.g. from an atlas).
Type(rttov_emissivity)	Intent(inout), optional	emissivity(nchanprof)	Input/output emissivities.
Type(rttov_emissivity)	Intent(inout), optional	emissivity_ad(nchanprof)	AD on surface emissivity.
Logical	Intent(in), optional	calcrefl(nchanprof)	.true. if RTTOV should calculate/select surface BRDF internally or .false. if user is supplying a BRDF value (e.g. from the atlas).
Type(rttov_reflectance)	Intent(inout), optional	reflectance(nchanprof)	Input/output BRDFs.
Type(rttov_reflectance)	Intent(inout), optional	reflectance_ad(nchanprof)	AD on surface BRDF.
Type(rttov_opt_param)	Intent(in), optional	aer_opt_param	Aerosol optical parameter input profiles if **opts%rt_ir%addaerosl** and **opts%rt_ir%user_aer_opt_param** are true.
Type(rttov_opt_param)	Intent(inout), optional	aer_opt_param_ad	AD on aerosol optical properties. Optional even if **aer_opt_param** argument is present.
Type(rttov_opt_param)	Intent(in), optional	cld_opt_param	Cloud optical parameter input profiles if **opts%rt_ir%addclouds** and **opts%rt_ir%user_cld_opt_param** are true.
Type(rttov_opt_param)	Intent(inout), optional	cld_opt_param_ad	AD on cloud optical properties. Optional even if **cld_opt_param** argument is present.
Type(rttov_traj)	Intent(inout), optional	traj	Trajectory structure to hold temporary data.
Type(rttov_traj)	Intent(inout), optional	traj_ad	Trajectory structure to hold temporary data.
Type(rttov_pccomp)	Intent(inout), optional	pccomp	Structure to hold output PC scores and reconstructed radiances.
Type(rttov_pccomp)	Intent(inout), optional	pccomp_ad	Input PC or reconstructed radiance perturbations (see text).
Integer	Intent(in), optional	channels_rec(nchannelsrec)	Channels for which to compute reconstructed radiances if **opts%rt_ir%pc%addradrec** is .true.

Annex M – RTTOV_SCATT interface

```
call rttov_scatt (errorstatus, opts_scatt, nlevels, chanprof, frequencies,
                  profiles, cld_profiles, coef_rttov, coef_scatt, calcemis,
                  emissivity, radiance, cfrac, emis_retrieval_terms)
```

rttov_scatt should be called for each instrument required. Each call carries out computations for the profiles defined in the arrays **profiles(:)** and **cld_profiles(:)**. See **src/mw_scatt/example_rttovscatt.F90** and **src/test/example_rttovscatt_fwd.F90** for examples of calling **rttov_scatt**.

In RTTOV v12 it is possible to control a limited number of the internal RTTOV options via the **opts_scatt** parameter (see Annex O). The **lusercfrac** option has been moved into this structure. Also note that the optional switch to revert to the "old cloud scheme" has been removed.

Surface emissivities may now be supplied via the **emissivity** input variable of type **rttov_emissivity** (see Annex O) as for **rttov_direct**. This also passes the emissivities calculated by RTTOV back out via the **emis_out** member.

The **frequencies(:)** and **chanprof(:)** arrays should be populated by calling **rttov_scatt_setupindex** (Annex D) before calling **rttov_scatt**.

The **rttov_parallel_scatt** subroutine has the same arguments as **rttov_scatt** plus an optional final argument **nthreads** to specify the number of threads.

In the following table, **nchanprof** is the size of the **chanprof(:)** array, and **nprof** is the size of the **profiles(:)** array.

Type	In/Out	Variable	Description
Integer	Intent(out)	errorstatus	Return code.
Type(rttov_options_scatt)	Intent(in)	opts_scatt	Options to control aspects of RTTOV.
Integer	Intent(in)	nlevels	Number of input profile levels.
Type(rttov_chanprof)	Intent(in)	chanprof(nchanprof)	Chanprof structure.
Integer	Intent(in)	frequencies(nchanprof)	Frequency indices.
Type(rttov_profile)	Intent(in)	profiles(nprof)	Profiles structure.
Type(rttov_profile_cloud)	Intent(in)	cld_profiles(nprof)	Cloud profile structure.
Type(rttov_coefs)	Intent(in)	coef_rttov	RTTOV coefficient structure.
Type(rttov_scatt_coef)	Intent(in)	coef_scatt	RTTOV_SCATT coefficient structure.
Logical	Intent(in)	calcemis(nchanprof)	.true. if RTTOV should calculate surface emissivity or .false. if user is supplying an emissivity value (e.g. from the atlas).
Type(rttov_emissivity)	Intent(inout)	emissivity(nchanprof)	Input/output emissivities.
Type(rttov_radiance)	Intent(inout)	radiance	Output radiances (mW/cm^{-1}/sr/m^2 & degK).
Real	Intent(out), optional	cfrac(nprof)	Effective cloud fractions actually used (diagnostic).
Type(rttov_scatt_emis_retrieval_type)	Intent(inout), optional	emis_retrieval_terms	Output structure containing radiance and transmittance data which can be used for all-sky emissivity retrievals.

```
call rttov_scatt_tl (errorstatus, opts_scatt, nlevels, chanprof, frequencies,
                     profiles, cld_profiles, coef_rttov, coef_scatt, calcemis,
                     emissivity, profiles_tl, cld_profiles_tl, emissivity_tl,
                     radiance, radiance_tl)
```

rttov_scatt_tl should be called for each instrument required. Each call carries out computations for the profiles defined in the arrays **profiles(:)** and **cld_profiles(:)**. **rttov_scatt_tl** calculates the tangent linear of the RTTOV-SCATT direct model given a profile perturbation in **profiles_tl(:)** and **cld_profiles_tl(:)**. The emissivity TL values are treated as for **rttov_tl**. See section 7.9 for more information about the RTTOV TL model.

See the description of the **rttov_scatt** subroutine above for information on common arguments.

The **rttov_parallel_scatt_tl** subroutine has the same arguments as **rttov_scatt_tl** plus an optional final argument **nthreads** to specify the number of threads.

In the following table, **nchanprof** is the size of the **chanprof(:)** array, and **nprof** is the size of the **profiles(:)** array.

Type	In/Out	Variable	Description
Integer	Intent(out)	errorstatus	Return code.
Type(rttov_options_scatt)	Intent(in)	opts_scatt	Options to control aspects of RTTOV.
Integer	Intent(in)	nlevels	Number of input profile levels.
Type(rttov_chanprof)	Intent(in)	chanprof(nchanprof)	Chanprof structure.
Integer	Intent(in)	frequencies(nchanprof)	Frequency indices.
Type(rttov_profile)	Intent(in)	profiles(nprof)	Profiles structure.
Type(rttov_profile_cloud)	Intent(in)	cld_profiles(nprof)	Cloud profile structure.
Type(rttov_coefs)	Intent(in)	coef_rttov	RTTOV coefficient structure.
Type(rttov_scatt_coef)	Intent(in)	coef_scatt	RTTOV_SCATT coefficient structure.
Logical	Intent(in)	calcemis(nchanprof)	.true. if RTTOV should calculate surface emissivity or .false. if user is supplying an emissivity value (e.g. from the atlas).
Type(rttov_emissivity)	Intent(inout)	emissivity(nchanprof)	Input/output emissivities.
Type(rttov_profile)	Intent(in)	profiles_tl(nprof)	Input profile variable increments.
Type(rttov_profile_cloud)	Intent(in)	cld_profiles_tl(nprof)	Input cloud profile variable increments.
Type(rttov_emissivity)	Intent(inout)	emissivity_tl(nchanprof)	Input/output surface emissivity TL.
Type(rttov_radiance)	Intent(inout)	radiance	Output radiances (mW/cm^{-1}/sr/m^2 & degK).
Type(rttov_radiance)	Intent(inout)	radiance_tl	TL of radiances.

```
call rttov_scatt_ad (errorstatus, opts_scatt, nlevels, chanprof, frequencies,
                     profiles, cld_profiles, coef_rttov, coef_scatt, calcemis,
                     emissivity, profiles_ad, cld_profiles_ad, emissivity_ad,
                     radiance, radiance_ad)
```

rttov_scatt_ad should be called for each instrument required. Each call carries out computations for the profiles defined in the arrays **profiles(:)** and **cld_profiles(:)**.

rttov_scatt_ad calculates the adjoint or the Jacobian of the RTTOV direct model, the output being written to **profiles_ad** and **cld_profiles_ad**. In order to run the adjoint model the **profiles_ad(:)** and **cld_profiles_ad(:)** arrays should be the same size as the **profiles(:)** array (i.e. the number of profiles being simulated). To run the Jacobian model the **profiles_ad(:)** and **cld_profiles_ad(:)** arrays should be the same size as the **chanprof(:)** array (i.e. the total number of channels being simulated over all profiles). In either case **profiles_ad(:)** and **cld_profiles_ad(:)** must be the same size.

The input perturbation to the RTTOV-SCATT AD/K models is always in brightness temperature and should usually be supplied in the **radiance_ad%bt(:)** array. All other AD/K arguments should be initialised to zero before calling **rttov_scatt_ad**. The **emissivity_ad** argument is the same as for **rttov_ad/rttov_k**. See section 7.9 for more information about the RTTOV AD and K models.

See the description of the **rttov_scatt** subroutine above for information on common arguments.

The **rttov_parallel_scatt_ad** subroutine has the same arguments as **rttov_scatt_ad** plus an optional final argument **nthreads** to specify the number of threads.

In the following table, **nchanprof** is the size of the **chanprof(:)** array, and **nprof** is the size of the **profiles(:)** array.

Type	In/Out	Variable	Description
Integer	Intent(out)	errorstatus	Return code.
Type(rttov_options_scatt)	Intent(in)	opts_scatt	Options to control aspects of RTTOV.
Integer	Intent(in)	nlevels	Number of input profile levels.
Type(rttov_chanprof)	Intent(in)	chanprof(nchanprof)	Chanprof structure.
Integer	Intent(in)	frequencies(nchanprof)	Frequency indices.
Type(rttov_profile)	Intent(in)	profiles(nprof)	Profiles structure.
Type(rttov_profile_cloud)	Intent(in)	cld_profiles(nprof)	Cloud profile structure.
Type(rttov_coefs)	Intent(in)	coef_rttov	RTTOV coefficient structure.
Type(rttov_scatt_coef)	Intent(in)	coef_scatt	RTTOV_SCATT coefficient structure.
Logical	Intent(in)	calcemis(nchanprof)	.true. if RTTOV should calculate surface emissivity or .false. if user is supplying an emissivity value (e.g. from the atlas).
Type(rttov_emissivity)	Intent(inout)	emissivity(nchanprof)	Input/output emissivities.
Type(rttov_profile)	Intent(inout)	profiles_ad(nprof) OR profiles_ad(nchanprof)	AD or Jacobian on profile variables.
Type(rttov_profile_cloud)	Intent(inout)	cld_profiles_ad(nprof) OR cld_profiles_ad(nchanprof)	AD or Jacobian on cloud profile variables.
Type(rttov_emissivity)	Intent(inout)	emissivity_ad(nchanprof)	AD or Jacobian on surface emissivity.
Type(rttov_radiance)	Intent(inout)	radiance	Output radiances (mW/cm^{-1}/sr/m^2 & degK).
Type(rttov_radiance)	Intent(inout)	radiance_ad	Input BT perturbations (see text).

Annex N – RTTOV Utility routines

1. RTTOV_USER_OPTIONS_CHECKINPUT interface

```
call rttov_user_options_checkinput(err, opts, coefs)
```

This subroutine checks that the input options are consistent with one another and with the supplied coefficient structure. The routine will set **err** to **errorstatus_fatal** and print an error message if any inconsistencies are found. This can be useful for debugging problems.

The source can be found in the **src/main/** directory.

Type	In/Out	Variable	Description
Integer	Intent(out)	err	Return code.
Type(rttov_options)	Intent(in)	opts	Options structure.
Type(rttov_coefs)	Intent(in)	coefs	RTTOV coefficients structure.

2. RTTOV_USER_PROFILE_CHECKINPUT interface

```
call rttov_user_profile_checkinput(err, opts, coefs, prof, aer_opt_param,
                                   cld_opt_param, reg_limits_exceeded)
```

This subroutine checks profiles on user levels against the regression limits. You may wish to use this to check profiles for unphysical or out-of-specification values before calling RTTOV. If you are supplying scattering optical parameters for aerosol or cloud simulations, you can optionally pass in these structures as well. In this case, **opts%config%do_checkinput** may be set to false so that input profiles are not checked again within the call to RTTOV.

Only one profile is checked at a time. The value of **err** is **errorstatus_fatal** if any unphysical values are found in the profile. If the additional logical flag **reg_limits_exceeded** is passed in, it is set to true if any profile variable exceeds the RTTOV regression limits. See section 7.3 for details on the implications of this. Comparisons are made using the values on the nearest coefficient pressure level below the input profile pressure level.

The source can be found in the **src/main/** directory.

Type	In/Out	Variable	Description
Integer	Intent(out)	err	Return code.
Type(rttov_options)	Intent(in)	opts	Options structure.
Type(rttov_coefs)	Intent(in)	coefs	RTTOV coefficients structure.
Type(rttov_profile)	Intent(in)	prof	Profile structure.
Type(rttov_opt_param)	Intent(in), optional	aer_opt_param	Aerosol optical parameter profiles.
Type(rttov_opt_param)	Intent(in), optional	cld_opt_param	Cloud optical parameter profiles.
Logical	Intent(out), optional	reg_limits_exceeded	Set to true if any profile variable exceeds the RTTOV regression limits

3. RTTOV_PRINT_OPTS interface

`call` **`rttov_print_opts`** `(opts, lu, text)`

This subroutine prints out the contents of the options structure to the selected logical unit (or to **error_unit** if the **lu** argument is omitted). The ability to see the option values being input to RTTOV can be useful for debugging problems.

The source can be found in the **src/other/** directory.

Type	In/Out	Variable	Description
Type(rttov_options)	Intent(in)	opts	Options structure.
Integer	Intent(in), optional	lu	Logical unit for output (defaults to error_unit).
Character	Intent(in), optional	text	Additional text to print out.

4. RTTOV_PRINT_OPTS_SCATT interface

`call` **`rttov_print_opts_scatt`** `(opts_scatt, lu, text)`

This subroutine prints out the contents of the RTTOV-SCATT options structure to the selected logical unit (or to **error_unit** if the **lu** argument is omitted). The ability to see the option values being input to RTTOV-SCATT can be useful for debugging problems.

The source can be found in the **src/other/** directory.

Type	In/Out	Variable	Description
Type(rttov_options_scatt)	Intent(in)	opts_scatt	Options structure.
Integer	Intent(in), optional	lu	Logical unit for output (defaults to error_unit).
Character	Intent(in), optional	text	Additional text to print out.

5. RTTOV_PRINT_INFO interface

`call` **`rttov_print_info`** `(coefs, lu, text)`

This subroutine prints out some information about RTTOV (the library version number, and largest integer and real values allowed). If a coefficient structure is supplied, information about this is also displayed. Output is printed to the supplied logical unit (or to **error_unit** if the **lu** argument is omitted). The ability to see information about the coefficient file being input to RTTOV can be useful for debugging problems.

The source can be found in the **src/other/** directory.

Type	In/Out	Variable	Description
Type(rttov_coefs)	Intent(in), optional	coefs	Coefficients structure.
Integer	Intent(in), optional	lu	Logical unit for output (defaults to error_unit).
Character	Intent(in), optional	text	Additional text to print out.

6. RTTOV_PRINT_PROFILE interface

`call` **`rttov_print_profile`** `(profile, lu, text)`

This subroutine prints out the contents of the profile structure to the selected logical unit (or to **error_unit** if the **lu** argument is omitted). The ability to see the profile values being input to RTTOV can be useful for debugging problems.

The source can be found in the **src/other/** directory.

Type	In/Out	Variable	Description
Type(rttov_profile)	Intent(in)	profile	Profile structure.
Integer	Intent(in), optional	lu	Logical unit for output (defaults to error_unit).
Character	Intent(in), optional	text	Additional text to print out.

7. RTTOV_PRINT_CLD_PROFILE interface

`call` **`rttov_print_cld_profile`** `(cld_profile, lu, text)`

This subroutine prints out the contents of the RTTOV-SCATT cloudy profile structure to the selected logical unit (or to **error_unit** if the **lu** argument is omitted). The ability to see the profile values being input to RTTOV-SCATT can be useful for debugging problems.

The source can be found in the **src/other/** directory.

Type	In/Out	Variable	Description
Type(rttov_profile_cloud)	Intent(in)	cld_profile	Cloud profile structure.
Integer	Intent(in), optional	lu	Logical unit for output (defaults to error_unit).
Character	Intent(in), optional	text	Additional text to print out.

8. RTTOV_PRINT_RADIANCE_QUALITY interface

`call` **`rttov_print_radiance_quality`** `(quality, lu, text)`

This subroutine prints out a human-readable interpretation of a single **radiance%quality** output. A non-zero value indicates a warning flag has been set for the given output radiance. The output is written to the selected logical unit (or to **error_unit** if the **lu** argument is omitted). This is intended for debugging purposes.

The source can be found in the **src/other/** directory.

Type	In/Out	Variable	Description
Integer	Intent(in)	quality	Radiance quality output.
Integer	Intent(in), optional	lu	Logical unit for output (defaults to error_unit).
Character	Intent(in), optional	text	Additional text to print out.

9. CREATE_AER_CLIM_PROF.EXE

This executable may be used to generate profiles on the layers defined by an arbitrary set of pressure levels for the first 10 aerosol types in the RTTOV OPAC aerosol property files (i.e. excluding the volcanic ash and Asian dust types) for 10 different climatological compositions. The source code can be found in **src/other/**.

The routine requires files in the current directory specifying the pressure profile (**plevs.dat**), and the temperature and water vapour profiles (**prof.dat**). The units of water vapour are ppmv. Example input files can be found in the **data/** directory of the RTTOV distribution.

It is perhaps easiest to run **create_aer_clim_prof.exe** from the **data/** directory. The program prompts you for a latitude, an elevation (km), the level of the surface (where the lowest level is 1 and the top of the atmosphere is **nlev**), and a scale factor. The calculated profiles are multiplied by the scale factor.

The output is written to the file **prof_aerosl_cl.dat** in the current directory and consists of 10 columns, one for each of the RTTOV aerosol types (excluding volcanic ash). Each column contains 10 consecutive profiles for the following climatological compositions:

1 Continental clean
2 Continental average
3 Continental polluted
4 Urban
5 Desert
6 Maritime clean
7 Maritime polluted
8 Maritime tropical
9 Arctic
10 Antarctic

Note the units are number densities (cm^{-3}) so you must set the RTTOV **profiles(:)%mmr_cldaer** variable to false when passing these profiles intro RTTOV.

10. RTTOV_AER_CLIM_PROF

This subroutine is called by **create_aer_clim_prof.exe** to generate the aerosol profiles. You may wish to call this subroutine directly in your own code: this is done in **src/test/example_aer_file_fwd.F90**.

```
call rttov_aer_clim_prof (p, t, q, levsurf, latitude, elevation, &
                          scalefactor, aerprof)
```

Type	In/Out	Variable	Description
Real	Intent(in)	p(nlevels)	Pressure profile (hPa).
Real	Intent(in)	t(nlevels)	Temperature profile (K).
Real	Intent(in)	q(nlevels)	Water vapour profile (units: see **gas_units** argument).
Integer	Intent(in)	gas_units	Units for water vapour: 0=>ppmv over dry air; 1=>kg/kg over moist air; 2=>ppmv over moist air
Logical	Intent(in)	mmr_aer	Output aerosol units: true => kg/kg, false => cm^{-3}
Integer	Intent(in)	levsurf	Index of level nearest the surface.
Real	Intent(in)	latitude	Latitude of profile (degrees).
Real	Intent(in)	elevation	Elevation of profile (km)
Real	Intent(in)	scalefactor	Output profiles are multiplied by this scale factor.
Real	Intent(out)	aerprof(nlayers,10,13)	Output aerosol number densities on layers for each of the 10 compositions listed above, and for each of the 13 RTTOV OPAC aerosol types (units determined by **mmr_aer** argument)

11. RTTOV_ZUTILITY

The module **src/other/rttov_zutility.F90** may be used to obtain values for the magnetic field strength and orientation for use with the Zeeman coefficient files.

The look-up table (LUT) must first be loaded with the following function:

```
errorstatus = load_bfield_lut(filename_LUT)
```

One LUT is supplied with RTTOV v11 in **data/Be_LUT.2007.txt**. The LUT is stored in the **rttov_zutility** module in the array **BField**. To return the field strength and orientation there are three options:

```
call compute_bfield(latitude, longitude,
                    Bx, By, Bz, Be)

call compute_bfield(latitude, longitude, sensor_zenang, sensor_aziang,
                    Be, cos_bkang, cos_baziang)

call compute_bfield(latitude, longitude, sensor_zenang, sensor_relative_aziang,
                    Julian_day, utc_time, Be, cos_bkang, cos_baziang)
```

Finally, a subroutine is available to return the field orientation:

```
call compute_kb_angles(Bx, By, Bz, sensor_zenang, sensor_aziang,
                       cos_bkang, cos_baziang)
```

The arguments to these routines are detailed in the table below.

Type	In/Out	Variable	Description
Real	Intent(in)	latitude	Latitude of location (-90 to +90)
Real	Intent(in)	longitude	Longitude of location (accepts 0 to 360 or -180 to 180)
Real	Intent(in)	sensor_zenang	Sensor zenith angle
Real	Intent(in)	sensor_aziang	Sensor azimuth angle (0 to 360, North=0, positive clockwise)
Real	Intent(in)	sensor_relative_aziang	Solar azimuth angle minus sensor azimuth angle
Integer	Intent(in)	Julian_day	Julian day 1=Jan 1, 365=Dec 31 (366 leap year)
Real	Intent(in)	utc_time	Universal time 0.00-23.999 (GMT, Z time)
Real	Intent(out)	Bx, By, Bz	Magnetic field components: east, north and zenith (positive upwards) respectively. Units: Gauss.
Real	Intent(out)	Be	Magnetic field strength. Units: Gauss.
Real	Intent(out)	cos_bkang	Cosine of the angle between the magnetic field Be vector and the wave propagation direction k.
Real	Intent(out)	cos_baziang	Cosine of the azimuth angle of the Be vector in the (v, h, k) coordinates system, where v, h and k comprise a right-hand orthogonal system, similar to the (x, y, z) Cartesian coordinates. The h vector is normal to the plane containing the k and z vectors, where k points to the wave propagation direction and z points to the zenith. h = (z cross k)/|z cross k|. The azimuth angle is the angle on the (v, h) plane from the positive v axis to the projected line of the Be vector on this plane, positive counter-clockwise.

12. RTTOV_OBS_TO_PC.EXE

The program **rttov_obs_to_pc.exe** (located in the **bin/** directory, source code is in **src/other/rttov_obs_to_pc.F90**) demonstrates how to convert radiance observations into PC-space which is necessary, for example, in applications involving the assimilation of PCs.

The usage of the example executable is as follows:

```
$ rttov_obs_to_pc.exe \
    --rtcoef_file ... \
    --pccoef_file ... \
    --obs_file ... \
    --ipcbnd ... \
    --ipcreg ... \
    --npcscores ... \
    --obs_bt
```

This program reads in a set of observations from the specified "obs_file" and calculates the required number of PC scores based on the specified input coefficient files and the regression band and predictor sets (see section 8.8 for information on PC simulations). PC scores are written to standard out.

You would typically need to modify the source code to suit your own application.

Argument	Description
--rtcoef_file	Optical depth coefficient file.
--pccoef_file	PC coefficient file.
--obs_file	ASCII file containing white-space-separated observation values. Units are Kelvin if --obs_bt is present, otherwise mW/m-2/sr-1/cm-1.
--ipcbnd	PC band to use.
--ipcreg	PC regression predictor set to use.
--npcscores	The number of PC scores to calculate.
--obs_bt	Optional: if present, input observations are BTs, otherwise they are radiances (see --obs_file).

13. RTTOV_SCATT_EMIS_RETRIEVAL

After calling **rttov_scatt** with the optional **emis_retrieval_terms** argument, this subroutine can be used to carry out the emissivity retrieval calculation given the corresponding observed brightness temperatures.

```
call rttov_scatt_emis_retrieval(chanprof, coef, emis_terms, obs_tb, land_emis)
```

Type	In/Out	Variable	Description
Type(rttov_chanprof)	Intent(in)	chanprof(nchanprof)	RTTOV chanprof structure as used in call to **rttov_scatt**.
Type(rttov_coef)	Intent(in)	coefs	RTTOV coefficients structure.
Type(rttov_scatt_emis_retrieval_type)	Intent(in)	emis_terms	Emissivity retrieval terms structure populated by a call to **rttov_scatt**.
Real	Intent(in)	obs_tb(nchanprof)	Observed brightness temperatures corresponding to simulated brightness temperatures (K).
Real	Intent(out)	land_emis(nchanprof)	Retrieved emissivities.

14. RTTOV_MAKE_SCAERCOEF.EXE

This executable can be used to generate custom aerosol optical property files (*scaercoef*) containing an arbitrary number of species for visible/IR scattering simulations. The primary input data required are refractive indices over a range of wavelengths covering the sensor channels and the size distribution. Aerosols may be hydrophobic or hydrophilic: in the latter case, the refractive indices and size distribution data must be supplied for each of several (user-specified) relative humidity values. The executable is controlled by a namelist file which defines the optical properties and specifies paths to files containing the refractive index and size distribution data in simple ASCII format. The tool is fully described in a separate user guide: **docs/readme_rttov_make_scaercoef.txt** and there are example input files in the **data/** directory.

15. RTTOV_SCALE_REF_GAS_PROF

This subroutine is intended to make it easy to pass scaled copies of the RTTOV reference trace gas profiles into RTTOV simulations. A typical use case would be in simulating an older instrument for which you do not have an explicit CO2 profile, but for which the RTTOV reference profile is inappropriate because the concentrations are contemporary (~400ppmv). You first set up your simulation to specify all trace gases you wish to include: for example, by setting **opts%rt_ir%co2_data** to true. You then populate the **profiles** structure with all of the profile data as usual except for the CO2 profiles. Finally, before the call to RTTOV, you call **rttov_scale_ref_gas_prof** passing in the **profiles(:)** and, for example, the maximum CO2 concentration in ppmv. The subroutine populates the **co2(:)** member of each element of **profiles(:)** with the RTTOV reference (background) CO2 profile scaled to have the maximum CO2 value you specified. The subroutine may be used to populate one or more of the optional trace gas species in a single call, and the scale factors are determined either by providing maximum values in ppmv over dry air, or by providing column-integrated amounts in kg/m^2 (or, for ozone only, a column-integrated amount in Dobson units). The column-integration is calculated as the sum of the layer delta-pressure multiplied by the layer average gas concentration in kg/kg over moist air divided by the acceleration due to gravity. The subroutine handles any value of **gas_units** in the input profiles. The coefficient file must support the variable gases being requested via the arguments. This subroutine can also be used with HTFRTC in exactly the same way as for standard RTTOV.

```
call rttov_scale_ref_gas_prof(err, coefs, profiles, &
                              o3_col_int,  o3_col_int_du, o3_max_ppmv, &
                              co2_col_int, co2_max_ppmv, &
                              n2o_col_int, n2o_max_ppmv, &
                              co_col_int,  co_max_ppmv,  &
                              ch4_col_int, ch4_max_ppmv, &
                              so2_col_int, so2_max_ppmv)
```

Type	In/Out	Variable	Description
Integer	Intent(out)	err	Return code.
Type(rttov_coef)	Intent(in)	coefs	RTTOV coefficients structure.
Type(rttov_profiles)	Intent(inout)	profiles(nprofiles)	RTTOV profiles structures populated with all data except the gas profiles in question.
Real	Intent(in), optional	o3_col_int o3_col_int_du o3_max_ppmv	Specify one of these (column integrated amount kg/m^2, column integrated amount in Dobson units, or maxmimum amount in ppmv over air) to populate profiles with suitably-scaled copies of the RTTOV reference ozone profile.
Real	Intent(in), optional	co2_col_int co2_max_ppmv	Specify one of these (column integrated amount kg/m^2, or maxmimum amount in ppmv over air) to populate profiles with suitably-scaled copies of the RTTOV reference CO2 profile.
Real	Intent(in), optional	n2o_col_int n2o_max_ppmv	Specify one of these (column integrated amount kg/m^2, or maxmimum amount in ppmv over air) to populate profiles with suitably-scaled copies of the RTTOV reference N2O profile.
Real	Intent(in), optional	*_col_int *_max_ppmv	… similarly for the other trace gases …

Annex O – RTTOV v12 derived types

RTTOV's derived types are defined in the module **rttov_types.F90**. The derived types which are required in your code which calls RTTOV are described here.

Options structure

The **rttov_options** structure holds switches which configure various aspects of RTTOV. The first step in running RTTOV is to declare an instance of this structure and to set the members to appropriate values. The options are held in several sub-types which group them by function.

Type	Variable	Description
General configuration options: ***opts % config***		
Logical	opts%config%**do_checkinput**	If true checks whether input profiles are within both absolute and regression limits. If false no check is performed (default = true)
Logical	opts%config%**apply_reg_limits**	If true input profiles outside the limits specified in the coefficient files are reset to the min/max. If false such profiles will generate warning messages unless *verbose* is false (default = false).
Logical	opts%config%**verbose**	If false only messages for fatal errors are output (default = true).
Logical	opts%config%**fix_hgpl**	If false the input surface elevation is assigned to the pressure level at or immediately below the input 2m (surface) pressure. If true the surface elevation is assigned to the specified surface pressure (default = false).

General radiative transfer options: ***opts % rt_all***		
Logical	opts%rt_all%**switchrad**	Determines input perturbation for AD/K routines: if true **radiance_ad/k % bt** is used for channels with wavelengths > 3µm, otherwise **radiance_ad/k % total** is used (default = false)
Logical	opts%rt_all%**do_lambertian**	If true activate treatment of surface as Lambertian instead of specular reflector for downwelling emitted radiance (default = false). The **specularity** profile variable determines how specular or Lambertian the surface is.
Logical	opts%rt_all%**lambertian_fixed_ angle**	If true the Lambertian downwelling radiance is computed using a fixed effective angle. If false the angle is derived from a parameterisation computed from the total atmospheric optical depth (default = true). Only relevant if **do_lambertian** is true.
Logical	opts%rt_all%**rad_down_lin_tau**	If true the downwelling atmospheric emission is computed using the linear-in-tau approximation for the Planck source term. If false a simpler layer average source term is used which is slightly faster and has negligible impact on radiances (default = true, recommended = false). Note that the Lambertian downwelling radiances *always* use the layer average source term.
Logical	opts%rt_all%**use_q2m**	If true activate use of surface humidity (default = true)
Logical	opts%rt_all%**addrefrac**	If true RTTOV calculations account for atmospheric refraction (default = false)
Logical	opts%rt_all%**plane_parallel**	If true treat atmosphere as strictly plane-parallel (no curvature or refraction); automatically activated for **all** simulated radiances if the DOM scattering solver is used (default = false)

MW-only radiative transfer options: ***opts % rt_mw***		
Logical	opts%rt_mw%**apply_band_correction**	Selects whether to apply or ignore any band correction coefficients present in the coefficient file. These only have a significant impact on radiances, not brightness temperatures (default = true, recommended = true).
Logical	opts%rt_mw%**clw_data**	If true user is supplying cloud liquid water profiles (default = false). NB This applies to "clear-sky" simulations only: the cloud

		is treated as a purely absorbing medium. For full scattering use RTTOV-SCATT instead.
Logical	opts%rt_mw%**clw_calc_on_coef_lev**	If true the CLW optical depths are computed on coefficient levels, if false they are computed on input levels (default = true, recommended = false)
Integer	opts%rt_mw%**clw_scheme**	Choose CLW permittivity parameterisation: 1=>Liebe (1989), 2=>Rosekranz (2015), 3=>Turner,Kneifel,Cadeddu(2016) (default = 1/Liebe, recommended = 2/Rosenkranz)
Real	opts%rt_mw%**clw_cloud_top**	Lower pressure limit for MW CLW calculations (hPa): any CLW in levels with pressures smaller than this value is ignored (default = 322 hPa).
Integer	opts%rt_mw%**fastem_version**	Select the MW sea surface emissivity model to use. Valid values are 0-6; values 1-6 specify FASTEM-1 to FASTEM-6; value 0 specifies TESSEM2 (default = 6).
Logical	opts%rt_mw%**supply_foam_fraction**	If true use the foam fraction value specified in **profiles(:)%skin%foam_fraction** in FASTEM (default = false). Does not apply to TESSEM2.

Visible/IR-only radiative transfer options: ***opts % rt_ir***		
Logical	opts%rt_ir%**ozone_data**	If true user is supplying ozone profiles (default = false)
Logical	opts%rt_ir%**co2_data**	If true user is supplying CO_2 profiles (default = false)
Logical	opts%rt_ir%**n2o_data**	If true user is supplying N_2O profiles (default = false)
Logical	opts%rt_ir%**co_data**	If true user is supplying CO profiles (default = false)
Logical	opts%rt_ir%**ch4_data**	If true user is supplying CH_4 profiles (default = false)
Logical	opts%rt_ir%**so2_data**	If true user is supplying SO_2 profiles (default = false)
Logical	opts%rt_ir%**addsolar**	If true enable solar calculations for solar-affected channels (default = false)
Logical	opts%rt_ir%**rayleigh_single_scatt**	If false disables the Rayleigh single-scattering calculation for visible channels when **addsolar** is true. Note that Rayleigh extinction is included in the LBL simulations used to train the optical depth coefficients so this is always included (default = true, not recommended to set this to false without good reason).
Logical	opts%rt_ir%**do_nlte_correction**	If true includes non-LTE bias correction for hi-res sounders (default = false) NB this is independent of **addsolar**.
Integer	opts%rt_ir%**solar_sea_brdf_model**	Select the solar sea BRDF model: 1=>JONSWAP, 2=>Elfouhaily et al. (default = 1, recommended = 2).
Integer	opts%rt_ir%**ir_sea_emis_model**	Select the IR sea surface emissivity model to use. Valid values: 1 => ISEM; 2 => IREMIS (default and recommended = 2).
Logical	opts%rt_ir%**addaerosl**	If true account for scattering due to aerosols (default = false)
Logical	opts%rt_ir%**addclouds**	If true account for scattering due to clouds (default = false)
Logical	opts%rt_ir%**user_aer_opt_param**	If true and **addaerosl** is true the user specifies the aerosol scattering optical parameters instead of supplying number density profiles for pre-defined particle types (default = false).
Logical	opts%rt_ir%**user_cld_opt_param**	If true and **addclouds** is true the user specifies the cloud scattering optical parameters instead of supplying concentration profiles for pre-defined particle types (default = false).
Logical	opts%rt_ir%**grid_box_avg_cloud**	If true input cloud concentrations should represent grid box average values (i.e. not accounting for cloud fraction). If false the cloud concentration should represent the value for the cloudy fraction of the layer (i.e. the grid box average divided by the layer cloud fraction). Only applies when **addclouds** is true and **user_cld_opt_param** is false (default = false)
Real	opts%rt_ir%**cldstr_threshold**	Threshold for cloud stream (column) weights to compute for cloud scattering calculations (default = -1); recommended to be set negative when calling TL, AD or K models. Only applies when **addclouds** is true.

Logical	opts%rt_ir%**cldstr_simple**	If true, a simplified cloud stream approach is used with one clear and one cloudy column. The "effective" cloud fraction for the profile is taken as the maximum input cloud fraction in the profile layers from the ToA down to the pressure specified by **cldstr_low_cloud_top**. Use with caution: this is intended for mid- to upper-tropospheric channels (default = false). Only applies when **addclouds** is true.
Real	opts%rt_ir%**cldstr_low_cloud_top**	Only applies when **cldstr_simple** is true: cloud fraction is computed as maximum in layers from ToA down to the specified pressure in hPa (default = 750 hPa). Use with caution.
Integer	opts%rt_ir%**ir_scatt_model**	Scattering model to use for thermal emission source term: 1 => DOM; 2 => Chou-scaling (default = 2); only applies when **addclouds** or **addaerosl** is true.
Integer	opts%rt_ir%**vis_scatt_model**	Scattering model to use for solar source term: 1 => DOM; 2 => single-scattering (default = 1); only applies when **addclouds** or **addaerosl** is true and **addsolar** is true.
Integer	opts%rt_ir%**dom_nstreams**	Number of streams (discrete ordinates) to use with DOM scattering solver. Must be >=2 and even (default = 8); only applies when **addclouds** or **addaerosl** is true and DOM is selected as a scattering solver.
Real	opts%rt_ir%**dom_accuracy**	Parameter to determine convergence criterion for DOM azimuthal loop for solar scattering simulations. If zero or less the loop never exits early (default = 0.); only applies when **addclouds** or **addaerosl** is true and DOM is selected as a scattering solver.
Real	opts%rt_ir%**dom_opdep_threshold**	DOM ignores layers below this total level-to-space absorption optical depth. If zero or less all layers are treated in solver (default = 0.); only applies when **addclouds** or **addaerosl** is true and DOM is selected as a scattering solver.

Principal Components-only radiative transfer options: ***opts % rt_ir % pc***		
Logical	opts%rt_ir%pc%**addpc**	If true carry out Principal Components calculations (default = false).
Integer	opts%rt_ir%pc%**npcscores**	Specifies the number of PC scores to use in PC-RTTOV calculations. If <=0 the number of PC scores is determined from the size of the arrays in the **pccomp** structure. The advantage of using this option is that you can allocate **pccomp** for the maximum number of profiles you want to pass to RTTOV, but then call RTTOV with any number of profiles up to this maximum (which was not possible before this option was introduced in RTTOV v12.3) (default = -1).
Logical	opts%rt_ir%pc%**addradrec**	If true the PC calculations will return reconstructed radiances as well as the PC scores (default = false)
Integer	opts%rt_ir%pc%**ipcbnd**	The index of the PC spectral band: this should be 1 for PC coefficients current as of v12.3
Integer	opts%rt_ir%pc%**ipcreg**	The index of the required set of PC predictors: 1-4 for IASI/IASI-NG (see table 29 for the number of predictors), and 1-3 for AIRS (for 200, 300 and 400 predictors).

Options related to interpolation and the vertical grid: ***opts % interpolation***		
Logical	opts%interpolation%**addinterp**	If true input profiles may be supplied on user-defined levels, and internal interpolation is used (default = false).
Integer	opts%interpolation%**interp_mode**	Set the interpolation mode (see Table 9 in section 7.3). Valid values are 1-5 (default = 1).
Logical	opts%interpolation%**reg_limit_extrap**	Extrapolate input profiles up to top coefficient level maintaining relative values with respect to regression limits – see section 7.3 (default = false, recommended = true); only applies if **addinterp**

		is true.
Logical	opts%interpolation%**lgradp**	Allow TL/AD/K of user pressure levels (default = false); only applies if **addinterp** is true.
Logical	opts%interpolation%**spacetop**	If true treat user's model top as space boundary (default = true, it is not recommended to set this false).

Options related to HTFRTC: ***opts % htfrtc_opts***		
Logical	opts%htfrtc_opts%**htfrtc**	Set to true to call HTFRTC (default = false)
Integer	opts%htfrtc_opts%**n_pc_in**	The number of Princpal Components to calculate. If <= 0 then HTFRTC uses the maximum number of PC scores allowed by the coefficient file (default = -1)
Logical	opts%htfrtc_opts%**reconstruct**	Set to true to enable computation of reconstructed radiances (default = false)
Logical	opts%htfrtc_opts%**simple_cloud**	Enable simple cloud scheme (using **cfraction** and **ctp** profile variables) in HTFRTC simulations (default = false)
Logical	opts%htfrtc_opts%**overcast**	Enable calculation of overcast radiances (analogous to the RTTOV overcast radiance outputs) in HTFRTC simulations (default = false)

Options structure for RTTOV_SCATT

The **rttov_options_scatt** type defined in **rttov_types.F90** gives control over some of the internal RTTOV options for MW scattering calculations. Note that RTTOV_SCATT purposefully does not make all core radiative transfer options available.

Type	Variable	Description
Logical	opts_scatt%config%**do_checkinput**	If true, checks whether input profiles are within both absolute and regression limits. If false, no check is performed (default = true).
Logical	opts_scatt%config%**apply_reg_limits**	If true, input profiles outside the limits specified in the coefficient files are reset to the min/max. If false, such profiles will generate warning messages unless **verbose** is false (default = false).
Logical	opts_scatt%config%**verbose**	If false, only messages for fatal errors are output (default = true)
Logical	opts_scatt%config%**fix_hgpl**	If false the input surface elevation is assigned to the pressure level at or immediately below the input 2m (surface) pressure. If true the surface elevation is assigned to the specified surface pressure (default = false).
Logical	opts_scatt%**rad_down_lin_tau**	Sets the **opts%rt_all%rad_down_lin_tau** option for the call to RTTOV (see above, default = true, recommended = false).
Logical	opts_scatt%**apply_band_correction**	Selects whether to apply or ignore any band correction coefficients present in the coefficient file. These only have a significant impact on radiances, not brightness temperatures (default = true, recommended = true).
Integer	opts_scatt%**interp_mode**	Set the interpolation mode (see Table 9 in section 7.3). Valid values are 1-5 (default = 1).
Logical	opts_scatt%**lgradp**	Enable TL/AD/K of user pressure levels in the call to RTTOV (default = false).
Integer	opts_scatt%**reg_limit_extrap**	Extrapolate input profiles up to top coefficient level maintaining relative values with respect to regression limits – see section 7.3 (default = false, recommended = true).
Integer	opts_scatt%**fastem_version**	Select the MW sea surface emissivity model to use. Valid values are 0-6; values 1-6 specify FASTEM-1 to FASTEM-6; value 0 specifies TESSEM2 (default = 6).
Logical	opts_scatt %**supply_foam_fraction**	If true use the foam fraction value specified in **profiles(:)%skin%foam_fraction** in FASTEM (does not apply to TESSEM2).
Logical	opts_scatt%**use_q2m**	If true, activate use of surface humidity (default = true)
Logical	opts_scatt%**lradiance**	If true do RT calculation in radiances instead of brightness

		temperatures (default = false, recommended = true).
Logical	opts_scatt%**lusercfrac**	If true, you should supply supply the effective cloud fraction in the **rttov_profile_cloud** (see below). If false, this is calculated internally in RTTOV-SCATT (default = false).
Real	opts_scatt%**cc_threshold**	If the effective cloud fraction for the profile is below this value, it is ignored and the simulation is clear-sky. The value must be in the range 0-1 (default = 0.05).

Profile structure

The **rttov_profile** structure is composed of the atmospheric part and two other structures for 2 meter air and skin surface. If you are not able to provide ozone, CO_2, etc profiles the flags **ozone_data**, **co2_data** and so on in the options structure should be set to false.

Type	Variable	Description
Surface skin – type *rttov_skin*		
Integer	surftype	0=land, 1=sea, 2=sea-ice
Integer	watertype	0=fresh water, 1=ocean water. *Used only for sea surface solar BRDF model and BRDF atlas.*
Real	t	Radiative skin temperature (K)
Real	salinity	Practical salinity unit %o – *FASTEM-4/5/6 only.*
Real	fastem(1:5)	Land/sea-ice surface parameters for FASTEM.
Real	soil_moisture	Soil moisture (m^3/m^3) – *not currently used.*
Real	snow_fraction	Surface snow coverage fraction (0-1). *Used only by IR emissivity atlas.*
Real	foam_fraction	Foam fraction (0-1) to use in FASTEM if **opts%rt_mw%supply_foam_fraction** is true. By default FASTEM calculates the foam fraction internally.
Real	specularity	When **opts%rt_all%do_lambertian** is true, the Lambertian and specular downwelling radiances are linearly combined with the specularity as the weight: a value of zero implies fully Lambertian, a value of one implies fully specular (valid range is 0-1).

Surface 2m – type *rttov_s2m*		
Real	t	Temperature (K)
Real	q	Water vapour (units as per *gas_units*) – *only used if **opts%rt_all%use_q2m** is true.*
Real	o	Ozone (units as per *gas_units*) – *not currently used.*
Real	p	Surface pressure (hPa)
Real	u	U 10m wind component (m/s) – *only used by IREMIS, FASTEM and TESSEM2 sea surface emissivity models and sea surface solar BRDF model.*
Real	v	V 10m wind component (m/s) – *only used by IREMIS, FASTEM and TESSEM2 sea surface emissivity models and sea surface solar BRDF model.*
Real	wfetc	Wind fetch (m) (length of water over which the wind has blown, typical value 100000m for open ocean) – *only used by sea surface solar BRDF model.*

Atmospheric profile – type *rttov_profile*		
Character (len=128)	id	User may give text ID to each profile.
Integer	date(3)	Year, month, day – *used by solar calculations to adjust the solar irradiance based on the time of year.*
Integer	time(3)	Hour, minute, second – *not currently used.*
Integer	nlevels	Number of atmospheric levels –*automatically populated when the profile structure is allocated*.
Integer	nlayers	Number of atmospheric layers (i.e. nlevels-1) – *automatically populated*

		when the profile structure is allocated.
Integer	gas_units	Units for gas abundances: *(must be the same for all profiles)* 2 => ppmv over moist air (default) 1=> kg/kg over moist air 0 (or less) => ppmv over dry air
Logical	mmr_cldaer	Units for clouds and aerosols: *(must be the same for all profiles)* True => kg/kg for cloud and aerosol (default) False => g/m^3 for cloud, cm^{-3} for aerosol
Real	p(nlevels)	Pressure (hPa)
Real	t(nlevels)	Temperature (K)
Real	q(nlevels)	Water vapour (units as per *gas_units*)
Real	o3(nlevels)	Ozone (units as per *gas_units*)
Real	co2(nlevels)	CO_2 (units as per *gas_units*)
Real	n2o(nlevels)	N_2O (units as per *gas_units*)
Real	co(nlevels)	CO (units as per *gas_units*)
Real	ch4(nlevels)	CH_4 (units as per *gas_units*)
Real	so2(nlevels)	SO_2 (units as per *gas_units*)
Real	clw(nlevels)	Cloud liquid water (kg/kg) – *MW only, treats cloud as absorbing medium only in "clear-sky"; not used with RTTOV-SCATT.*
Real	aerosols(naertyp,nlayers) *naertyp is the number of aerosol types defined in the scaercoef file*	Aerosols (units as per **mmr_cldaer** switch) – *visible/IR only.* Note: the number of aerosol types is defined in the *scaercoef* scattering coefficient file. Not used if **opts%rt_ir%user_aer_opt_param** is true.
Real	cloud(ncldtyp,nlayers) *ncldtyp=6: five water cloud types and ice cloud*	Cloud water/ice (units as per **mmr_cldaer** switch) – *visible/IR only.* Note: the number of water cloud types is defined in the *sccldcoef* scattering coefficient file. Not used if **opts%rt_ir%user_cld_opt_param** is true.
Real	cfrac(nlayers)	Cloud fractional cover (0-1) – *visible/IR only.*
Real	clwde(nlayers)	Cloud liquid water particle effective diameter (microns) – *this is mandatory if **clw_scheme=2**.* Not used if **opts%rt_ir%user_cld_opt_param** is true or if the OPAC clw scheme is used.
Real	icede(nlayers)	Ice particle effective diameter (microns) – *this is optional, where non-zero this value is used in preference to the parameterisation specified by **idg**.* Not used if **opts%rt_ir%user_cld_opt_param** is true or if the Baran ice scheme is used.
Integer	clw_scheme	Optical properties for cloud liquid water: 1=OPAC; 2="Deff" scheme. This is not used if **opts%rt_ir%user_cld_opt_param** is true.
Integer	ice_scheme	Optical properties for ice crystals: 1=SSEC; 2=Baran scheme. This is not used if **opts%rt_ir%user_cld_opt_param** is true.
Integer	idg	Scheme for ice water content to effective diameter, Dg 1=Ou and Liou; 2=Wyser et al (recommended); 3=Boudala et al; 4=McFarquhar et al. This is not used if **opts%rt_ir%user_cld_opt_param** is true or if the Baran ice scheme is used or in layers where **profiles(:)%icede(:)**>*0*.
Real	zenangle	Local satellite zenith angle (degrees), maximum valid value depends on coefficient file (see Table 4).
Real	azangle	Local satellite azimuth angle (0-360º; measured clockwise, east=90º, see Figure 4) –*used in solar simulations and by FASTEM.*
Real	sunzenangle	Local solar zenith angle (degrees), solar radiation only included up to 85º.
Real	sunazangle	Local solar azimuth angle (0-360º; measured clockwise, east=90 º, see Figure 4)
Real	elevation	Elevation (km)
Real	latitude	Latitude (deg) -90º to +90º
Real	longitude	Longitude (deg) 0-360º – *used only by emissivity and BRDF atlases.*
Real	Be	Earth magnetic field strength (Gauss) – *Zeeman only.*
Real	cosbk	Cosine of the angle between the Earth magnetic field and wave propagation direction – *Zeeman only.*
Real	ctp	Cloud top pressure (hPa) – *simple cloud scheme only.*
Real	cfraction	Cloud fraction (0 - 1), 1 for 100% cloud cover – *simple cloud scheme only.*

Profile structure for RTTOV_SCATT cloud/precipitation

The **rttov_profile_cloud** defined in **rttov_types.F90** is for the RTTOV_SCATT microwave scattering calculations.

Type	Profile variable	Contents
Integer	nlevels	number of atmospheric levels, which should match that supplied in the other input profiles – *automatically populated when the profile structure is allocated*.
Logical	use_totalice	logical flag to switch between using separate ice and snow, or total ice hydrometeor types – *automatically populated when the profile structure is allocated*.
Logical	mmr_snowrain	logical flag to select units for snow and rain hydrometeors: True => kg/kg (default); False => kg/m^2/s – *automatically populated when the profile structure is allocated*.
Real	cfrac	***Optional:*** *if* ***opts_scatt%lusercfrac****=true., supply the effective cloud fraction, C, here. This is normally calculated internally in RTTOV-SCATT*
Real	ph(:)	nlevels+1 of half-level pressures (hPa)
Real	cc(:)	nlevels of cloud cover (0-1)
Real	clw(:)	nlevels of cloud liquid water (kg/kg)
Real	ciw(:)	nlevels of cloud ice water (kg/kg)
Real	totalice(:)	nlevels of total ice (kg/kg)
Real	rain(:)	nlevels of rain flux (units depend on **mmr_snowrain**)
Real	sp(:)	nlevels of solid precipitation flux (units depend on **mmr_snowrain**)

Optical parameter structure

The **rttov_opt_param** structure is used to specify profiles of optical parameters for each channel for aerosol and cloud scattering simulations if the **opts%rt_ir%user_aer_opt_param** and/or the **opts%rt_ir%user_cld_opt_param** flags are true.

Type	Variable	Description
Real	abs(nlayers,nchanprof)	Absorption coefficients (km^{-1}). For each layer one value is provided for every channel being simulated corresponding to the chanprof structure.
Real	sca(nlayers,nchanprof)	Scattering coefficients (km^{-1})
Real	bpr(nlayers,nchanprof)	"*b*" parameters (no units): represents the fraction of backscattered radiation at each layer. The **rttov_bpr_calc** subroutine is provided to calculate these from the phase functions. Required by the Chou-scaling scattering model.
Real	phangle(:)	Angles over which the phase function is defined (degrees).
Real	pha(nphangle,nlayers,nchanprof)	Phase functions (no units, must integrate to 4π over all scattering angles): used for calculating the *b* parameter for all channels, and used in solar scattering calculations.
Integer	nmom	Number of Legendre coefficients for phase function expansions (excluding the first coefficient which is always unity); required by the DOM scattering solver, must be greater than or equal to **opts%rt_ir%dom_nstreams**.
Real	legcoef(1:nmom+1,nlayers,nchanprof)	Phase function Legendre coefficients. First coefficient is always 1. The **rttov_legcoef_calc** subroutine is provided to calculate the coefficients from the phase functions. Required by the DOM scattering solver.

Chanprof structure

The **rttov_chanprof** structure is used to specify the channel and profile indices for each call to RTTOV. An array should be declared of size equal to the total number of radiances to be computed (across all channels and profiles). Each element of the **chanprof(:)** array provides a channel index and a profile index. The array should be ordered so that all channels for the first profile are listed, followed by all channels for the second profile, and so on (see Table 13).

Type	Variable	Description
Integer	chan	Channel index.
Integer	prof	Profile index.

Emissivity structure

The **rttov_emissivity** structure is used to pass emissivity values into RTTOV (where **calcemis** is false) and to return the emissivity values used by RTTOV. An array should be declared of size equal to the total number of radiances to be computed (across all channels and profiles).

Type	Variable	Description
Real	emis_in	Input emissivity (only used if corresponding element of **calcemis** is false).
Real	emis_out	Output emissivity (same as **emis_in** where **calcemis** is false).

Reflectance structure

The **rttov_reflectance** structure is used to pass BRDF values into RTTOV (where **calcrefl** is false) and to return the BRDF values used by RTTOV. An array should be declared of size equal to the total number of radiances to be computed (across all channels and profiles). Note that **refl_cloud_top** is *not* an active variable in the TL/AD/K models.

Type	Variable	Description
Real	refl_in	Input BRDF (only used if corresponding element of **calcrefl** is false).
Real	refl_out	Output BRDF (same as **refl_in** where **calcrefl** is false).
Real	refl_cloud_top	Optionally specify BRDFs for cloud tops in the simple cloud scheme. Set to zero to use internal defaults.

Radiance structure

The **rttov_radiance** structure is composed of the output radiances in units of mW/cm^{-1}/sr/m^2 and the output brightness temperatures in degK for each channel. Single element arrays are of size **nchanprof** (i.e. the size of the **chanprof(:)** array), and arrays of 2 dimensions are of size (**nlayers**, **nchanprof**), where **nlayers** = **nlevels** - 1. Radiances, brightness temperatures and reflectances (for solar calculations) are computed.

Type	Variable	Description
Radiances - units of mW/cm^{-1}/sr/m^2		
Real	clear(nchanprof)	Clear sky top of atmosphere radiance output for each channel. This includes aerosol scattering if **opts%rt_ir%addaerosl** is true.
Real	total(nchanprof)	Clear+cloudy top of atmosphere radiance for given cloud top pressure and fraction for each channel for simple cloud scheme or fully cloudy radiance if **opts%rt_ir%addclouds** is true.
Real	cloudy(nchanprof)	Cloudy top of atmosphere radiance for 100% fraction for each channel at given cloud top pressure for simple cloud scheme or same as total if **opts%rt_ir%addclouds** is true.
Real	overcast(nlayer,nchanprof)	Level to space overcast radiance at the level bounding the bottom of each layer. For the layer **j** containing the surface (or the layer immediately above the pressure level on which the surface lies), **overcast(j,:)** contains the overcast radiance as if the cloud was at the surface pressure and 2m temperature. For IR channels (wavelengths > 3µm) this assumes an opaque black cloud. For solar channels with no thermally emitted component (wavelengths < 3µm), this consists of reflected solar radiation according to assumptions described in section 8.3. This is not calculated for PC-RTTOV or IR aerosol simulations or scattering simulations using the DOM solver.
Brightness temperatures – units of deg K.		
Real	bt(nchanprof)	BT equivalent to total (clear+cloudy) top of atmosphere radiance output for each channel
Real	bt_clear(nchanprof)	BT equivalent to clear top of atmosphere radiance output for each channel
Bi-directional reflectance factors (BRFs) – unitless		
Real	refl(nchanprof)	Reflectance (BRF) equivalent to total (clear+cloudy) top of atmosphere radiance output for each channel
Real	refl_clear(nchanprof)	Reflectance (BRF) equivalent to clear top of atmosphere radiance output for each channel
Additional outputs		
Logical	plane_parallel	This flag is set to true if RTTOV used strict plane-parallel geometry. This is automatically enforced for certain types of simulations.
Integer	quality(nchanprof)	This is a bit mask which flags possible issues with simulated radiances (see section 7.8 for the bit positions and their descriptions). If zero there are no problems.

Secondary radiance structure

The **rttov_radiance2** structure holds the "secondary" output radiances in units of mW/cm^{-1}/sr/m^2. These are only calculated within the direct model, and are not calculated for PC or IR aerosol simulations or scattering simulations using the DOM solver. They do not contain any solar contributions. One-dimensional arrays are of size **nchanprof** (i.e. the size of the **chanprof(:)** array), and arrays of 2 dimensions are of size (**nlayers**, **nchanprof**), where **nlayers** = **nlevels** - 1.

Similarly to **rad%overcast**, for the layer containing the surface (or the layer immediately above the pressure level on which the surface lies), the radiance quantities defined on layers (**up**, **down** and **surf**) contain the radiances from/to the surface pressure. For **surf**, this is evaluated at the surface skin temperature.

Type	Variable	Description
Radiances - units of mW/cm^{-1}/sr/m^2		
Real	upclear(nchanprof)	Clear sky upwelling radiance at top of atmosphere including surface emission term, but omitting downwelling reflected radiance term.
Real	dnclear(nchanprof)	Clear sky downwelling radiance at surface.
Real	refldnclear(nchanprof)	Reflected clear sky downwelling radiance contribution to top of atmosphere radiance.
Real	up(nlayer,nchanprof)	Summed upwelling atmospheric emission term at top of atmosphere for layers down to the level bounding the bottom of each layer.
Real	down(nlayer,nchanprof)	Summed downwelling atmospheric emission term at bottom of layer for layers down to the level bounding the bottom of each layer.
Real	surf(nlayer,nchanprof)	Radiance emitted by a black cloud at the level bounding the bottom of each layer; for the surface layer this is evaluated for the surface skin temperature.

Transmission structure

The **rttov_transmission** structure contains output transmittances. The transmittances are unitless and lie in the interval [0, 1]. There are separate outputs for "thermal" IR channels and solar-affected channels for the surface-satellite transmittances. This is because the solar optical depth calculation uses the "effective" path length (the combined path length along the sun-surface-satellite path) while the thermal optical depths are calculated using the path length along the surface-satellite path. In addition, some channels use Planck-weighted coefficients for the thermal component calculations and for these channels it is necessary also to calculate non-Planck-weighted transmittances for the solar calculations. Also note that for IR aerosol simulations the output transmittances are those which result from the Chou scaling scheme (if used), not the clear-sky transmittances.

Type	Variable	Description
Real	tau_total(nchanprof)	Transmittance from surface to top of atmosphere (TOA) along the satellite view path. Only populated for channels with a significant thermally emitted contribution.
Real	tau_levels(nlevel, nchanprof)	Transmittance from each standard pressure level to TOA along the satellite view path. Only populated for channels with a significant thermally emitted contribution.
Real	tausun_total_path2(nchanprof)	Transmittance for combined sun-surface-satellite path. Only populated for solar-affected channels.
Real	tausun_levels_path2(nlevel, nchanprof)	Transmittance from TOA to each standard pressure level to TOA along combined sun-surface-satellite path. Only populated for solar-affected channels.
Real	tausun_total_path1(nchanprof)	Transmittance from surface to TOA along the satellite view path. Only populated for solar-affected channels.
Real	tausun_levels_path1(nlevel, nchanprof)	Transmittance from each standard pressure level to TOA along the satellite view path. Only populated for solar-affected channels.

RTTOV-SCATT emissivity retrieval terms structure

The **rttov_scatt_emissivity_retrieval_type** structure holds radiance and transmittance outputs from RTTOV-SCATT which can be used for dynamic all-sky emissivity retrievals. The radiances are in units of mW/cm^{-1}/sr/m^2. These are only calculated within the RTTOV-SCATT direct model. All arrays are of size **nchanprof** (i.e. the size of the **chanprof(:)** array).

Type	Variable	Description
Real	cfrac(nchanprof)	RTTOV-SCATT effective cloud fraction (Tallsky = cfrac * Tcld + (1-cfrac) * Tclr
Real	bsfc(nchanprof)	Surface blackbody Planck radiance.
Real	tau_cld(nchanprof)	Along-path transmittance, surface to space (cloudy column).

Real	up_cld(nchanprof)	TOA upwelling cloudy radiance or TB from atmosphere (not inc. surface emission or reflection).
Real	down_cld(nchanprof)	Surface downwelling cloudy radiance or TB (inc. cosmic term).
Real	tau_clr(nchanprof)	Along-path transmittance, surface to space (clear column).
Real	up_clr(nchanprof)	TOA upwelling clear radiance or TB from atmosphere (not inc. surface emission or reflection).
Real	down_clr(nchanprof)	Surface downwelling clear radiance or TB (inc. cosmic term).

PCCOMP structure

The **rttov_pccomp** structure contains the results of the Principal Component calculations for PC-RTTOV or HTFRTC. You must specify the options you require for PC-RTTOV or HTFRTC in the options structure before calling the **rttov_alloc_pccomp** subroutine to ensure the correct arrays are allocated. The "***pcscores"** members have size equal to the number of principal components multiplied by the number of profiles. The output radiance and BT arrays are only allocated if **opts%rt_ir%pc%addradrec** or **opts%htfrtc_opts%reconstruct** is true. These are sized according to the number of channels for which reconstructed radiances are required multiplied by the number of profiles. Unless otherwise indicated, arrays are allocated for both PC-RTTOV and HTFRTC.

Type	Variable	Description
Real	pcscores(npcscores)	Computed PC scores, equal to **clear_pcscores** for HTFRTC.
Real	total_pcscores(npcscores)	Computed PC scores for cloudy radiances, equal to **clear_pcscores** if **simple_cloud** is false (HTFRTC only)
Real	clear_pcscores(npcscores)	Computed PC scores for clear-sky radiances (HTFRTC only)
Real	cloudy_pcscores(npcscores)	Computed PC scores for 100% cloudy radiances. Only allocated if **simple_cloud** is true. (HTFRTC only).
Real	overcast_pcscores(nlayers, npcscores)	Computed PC scores for level to space overcast radiances at the level bounding the bottom of each layer. For the layer **j** containing the surface (or the layer immediately above the pressure level on which the surface lies), **overcast(j,:)** contains the overcast radiance as if the cloud was at the surface pressure and 2m temperature. Only allocated if **overcast** is true (HTFRTC only).
Real	total_pccomp(nchannels_rec)	Radiances constructed using **pcscores** for PC-RTTOV. Radiances constructed using **total_pcscores** for HTFRTC.
Real	clear_pccomp(nchannels_rec)	Clear-sky radiances constructed using **clear_pcscores** (HTFRTC only).
Real	cloudy_pccomp(nchannels_rec)	100% cloudy radiances constructed using **cloudy_pcscores**, only allocated if **simple_cloud** is true (HTFRTC only)
Real	overcast_pccomp(nlayers, nchannels_rec)	Overcast radiances constructed using **overcast_pcscores**, only allocated if **overcast** is true (HTFRTC only)
Real	bt_pccomp(nchannels_rec)	BTs equivalent to **total_pccomp** reconstructed radiances.
Real	bt_clear_pccomp(nchannels_rec)	Clear-sky BTs equivalent to **clear_pccomp** reconstructed radiances (HTFRTC only).

End of User Guide