RTTOV v11 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 1 December, 2006, 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, KNMI and Météo France.

Change record			
Version	Date	Author / changed by	Remarks
0.1	26/11/12	JH	Initial draft
0.4	11/01/13	JH, DR, RS, PR	Beta release version
1.0	26/04/13	JH	Post-beta update.
1.1	17/05/13	JH	Post-DRI update.
1.2	24/05/13	JH	Update for changes to instrument IDs and names.
1.3	13/06/14	JH	Updates for v11.2.
1.4	25/09/15	JH	Updates for v11.3.

TABLE OF CONTENTS

1. Introduction and Scope

This document is structured as follows. Section 2 gives a broad overview of the RTTOV v11 fast radiative transfer model. Section 3 provides a very brief introduction to the key concepts involved in running RTTOV v11 and it is recommended to all users. Section 4 provides details of all changes since RTTOV v10: this is intended to be a reference for users upgrading from v10. 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 the user's application. Section 8 gives gives details of specific aspects of RTTOV: users should consult the parts of this section relevant to their application. Section 9 lists the limitations of RTTOV v11. 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 11 of the RTTOV code and all its sub-versions (11.x). The document will not be systematically updated due to a change or new coefficient tables but users will be notified by email of these changes or can check on the RTTOV v11 web site (URL given below). Changes to this document are occasionally made to improve it and the document version is given in the header. If you want to request a copy of the RTTOV v11 code, go to http://nwpsaf.eu/request_forms/request.html?deliverable=rttov and complete a licence agreement form on-line. You will then be emailed instructions to download the code.

RTTOV v11 is an evolution of RTTOV v10, adding many more features as documented here. The RTTOV v10 code is still available, but cannot be guaranteed to be upgraded for new instruments and capability. RTTOV v11 can read most existing v10 coefficient files and RTTOV v10 can read most new v11 optical depth coefficient files (more details are available in this document and on the v11 web page). Existing coefficient files for RTTOV v7, RTTOV v8 and RTTOV v9 will continue to be made available from the NWP SAF web site. Any users wishing to use these old coefficients with RTTOV v11 must convert the files to the v10/v11 format: see Annex A for information on how to do this. However, the latest v11 coefficients provided with the package and available via the RTTOV v11 web page are the recommended ones for RTTOV v11.

The RTTOV v11 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/deliverables/rtm/index.html in pdf format on the RTTOV v11 page. There is also a RTTOV v11 performance report which documents the run times of RTTOV v11 on a few platforms and compares these to the equivalent RTTOV v10 run times.

The NWP SAF web forum can be found here: http://www.nwpsaf.eu/forum/. 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 v11

This section gives a brief overview of the RTTOV v11 model. More details can be found in the references given in this section. RTTOV v11 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 v11 is the latest version. The model allows rapid simulations (~1 ms for 40 channel ATOVS on a desktop PC) of radiances for satellite 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 v11 is water vapour. Optionally ozone, carbon dioxide, nitrous oxide, methane and carbon monoxide can be variable with all other constituents assumed to be constant. The state vector for RTTOV v11 is given in Annex O. Not all parameters have to be supplied as RTTOV can assume default values. RTTOV v11 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 v11 is mainly based on the 91-level 83 diverse profile dataset from ECMWF analyses for temperature, water vapour and ozone. The limits are given in Table 1 and can be found in the coefficient files supplied. For other gases a range of profile datasets were used based on models and measurements and again the limits are documented in the header section of the relevant coefficient file. More details on

the profile datasets used for the different gases can be found in Matricardi (2008). Note that the majority of RTTOV v11 coefficient files are based on the 54 levels shown in Table 1.

The spectral range of the RTTOV v11 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 – 200 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, although this list will be updated as new sensors are launched. For the 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.

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 $\boldsymbol{\delta}\mathbf{y}$ for a change in any element of the state vector $\boldsymbol{\delta}\mathbf{x}$ assuming a linear relationship about a given atmospheric state $\mathbf{x_0}$:

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

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. Some more information on TL/AD and K codes is available at: http://cimss.ssec.wisc.edu/itwg/groups/rtwg/fastrt.html. For users who 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 plan. No changes to the optical depth predictors were made for RTTOV v11.

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 infrared cloudy and aerosol-affected radiances based on a scattering parameterisation (sections 8.5 and 8.6) and for cloud and precipitation affected microwave radiances (section 8.7).

Level Number	Pressure (hPa)	Tmax degK	Tmin degK	Qmax ppmv	Qmin ppmv	O3max ppmv	O3min ppmv	O3Ref ppmv
1	0.005	245.95	143.65	5.24	0.91	1.404	0.014	0.296
2	0.01	252.13	154.19	6.03	1.08	1.411	0.069	0.321
3	0.03	263.72	168.42	7.42	1.35	1.496	0.108	0.381
4	0.06	280.11	180.18	8.11	1.58	1.670	0.171	0.527
5	0.13	299.04	194.48	8.44	1.80	2.064	0.228	0.769
6	0.23	318.64	206.21	8.59	1.99	2.366	0.355	1.070
7	0.41	336.24	205.66	8.58	2.49	2.718	0.553	1.470
8	0.67	342.08	197.17	8.34	3.01	3.565	0.731	1.990
9	1.08	340.83	189.49	8.07	3.30	5.333	0.716	2.790
10	1.67	334.67	179.27	7.89	3.21	7.314	0.643	3.760
11	2.50	322.50	176.28	7.75	2.92	9.190	0.504	4.860
12	3.65	312.51	175.04	7.69	2.83	10.450	0.745	5.950
13	5.19	303.89	173.07	7.58	2.70	12.340	1.586	6.760
14	7.22	295.48	168.38	7.53	2.54	12.930	1.880	7.110
15	9.84	293.32	166.30	7.36	2.46	12.740	1.322	7.060
16	13.17	287.05	163.46	7.20	2.42	11.960	0.719	6.570
17	17.33	283.36	161.48	6.96	2.20	11.100	0.428	5.690
18	22.46	280.93	161.47	6.75	1.71	9.796	0.278	4.700
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	7.373	0.107	3.110
21	45.04	273.15	164.67	5.90	1.36	6.800	0.055	2.480
22	55.44	265.92	166.19	6.21	1.30	5.709	0.048	1.910
23	67.51	264.70	167.42	9.17	1.16	4.786	0.043	1.440
24	81.37	261.95	159.97	17.89	0.36	4.389	0.038	1.020
25	97.15	262.43	163.95	20.30	0.01	3.620	0.016	0.733
26	114.94	259.56	168.59	33.56	0.01	2.977	0.016	0.604
27	134.83	259.26	169.71	102.20	0.01	2.665	0.016	0.489
28	156.88	260.13	169.42	285.10	0.01	2.350	0.013	0.388
29	181.14	262.28	170.63	714.70	0.01	1.972	0.010	0.284
30	207.61	264.46	174.10	1464.00	0.01	1.481	0.013	0.198
31	236.28	270.10	177.12	2475.00	0.01	1.075	0.016	0.145
32	267.10	277.93	181.98	4381.00	0.01	0.774	0.015	0.110
33	300.00	285.17	184.76	6631.00	0.01	0.628	0.015	0.086
34	334.86	293.67	187.69	9450.00	1.29	0.550	0.016	0.073
35	371.55	300.13	190.34	12440.00	1.53	0.447	0.015	0.063
36	409.89	302.63	194.40	15470.00	2.12	0.361	0.015	0.057
37	449.67	304.43	198.46	18570.00	2.36	0.284	0.015	0.054
38	490.65	307.20	201.53	21690.00	2.91	0.247	0.015	0.052
39	532.58	312.17	202.74	24700.00	3.67	0.199	0.015	0.050
40	575.15	315.54	201.61	27480.00	3.81	0.191	0.012	0.050
41	618.07	318.26	189.96	30280.00	6.82	0.171	0.010	0.049
42	661.00	321.70	189.96	32800.00	6.07	0.128	0.009	0.048
43	703.59	327.95	189.96	35320.00	6.73	0.124	0.009	0.047
44	745.48	333.77	189.96	37690.00	8.72	0.117	0.008	0.046
45	786.33	336.46	189.96	39980.00	8.26	0.115	0.008	0.045
46	825.75	338.53	189.96	42190.00	7.87	0.113	0.008	0.043
47	863.40	342.56	189.96	44220.00	7.53	0.112	0.007	0.042
48	898.93	346.22	189.96	46270.00	7.23	0.108	0.006	0.040
49	931.99	349.24	189.96	47730.00	6.97	0.102	0.006	0.038
50	962.26	349.92	189.96	51260.00	6.75	0.099	0.006	0.034
51	989.45	350.08	189.96	49720.00	6.57	0.099	0.006	0.030
52	1013.29	350.08	189.96	47200.00	6.41	0.094	0.006	0.028
53	1033.54	350.08	189.96	47800.00	6.29	0.094	0.006	0.028
54	1050.00	350.08	189.96	47640.00	6.19	0.094	0.006	0.027

Table 1. Pressure levels adopted for RTTOV v11 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
NOAA¶	1	1 - 19
DMSP	2	8 - 19
Meteosat	3	1 - 7
GOES	4	4 - 16
GMS	5	5
FY2	6	2 - 4
TRMM	7	1
ERS	8	1 - 2
EOS	9	1 - 2
METOP	10	1 - 2
ENVISAT	11	1
MSG	12	1 - 4
FY1	13	3 - 4
ADEOS	14	2
MTSAT	15	1 - 2
CORIOLIS	16	1
JPSS/NPP	17	0
GIFTS	*18*	*1*
Sentinel3	19	1
MeghaTropique	20	1
Kalpana	*21*	*1*
Meteor	*22*	*1*
FY3	23	1 - 3

Platform	RTTOV ID	Sat ID range
COMS	24	1
METEOR-M	25	1
GOSAT	*26*	*1*
CALIPSO	27	1
Reserved	28	-
GCOM-W	29	1
Nimbus	30	3, 4, 6, 7
Himawari	31	8
MTG	32	1
Saral	33	1
Metop-SG	34	1
Landsat	35	4,5,7,8
Jason	36	2
GPM	37	1
INSAT-1	*38*	*1 - 4*
INSAT-2	*39*	*1 - 5*
INSAT-3	*40*	*1 - 5*
Reserved	41	-
DSCOVR	*42*	*1*
CLARREO	*43*	*1*
TICFIRE	*44*	*1*
Reserved	45	-

¶ Includes TIROS-N

Table 2. Platforms supported by RTTOV as at September 2015. Platforms in italics are not yet supported in the RTTOV v11 distribution but can be requested.

Sensor	RTTOV ID	Sensor Chans => RTTOV v11 Chans (IR/MW-only)	Sensor Chans => RTTOV v11 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	1-3 => 1-3	-
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 CrIS-FSR	27	1-1305 => 1-1305 1-2211 => 1-2211	-
Spare	28	-	-
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*
FY3 MWTS	40	1-4 => 1-4	-
FY3 MWHS	41	1-5 => 1-5	-
FY3 IRAS	42	1-20 => 1-20	-
FY3 MWRI	43	1-10 => 1-10	-
GOES-R 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*	-	-
Calipso 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	1-9 => 11-20	1-20 => 1-20
MWS	78	1-24 => 1-24	-
MWI	79	1-26 => 1-26	-
EPIC	*80*	-	-
MRIR	81	2-5 => 1-4	-
Reserved	82-86	-	-
MERSI-1	*87*	-	-
MERSI-2	*88*	-	-

*channels 19-21 are only simulated accurately with Zeeman coeff file
** channels in coefficient files are in order of decreasing wavenumber
*** channel numbering/ordering follows instrument convention

Table 3. Instruments supported by RTTOV v11 at September 2015. Sensors in italics are not yet supported in the RTTOV v11 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. Users 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. A detailed description of the steps involved in running RTTOV is given in section 7.

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.

Important notes on v7 vs v9 predictor coefficients:

- Where both v7 and v9 predictor files exist for an instrument (see Table 3), the v7 predictor file supports *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. Therefore in many cases the channel numbering for the IR channels is *not the same* for the v7 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 predictors. For IR-only simulations users may find the v7 predictors remain the optimal choice.

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 v11 coefficient files:
http://nwpsaf.eu/deliverables/rtm/rttov11_coefficients.html

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. Once all simulations are complete, you should call a number of deallocation subroutines to release allocated memory.

	v7 predictors	v8 predictors	v9 predictors
Maximum zenith angle	75 degrees	75 degrees	84* degrees
Solar computations?	No	No	Yes
MW instruments	All MW sensors. Coefs on 54 levels. Trace gases: H2O	N/A	N/A
Hyperspectral sounders	Coefs on 54 and 101 levels. Trace gases: H2O, O3	Coefs on 101 levels Trace gases: H2O, O3, CO2	Coefs on 101 levels Trace gases: H2O, O3, CO2, CO, NO2, CH4
IR instruments (including those with visible/near-IR channels)	All IR instruments except SSU IR channels only Coefs on 54 levels Trace gases: H2O, O3	SSU on 51 levels HIRS on 54 levels IR channels only Trace gases: H2O, O3, CO2	Selected instruments (see final column in Table 3) Coefs on 54 levels Visible/near-IR/IR channels Trace gases: H2O, O3

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 RTTOV v10

This section provides details of the differences in the user interface between RTTOV v10 and RTTOV v11. In particular, this section should be useful for users who wish to replace v10 with v11 in their application as it lists the changes to the user-interface. Table 5 below summarises the changes.

Coefficient files

The majority of coefficient files have been regenerated using the latest LBLRTM model. In addition they provide coefficients over a new set of 54 levels (see Table 1). The previous 51 levels have uneven layer thicknesses near the top of the profile (above 13hPa) which could result in artefacts appearing in Jacobians of high-peaking channels. The new 54 levels were derived from an analytic formula giving smoothly-varying layer thicknesses. See Table 4 for a summary of the available optical depth coefficient files. Note that the existing 51 level files are fully compatible with RTTOV v11.

Return status values

All RTTOV v11 subroutines have just two possible return codes: `errorstatus_success` and `errorstatus_fatal` (both defined in the `rttov_const` module) which indicate a successful call or a failure respectively. In RTTOV v10, the `errorstatus_warning` return code was used to indicate that an input profile had exceeded the regression limits. In RTTOV v11 this will print a warning, but there is no return status indication of the warning. Instead users have various options:

1. Call the `rttov_user_profile_checkinput` subroutine on the input profiles before calling RTTOV. This can give warnings when regression limits are exceeded and allows a profile to be rejected without running the full RTTOV simulation if desired.
2. Set `opts%config%do_checkinput` to .false. (recommended if calling `rttov_user_profile_checkinput`) so that no checks against the regression limits are performed.
3. Set `opts%config%apply_reg_limits` option to reset any values which exceed the limits back to the limit values.

User checkinput subroutines

The interfaces to the user-level subroutines `rttov_user_options_checkinput` and `rttov_user_profile_checkinput` have changed to be more consistent with one another and other subroutines (see Annex N).

Options structure

Changes in the `rttov_options` structure (Annex O tabulates the structure including notes on options which are new to RTTOV v11 and options which have changed in meaning since RTTOV v10):

- the options have been organised into sub-types according to their function. This should improve clarity regarding the conditions under which each option applies.
- the default value for `use_q2m` is now true (note this has little impact for IR window channels as surface emissivities are usually not far from 1.0).
- the default value for `fastem_version` is now 5.
- the `verbose_checkinput_warnings` flag present in RTTOV v10 has been replaced by a `verbose` flag. If `verbose` is false only error messages for fatal errors are output; all other messages are suppressed.

Error handling

- The optional `rttov_errorhandling` subroutine (see Annex B) now only allows the user to select the logical unit for error messages.
- There is no longer a "verbosity level" as in RTTOV v10: verbosity is controlled via the `opts%config%verbose` flag.

Reading coefficients

The `rttov_setup` subroutine has been removed. Coefficients should be read using `rttov_read_coefs` (see Annex C) and it is no longer necessary to call `rttov_init_coefs` after calling `rttov_read_coefs`. It is recommended to read coefficients by specifying the filename although it is still possible to use the instrument ID triplet. Note that a small number of instruments have different names and/or IDs compared to v10. See the RTTOV v11 coefficient web page for details on this.

Profile structure

- In the `profiles` structure, the `cfrac` input array for cloud fractions is one dimensional of size `nlayers`.
- IR ice cloud simulations now apply limits to the input parameters used in the selected parameterisation to conform to the range of values used in training. See section 8.5 for more details.
- There are now 13 aerosol particle types (instead of 11 in RTTOV v10) which affects the size of `profiles%aerosols(:,:)`.
- The definition of azimuth angle for solar calculations has been changed since RTTOV v10 to be consistent with that used elsewhere in RTTOV. Azimuth angle is measured clockwise with north being 0 degrees. (see Figure 4).

Cloudy IR simulations – ice cloud limits

When carrying out cloudy simulations for IR instruments using the built-in parameterisations for ice particles, RTTOV will clip the values of the ice effective diameter to the training limits given in Table 21. The `rttov_user_profile_checkinput` subroutine can be used to obtain warnings when the limits have been exceeded. See section 8.5 for more details.

Emissivity atlas interface

The emissivity atlas subroutines have been renamed for clarity and consistency with the new BRDF atlas subroutines. They are now: `rttov_setup_emis_atlas`, `rttov_get_emis` and `rttov_deallocate_emis_atlas`. They take the `coefs` structure as an argument rather than the `coefs%coef` structure as in RTTOV v10 (see Annex F). By default the IR emissivity atlas, once initialised, can be used to return emissivities for any IR instrument. There is now an option to initialise the atlas for a single instrument: this results in significantly faster calls to `rttov_get_emis`, but the atlas can only be used with the specific instrument/coefficient structure passed in to the setup subroutine.

Surface emissivity input/output

Surface emissivities are passed into and out from RTTOV via a single `emissivity(:)` array argument of type `rttov_emissivity` which has two members, `emis_in` and `emis_out`, as described in Annex O.

Output radiance structure

The output radiances have been split into two structures: the `radiance_type` contains the radiances, brightness temperatures and reflectances most often used. The `radiance2_type` contains additional, optional radiance outputs which are only calculated by the direct model for non-scattering, non-PC simulations (see Annex O for a description of these structures). These secondary radiances are now calculated only if the new optional `radiancedata2` argument is passed to `rttov_direct`. Also note that overcast radiances are *not* calculated for aerosol scattering or PC simulations.

Interfaces to main RTTOV subroutines

There are new optional arguments to `rttov_direct`, `rttov_tl`, `rttov_ad` and `rttov_k`. These are:

- `calcrefl`, `reflectance`, and `reflectance_tl/ad/k` for passing surface BRDFs
- `aer_opt_param` and `cld_opt_param` for passing the aerosol and/or cloud scattering optical parameter profiles.

The `calcemis` and `emissivity` arguments are now optional and the order of the arguments has changed so that all mandatory arguments precede optional arguments. The interfaces to these subroutines are detailed in Annexes I, J, K and L.

PC-RTTOV K input

When calling `rttov_k` for PC calculations, the user should now specify the input perturbation as for the AD (see Annex L). In RTTOV v10 the PC-score/radiance/BT perturbation was hardcoded for the K model.

RTTOV-SCATT

- In RTTOV-SCATT the old cloud fraction option present in previous versions has been removed.
- Hydrometeor profiles may be input in units of kg/kg or kg/m^2/s; the default is now kg/kg.
- Surface emissivities may now be passed in using the new `rttov_emissivity` structure and this also returns the emissivities calculated/used by RTTOV.
- An options structure specific to RTTOV-SCATT has been created to allow control over a limited number of RTTOV options and the `lusercfrac` flag is now included in this structure.

See Annex M and O for more information on the `rttov_options_scatt` structure and the `rttov_scatt` subroutine interface.

Type	**Name**	**Description of change**
Constants	errorstatus_warning	Removed.
Default values	opts%fastem_version	opts%rt_mw%fastem_version is 5 by default.
	opts%use_q2m	opts%rt_all%use_q2m is true by default.
Derived types	rttov_options	Options split into sub-types according to function.
	opts%verbose_checkinput_warnings	RTTOV verbosity is controlled by opts%config%verbose.
	profiles%cfrac	Now one-dimensional (with no loss of functionality).
	profiles%aerosols	Accommodates 13 particle types.
	profiles%azangle profiles%sunazangle	Change to the way azimuth is defined for solar simulations to be consistent with the rest of RTTOV (section 8.2).
	Radiance structure	Holds only the "primary" radiances; the radiance2_type structure holds the secondary direct model radiances. Overcast not calculated for PC or aerosol simulations.
New derived types in v11	rttov_emissivity	For surface emissivity input/output.
	rttov_reflectance	For surface reflectance input/output.
	rttov_opt_param	For input of aerosol/cloud optical parameter profiles.
	radiance2_type	For output of secondary direct model radiances.
	rttov_options_scatt	Options structure for RTTOV-SCATT.
Subroutines	rttov_errorhandling	No longer specifies a verbosity level.
	rttov_setup	Removed; use rttov_read_coefs instead.
	rttov_init_coefs	No longer necessary to call this.
	rttov_atlas_setup	Renamed to rttov_setup_emis_atlas, change to interface.
	rttov_get_emis	Change to interface.
	rttov_deallocate_atlas	Renamed to rttov_deallocate_emis_atlas.
	rttov_user_options_checkinput	Change to interface.
	rttov_user_profile_checkinput	Change to interface.
	rttov_direct/tl/ad/k	Changes to interface (see Annexes I, J, K, L).
	rttov_k for PC calculations	User must now specify the input perturbation.
	rttov_scatt	Changes to interface. Cloud fraction option removed. Hydrometeor profiles may be in units of [kg/kg]. Surface emissivities passed in/out via rttov_emissivity structure.

Table 5. Summary of user interface changes since RTTOV v10.

Additional capabilities of RTTOV v11 over RTTOV v10 are listed below and are described elsewhere in this user guide:

- Capability to perform simulations for visible/near-IR channels (section 8.2).
- New land surface BRDF atlas to specify input surface reflectance (sections 8.2 and 7.6).
- New volcanish ash and Asian dust aerosol particle types (section 8.6).
- New Baran parameterisation for ice particle optical parameters (section 8.5).
- Option to specify cloud and/or aerosol optical parameter profiles explicitly instead of using pre-defined particle types (sections 8.5 and 8.6).
- Capability to account for non-local thermodynamic equilibrium effects for hi-res sounders (section 8.9).
- Option to treat the surface as a Lambertian reflector for the reflected down-welling radiance calculation (section 8.10).
- PC-RTTOV can now be run for clear and cloudy scenes over sea surfaces, and for IASI it is possible to run clear-sky simulations for limited spectral bands as well as the full spectrum (section 8.8).
- Capability to read and write coefficients in HDF5 format (Annexes A and C).
- Capability to convert mietables to binary format for faster reading in RTTOV-SCATT (Annex A).
- Updates to SSU coefficients to account for time-variation of cell pressure (section 8.12).
- Additional options for the internal interpolation (section 7.3).
- Improved speed and memory usage of IR land surface emissivity atlas (Annex F).

Additional capabilities introduced in RTTOV v11.2 are:

- New MW sea surface emissivity option FASTEM-6 (section 8.4).
- Improved version of the Baran ice parameterisation for IR cloudy simulations (section 8.5).
- New option to extrapolate input profiles at the top of the atmosphere using the regression limits (section 7.3).
- RTTOV graphical user interface (GUI) is included in the package (see separate GUI user guide in docs/ directory).

Additional capabilities introduced in RTTOV v11.3 are:

- Options for units of input gas profiles and consistent treatment of gas units within RTTOV (section 7.3).
- Lambertian surface option available for IR instruments (section 8.10).
- Option to provide ocean surface foam_fraction explicitly to FASTEM MW emissivity model (section 8.4).
- New clear-sky PC-RTTOV coefficients which may be used over all surface types (section 8.8).
- Optional zenith angle correction for IR emissivity atlas (section 8.4).
- Improved treatment of snow in BRDF atlas (section 7.6).
- IR emissivity and BRDF atlases now available in HDF5 format (section 5.2).
- New subroutines which can be used to allocate any/all input/output arrays and structures for RTTOV in a single call (Annex D and example_*.F90 programs in src/test/).
- RTTOV example programs have been simplified further and new examples added (see src/test/).
- New wrapper which allows RTTOV to be called from Python, C and C++ (section 5.2 and see separate wrapper user guide in docs/ directory).

5. FORTRAN-90 UNIX/LINUX installation and testing

RTTOV v11 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 and the `readme.txt` file.

The following system components are needed before running RTTOV v11:

- UNIX or Linux operating system
- Fortran 90 compiler
- 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).
- Typically 10 Mbytes of memory are required to run the code, but this strongly depends on the instrument being simulated and the kind of simulations being performed.
- 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).
- If you do not compile against the HDF5 library, you can still use the emissivity or BRDF atlases by compiling against the NetCDF library (v3.6 or higher).
- 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.

Some basic information on installing the RTTOV v11 Fortran 90 code in a UNIX or Linux environment follows. This assumes the code is obtained as a compressed UNIX tar file via FTP or on CD-ROM. The file name should be *rttov113.tar.gz* and be copied to your 'top' RTTOV directory (e.g. ~user/rttov11) from which subdirectories will be created.

RTTOV v11 will not work with older versions of some compilers. The following list gives the versions of several common compilers known to be compatible:

- gfortran – v4.4.7 and later
- ifort – v12.0.4 and later
- NAG – v5.2, v5.3
- pgf90 – v11.7 and later
- IBM – xlf95 v12.1 and v13.1
- Cray Fortran – v8.3.4

5.1 Unpacking the code

This is achieved using the command:

```
$ tar zxvf rttov113.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_rttov11/`	RTTOV v11 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 v11 coefficients is available via the RTTOV v11 web page which provides details of which features are supported by each file:

http://nwpsaf.eu/deliverables/rtm/rttov11_coefficients.html

Users are recommended to refer to this web page to determine the most appropriate coefficient file for their application.

The `rtcoef_rttov11/` directory contains sub-directories for each kind of coefficient file:

`rttov7pred54L/`	v7 predictor files on 54 levels (most 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 O3 and CO2)
`rttov8pred54L/`	v8 predictor files on 54 levels (allows variable O3 and CO2)
`rttov9pred54L/`	v9 predictor files on 54 levels (for VIS/NIR/IR sensors allowing solar calculations)
`rttov9pred101L/`	v9 predictor files on 101 levels (hi-res IR sounder files allowing more variable gases)
`cldaer/`	IR cloud and aerosol scattering coefficient files
`mietable/`	MW scattering coefficient files
`pc/`	Principal Components coefficient files

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

- Zeeman coefficient files for AMSU-A and 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; new file format for v11.
- IR aerosol and cloud scattering coefficients for most IR sensors; note that scattering coefficients are not supplied for visible channels.
- MW scattering coefficients for most MW instruments.

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, and PC-RTTOV 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_rttov11/` 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.

All v10 coefficient files may be used with RTTOV v11 with the exception of PC coefficients.
All new v11 coefficient files may be used with RTTOV v10 with the following exceptions:

- v9 predictor visible/near-IR coefficient files
- aerosol coefficient files
- PC coefficient files

Note that any binary coefficient files used with v10 must be recreated from the ASCII or HDF5 files for v11.

For most instruments the coefficients are supplied in ASCII format. However due to the large file sizes HDF5 is now the preferred format for all hypersepctral IR sounder coefficient files including IR scattering and PC coefficients. 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 reduces file sizes and can improve performance, particularly for hyperspectral sounders. The `rttov_conv_coef.exe` program performs these tasks and is described in Annex A.

Older coefficient files from RTTOV v7, v8 and v9 cannot be used with RTTOV v11 without modification. The supplied coefficient files (and those on the web page given above) are the recommended ones for use with RTTOV v11. However a program is available to convert v7/8/9 coefficient files to be compatible with RTTOV v11 (see Annex A).

5.2 *Compiling the code*

An interactive shell script is available to compile RTTOV v11.3 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 (v5.3).

RTTOV may be compiled immediately without requiring any external libraries. However some features of RTTOV depend on either the HDF5 or netCDF library:

Reading HDF5 coefficient files	: requires the HDF5 library.
Emissivity/BRDF atlases	: require either the HDF5 or netCDF library.
Python/C/C++ interfaces	: require the emissivity/BRDF atlas code.
RTTOV GUI	: requires the HDF5 library.
Python interface and RTTOV GUI	: require that f2py is installed.

Compiling with the HDF5 library is recommended as all functionality is then available: in this case netCDF is not required. **Before compiling with HDF5 or netCDF you must first edit the `build/Makefile.local` file with the location of the required library.** This involves specifying the path to the library installation and uncommenting one set each of "FFLAGS_XXX" and "LDFLAGS_XXX" for the library.

When using the emissivity atlases with RTTOV v11.3, if you compile with the HDF5 library then the **HDF5** IR emissivity and BRDF atlas files **must** be used. Otherwise if you compiled with the netCDF library then you **must** use the **netCDF** atlas files. If you compile against the HDF5 library there is no need to specify the netCDF library as it will not be used. The HDF5 files are much smaller in size than the netCDF files, but they contain identical data. The atlas files can be downloaded from the RTTOV v11 web page.

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 *librttov11_main.a* and *librttov11_coef_io* – see section 6 for more information). Tables 7 and 8 list all libraries and executables produced by the build process.

Example stand-alone Makefiles are included for the `example_*.F90` demonstration programs. These are `src/test/Makefile_examples` and `src/brdf_atlas/Makefile_examples`. Each has a section at the top which should be edited with paths appropriate for your system. These are intended as demonstratations 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 these Makefiles 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. Before running RTTOV ensure the HDF5 library is in your `$LD_LIBRARY_PATH` or equivalent.
3. 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.
4. Note that if you do NOT compile with the HDF5 library, you must NOT supply the _RTTOV_HDF macro to the compiler (so you must comment out the HDF5 lines in `Makefile.local`).
5. If the code was previously compiled without HDF5 first then you should select a clean compilation if the recompilation fails.

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: this is the same as for RTTOV v11.2. **As noted above, if compiling with HDF5 or netCDF you must first edit the `build/Makefile.local` file with the location of the required library**. The general compilation procedure is then as follows:

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

The square brackets [] indicate optional arguments. 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 interface.

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).

all - this argument is the "target" to be compiled and is only required if you are compiling RTTOV with the netCDF library. This will ensure the emissivity and BRDF atlas code is compiled and also the RTTOV C/C++ interface code. If you do not require these parts of RTTOV or you are compiling with the HDF5 library this is not required.

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. However, if you are compiling RTTOV with the HDF5 library or you want to compile the RTTOV GUI or the RTTOV Python interface, you must run `Makefile.PL` with one or both of the arguments shown above:

RTTOV_HDF=1 - this is required if compiling RTTOV with the HDF5 library
RTTOV_F2PY=1 - this is required if compiling the RTTOV GUI or the RTTOV Python interface

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

Compile RTTOV with gfortran without any external dependencies:

```
$ make
```

Compile RTTOV including atlas code and Python interface with ifort-openmp compiler flags with the netCDF library:

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

```
$ ../build/Makefile.PL RTTOV_F2PY=1
$ make ARCH=ifort-openmp all
```

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

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:

```
$ make [ARCH=myarch] [INSTALLDIR=mydir]
```

The build directories will be placed within `mydir/` under the top-level RTTOV directory. 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 such as the NetCDF or HDF5 libraries. 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 the NetCDF and HDF libraries. 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]
$ make [ARCH=myarch] [INSTALLDIR=mydir] clean
```

where RTTOV_HDF=1 is optional and should only be supplied if the HDF5 code is required. 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 the 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 90 compiler.
`FC77` : the name of the Fortran 77 compiler; this might be the Fortran 90 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 v11 software must be compiled with a Fortran 90 compiler.

This configuration file may also define the following macros:

`FFLAG_MOD`: this is the flag used by the Fortran 90 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, IR scattering coefficients or PC-RTTOV coefficients). Note that as of v11.3 the majority of tests defined in the test suite expect hi-res IR sounder and PC-RTTOV coefficients in HDF5 format.

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

```
$ ./test_rttov11.sh ARCH=myarch [BIN=bindir]
```

The `ARCH` parameter should match the one used when you compiled RTTOV. The `BIN` parameter is optional (indicated by the square brackets []). 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`).

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-depedent 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_rttov11.sh`	tests the full code (direct/TL/AD/K) for a range of instruments
`test_rttov11_hires.sh`	tests the full code for hi-res IR sounders

`test_solar.sh`	tests visible/near-IR solar simulations
`test_pc.sh`	tests the Principal Component calculations
`test_multi_instrument.sh`	tests RTTOV running for multiple instruments together
`test_zeeman.sh`	tests Zeeman code (using Zeeman coefficient files)
`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)
`test_cpu.sh`	for performance testing (this test has no reference data)

You can run any of these scripts in exactly the same way as described above for `test_rttov11.sh`. Note that most tests for hyperspectral IR sounders expect HDF5 format coefficient files. The exceptions are the `test_coef_io*.sh` scripts which run tests which using ASCII format files. It is not necessary to run all the scripts to verify your RTTOV installation: calling `test_rttov11.sh` is sufficient.

A full description of the RTTOV v11 test suite may be found in `docs/rttov-test-v11.3.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, a reference to the RTTOV coefficients, and so on. 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 v11

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:

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

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 directories named test_example_*.2/. Input files for the script are in the test_example_*.1/ directories, and these are 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 is fine.

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_rttov11/rttov7pred54L/ and rtcoef_rttov11/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. Note there is no reference output for this example program.

Emissivity atlas testing

The emissivity atlas code must be compiled (see section 5.2). The test_iratlas.sh, test_mwatlas.sh and test_cnrmmwatlas.sh shell scripts may be used to verify the IR, TELSEM MW, and CNRM MW atlas code respectively: in each case, these test programs initialise the atlas, return emissivity values for a series of profiles/locations and then deallocate the atlas. The emissivities are written to the 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_rttov11/rttov7pred54L/), and the location of the emissivity atlas data files (by default assumed to be in emis_data/). The test scripts require emissivity data for the month of August (IR and TELSEM MW) and June (CNRM MW). Note that the IR emissivity atlas test requires the all of the IR atlas files to be downloaded (including the covariance files and the new angular correction files) for this month. You must use the HDF5 format IR atlas files if RTTOV was compiled against the HDF5 directory or otherwise the NetCDF files are required.

The scripts may be run by typing:

```
$ ./test_iratlas.sh ARCH=myarch [BIN=bindir]
$ ./test_mwatlas.sh ARCH=myarch [BIN=bindir]
$ ./test_cnrmmwatlas.sh ARCH=myarch [BIN=bindir]
```

Test reference output is in test_emisatlas.2/. Input files for the scripts are in the test_emisatlas.1/ directory, and this is also where the test output is written. The output consists of files named output_iratlas.NN.myarch, output_mwatlas.NN.myarch,and output_cnrmmwatlas.NN.myarch where NN is 01, 02, etc. The scripts also write diff files named diff_iratlas.NN.myarch, diff_mwatlas.NN.myarch and diff_cnrmmwatlas.NN showing the difference between the test output and the reference output. The difference files should have zero size.

BRDF atlas testing

The BRDF atlas code must be compiled (see section 5.2). The `test_brdf_atlas.sh` shell script may be used to verify the atlas code: the test program initialises the atlas, returns BRDF values for a series of profiles/locations and then deallocates the atlas. The BRDFs are written to the output file. You may need to edit the first few lines of the script to specify the location of the RTTOV coefficient files (by default assumed to be in `rtcoef_rttov11/rttov9pred54L/`), and the location of the BRDF atlas data files (by default assumed to be in `brdf_data/`). The test script requires BRDF data for the month of August. You must use the HDF5 format BRDF atlas files if RTTOV was compiled against the HDF5 directory or otherwise the NetCDF files are required.

The script may be run by typing:

```
$ ./test_brdf_atlas.sh ARCH=myarch [BIN=bindir]
```

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 libararies, executables and subroutines

This section summarises the output from the build process and the 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 users require in their applications. All types are defined in the `src/main/rttov_types.F90` module. Annex O provides details of all the derived types.

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 linking against RTTOV you must specify those libraries which depend on other libraries first. For example:

```
-lrttov11_mw_scatt -lrttov11_brdf_atlas -lrttov11_emis_atlas
-lrttov11_other -lrttov11_parallel -lrttov11_coef_io
-lrttov11_hdf -lrttov11_main -lnetcdf -lhdf5hl_fortran -lhdf5_hl
-lhdf5_fortran -lhdf5
```

The files `src/test/Makefile_examples` and `src/brdf_atlas/Makefile_examples` are example Makefiles for the `example_*.F90` executables which may be used as templates for compiling your own code which calls RTTOV. As noted previously the example executables are built when RTTOV is compiled: these example Makefiles are for demonstration purposes.

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.

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.
profile_type	Input profile.
profile_cloud_type	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).
radiance_type	Calculated radiances.
radiance2_type	Secondary direct model radiances (optional).
transmission_type	Calculated transmittances.
rttov_pccomp	Calculated PC scores and reconstructed radiances.

Table 6. User-level derived types.

Library name	Subroutines	Description
librttov11_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.
librttov11_hdf.a	N/A	Required if RTTOV was compiled with HDF5 capability.
librttov11_coef_io.a	rttov_read_coefs rttov_dealloc_coefs rttov_get_pc_predictindex	Coef input/output, always required.
librttov11_parallel.a	rttov_parallel_direct rttov_parallel_tl rttov_parallel_ad rttov_parallel_k	Parallel interfaces for multi-threaded execution using OpenMP.
librttov11_mw_scatt.a	rttov_scatt, rttov_scatt_tl rttov_scatt_ad rttov_read_scattcoeffs rttov_dealloc_scattcoeffs rttov_alloc_scatt_prof rttov_init_scatt_prof rttov_scatt_setupindex	Interface to RTTOV-SCATT for MW scattering simulations.
librttov11_emis_atlas.a	rttov_setup_emis_atlas rttov_get_emis rttov_deallocate_emis_atlas	Interface to emissivity atlases (requires HDF5 or NetCDF).
librttov11_brdf_atlas.a	rttov_setup_brdf_atlas rttov_get_brdf rttov_deallocate_brdf_atlas	Interface to BRDF atlas (requires HDF5 or NetCDF).
librttov11_other.a	rttov_bpr_calc rttov_bpr_init rttov_bpr_dealloc rttov_print_opts rttov_print_profile rttov_print_info rttov_aer_clim_prof rttov_zutility	Ancillary subroutines: • calculate "b" parameter for IR cloud/aerosol simulations with explicit optical parameters • print out contents of options, profile and coef structure for debugging • generate climatological aerosol profiles • obtain values of magnetic field strength for Zeeman simulations
rttov_wrapper_f2py.so	See separate wrapper user guide for wrapper API.	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: librttov11_test.a, librttov11_mw_scatt_coef.a, librttov11_coef_io_789.a, librttov11_gui.a, librttov11_wrapper.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_conv_coef.exe	Convert coefficients between formats (ASCII, binary, HDF5) and extract channels to reduce file sizes.
rttov789_conv_coef.exe	Convert older coefficient files to v10/v11 format.
rttov789_conv_coef_11to9.exe	Convert v10/v11 format coefficient files to v9 format.
rttov_ascii2bin_scattcoef.exe	Convert ASCII RTTOV-SCATT Mie tables to binary format.
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_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_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_k.exe	Example program demonstrating simple K model call.
test_iratlas.exe	Test program for IR emissivity atlas.
test_mwatlas.exe	Test program for TELSEM MW emissivity atlas.
test_cnrmmwatlas.exe	Test program for CNRM MW emissivity atlas.
test_brdf_atlas.exe	Test program for BRDF atlas.
rttovscatt_test.exe	RTTOV-SCATT test executable, should be run using the test_rttovscatt.sh script.
example_rttovscatt.exe	Example program demonstrating calling RTTOV-SCATT.
Additional executables.	
rttov_scatt_make_coef.exe	Generate RTTOV-SCATT Mie table files.
rttov_mie_params_aer.exe	Generate IR aerosol coefficient files.
rttov_mie_params_cld.exe	Generate IR cloud coefficient files (water particle properties only).
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).

Table 8. Executables produced by the RTTOV build process.

7. Running RTTOV v11 for your applications

RTTOV v11.3 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.

To run RTTOV v11 for a user's application there are several example programs which can serve as a basis for your application:

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

These may be found in the **src/test/** directory except for **example_atlas_fwd.F90** which is in **src/brdf_atlas/**.

Use the modules **rttov_types** and **rttov_const** in your program for the definition of derived types and constants (Table 6 gives a list of derived types with full details in Annex O, and Annex P gives the source of **rttov_const**), and module **parkind1** for the definition of standard RTTOV integer, real and logical kinds (**jpim**, **jprb** and **jplm** respectively). The default RTTOV real kind is double precision (64-bit). It is *not* recommended to change this to single precision (32-bit) if running the AD or K models as this can significantly affect the adjoint/Jacobian output. 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 v11. Annex Q gives the code for **example_fwd.F90** which includes comments to guide the user.

It is recommended that users look at the header section of the coefficient file for the sensor they wish to simulate as there is useful information there such as the definition of channel numbers and the polarisation assumed for each channel for that instrument etc.

The following sections describe the recommended steps to be taken in coding a program which calls RTTOV v11. 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 v10 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,
radiancedata, radiancedata2, calcemis, emissivity, calcrefl, reflectance,
aer_opt_param, cld_opt_param, traj, pccomp, channels_rec)
```

7.1. Set RTTOV options

The user must first declare the values of **opts** of the *rttov_options* derived type. This structure contains a number of flags with which the user can configure various aspects of RTTOV. The full structure is described in Annex O.

The user may 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**) comprises the following members:

- **coef** : structure containing data read from the RTTOV coefficient file(s) appropriate for the instrument(s) being simulated. This is mandatory.
- **coef_scatt_ir** : structure containing cloud and aerosol scattering coefficients (only used if **opts%rt_ir%addclouds** or **opts%rt_ir%addaerosl** set to true and **opts%rt_ir%user_cld_opt_param** or **opts%rt_ir%user_aer_opt_param** respectively are false).
- **optp** : structure containing optical properties (only used if **opts%rt_ir%addclouds** or **opts%rt_ir%addaerosl** set to true and **opts%rt_ir%user_cld_opt_param** or **opts%rt_ir%user_aer_opt_param** respectively are false).
- **coef_pccomp** : structure containing coefficients for the Principal Components calculations (only used if **opts%rt_ir%pc%addpc** set to true).

The user should declare an instance of the coefs structure and should 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 for reading the coefficient files, it is better to access Fortran unformatted or, new to RTTOV v11, HDF5 formatted coefficient files. The program **rttov_conv_coef.exe** (located in the `bin/` directory of the build) can be used to convert the ASCII coefficient files provided with the distribution to Fortran unformatted or HDF5 files. It can also produce ASCII, unformatted or HDF5 coefficient files for a subset of channels which can be useful in particular for hi-res sounders. The command-line arguments for this tool are described in Annex A.

Another program, **rttov789_conv_coef.exe**, is available to convert version 7-, 8- or 9- coefficient files to version 11-compatible files: this routine is also documented in Annex A. Take care of the compilation options because the user should always ensure that the compilation of the binary file creation program is consistent with the compilation for RTTOV. The routine **rttov_read_coefs** reads headers for checking the single/double precision and normally gives an error message if an incompatible binary coefficient file is being read, but this may not be fully failsafe.

The coefficient file defines the variable trace gases allowed in the input profile (see Table 4). There are two points for the user 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) are optional. MW simulations may also use a cloud liquid water input profile which is treated in a very similar manner to the gases.

If the coefficient file being used contains coefficients for a particular gas (e.g. **coefs%coef%nozone** > 0) then a profile may be supplied for that gas and 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 the MW coefficients include ozone as an active gas.

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 is ozone.

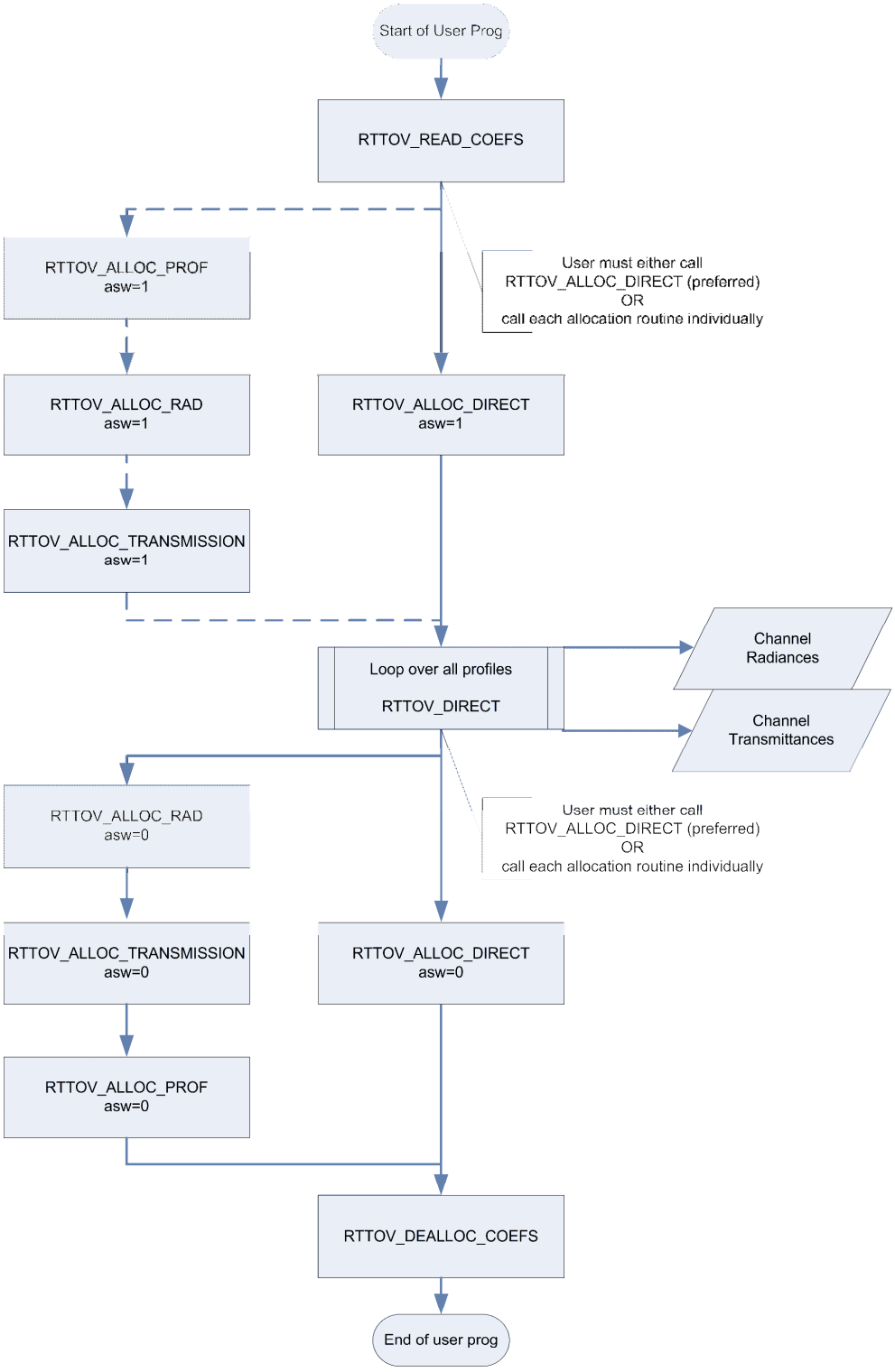

Figure 1. Process diagram of user program calling RTTOV v11.3 forward model.

7.3. Set up input profiles for RTTOV v11

Allocate a **profiles** array (derived type *profile_type*) 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 or false 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).

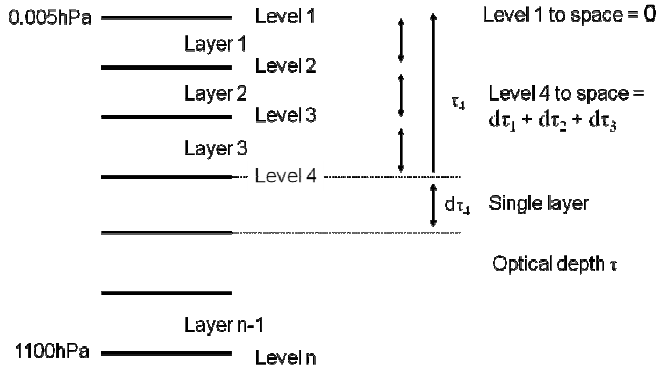

Figure 2. Internal RTTOV coefficient levels and optical depth computations.

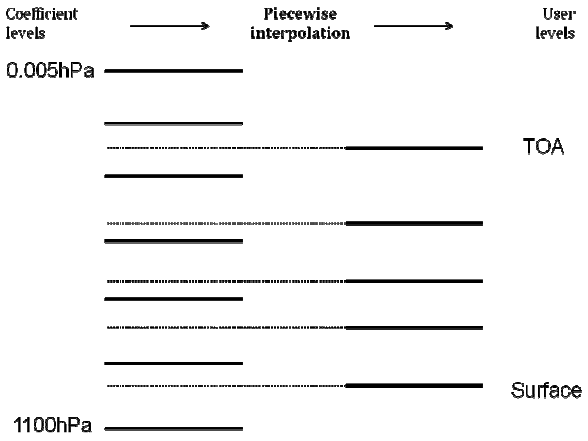

Figure 3. Interpolation to and from to user levels.

The user's input profile is normally on different pressure levels from that required internally by RTTOV (see Figures 2 and 3). RTTOV v11 has the capability to interpolate profiles specified on user levels onto the required RTTOV pressure levels defined in the coefficient file. The method of interpolation is given in Rochon *et al.* (2007) and is also described in the RTTOV v9 science and validation report. The interpolation is enabled by setting **opts%interpolation%addinterp** to true. Invoking this option does increase the run time of the model significantly. Note that input profiles can be on different levels, but for each call to RTTOV the number of levels must be the same for all profiles.

There are two interpolation steps within RTTOV: firstly the input profile is interpolated onto the coefficient levels. Once the fast optical depth calculation has been carried out on coefficient levels, the resulting optical depths are interpolated back onto the input levels and the radiative transfer equation is integrated over the input levels.

One key 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 v11 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 implemented in RTTOV v9 and v10. 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 of RTTOV such as 1D-VAR 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 RTTOV v9/v10 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). Users may wish to carry out experiments to determine the best interpolation option for their application.

Note that you can of course opt to interpolate their input profiles onto the RTTOV coefficient levels before input to RTTOV (and therefore switch off the RTTOV interpolation) if you find this beneficial for their application.

interp mode	Profile interp (user->coef levels)	Optical depth interp (coef->user levels)	Description
1	Rochon	Rochon on optical depths	Default, same as v9/v10, 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 “features” 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 v11.

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

i. Ideally, the user 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 user profile should span the range of pressures over which the weighting functions of the channels being simulated are significantly greater than zero.

ii. The user profile lowest level (**profiles%p(nlevels)**) should be equal or greater in value than the 2m pressure (**profiles%s2m%p**). Note that RTTOV will still run if this advice is not adhered to: the profile values will be extrapolated from the lowest input level down to the surface at constant value.

iii. 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.

iv. If the user profile is below the top of the coefficient file the user profile is extrapolated assuming a constant value of the uppermost user value for all levels above the top. Alterntively there is an option to carry out extrapolation based on the coefficient file regression limits (see below).

v. 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.

The **example_fwd.F90** program in Annex Q (and in **src/test/**) is a useful guide on how to set up the profile arrays. If a coefficient file with trace gas coefficients (e.g. AIRS or IASI) is used and suitable trace gas profile concentrations are not available then the **opts%rt_ir%co_data** variable for CO for example can be set to false and the reference CO profile is then used for the calculation. If the variable is true however and the CO profile contains zeroes then the program will abort. This applies to all gases except water vapour which is always mandatory. For Principal Components calculations only water vapour and ozone are variable gases. The value of the **opts%rt_ir%co2_data, co_data**, **n2o_data**, and **ch4_data** flags is ignored as the reference profiles for CO2, CO, N2O and CH4 are always used.

All profile variables must lie within the "hard" limits specified in section 1.9 of **rttov_const.F90** (see Table 10). 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 predictor calculations are carried out for every layer.

Variable	Minimum	Maximum
Temperature (inc. 2m T)	90 K	400 K
Water vapour (inc. 2m q)	1E-11 ppmv	60000 ppmv
O3	1E-11 ppmv	1000 ppmv
CO2	1E-11 ppmv	1000 ppmv
CO	1E-11 ppmv	10 ppmv
N2O	1E-11 ppmv	10 ppmv
CH4	1E-11 ppmv	50 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)	50 hPa	1100 hPa
Magnetic field Be (Zeeman only)	0.2 Gauss	0.7 Gauss

Table 10: Hard limits for input profiles variables as defined in rttov_const.F90. Input values must not lie outside the interval [minimum, maximum].

As of RTTOV v11.3 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 water vapour. It has come to light that prior to RTTOV v11.3 the treatment of gas units was not consistent between IR and MW coefficient files and in addition within RTTOV there were some discrepancies in the assumed units. RTTOV v11.3 allows for fully consistent treatment of gas units. Table 11 lists the available options for gas units. Full details of the changes and the impacts observed for several instruments are given in http://nwpsaf.eu/deliverables/rtm/docs_rttov11/rttov_v11.3_gas_units_and_new_coefs.pdf. New MW coefficient files have been generated which are consistent with the treatment of IR instruments.

profiles(:)%gas_units	Description
2	ppmv over moist air
1	kg/kg over moist air
0	"compatibility mode": this is the default in v11.3. This replicates the way RTTOV v11.2 and earlier treat gas units and so radiances are unchanged. There are some internal inconsistencies within RTTOV in the assumed gas units and so this is not strictly recommended although the errors are negligible in practice. For IR instruments gas profiles should be in ppmv over dry air. For MW instruments (old coefficient files) gas profiles should be in ppmv over moist air.
-1	ppmv over dry air: this is intended primarily for coefficient generation and testing, but is a valid input option.

Table 11: Options for gas units

The clear-sky optical depth calculations are based on regressions derived from line-by-line calculations, and the validity limits for the regressions are given in the coefficient files. There are two options for how to deal with input profiles that fall outside the regression limits. If **opts%config%apply_reg_limits** is set to false, RTTOV v11 will print a warning to the output logical unit whenever the input profiles are outside these regression limits (warning messages may be suppressed by setting **opts%config%verbose** to false), but RTTOV v11 will nevertheless perform the full radiative

transfer calculation, using the profile values outside the regression limits. If **opts%config%apply_reg_limits** is set to true the profiles used for the optical depth calculations are reset to the limit values if some input values fall outside the regression limits and no warning is printed. This is only applied to the values on levels at or above the surface pressure defined by **profile%s2m%p**. Note that the regression limits are applied to the clear-sky optical depth calculations only; the source function for the radiative transfer equation is always based on unadjusted user input profiles. The user is advised to perform their own sensitivity studies to decide which option works best for their given application. 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.

Since v11.2 RTTOV provides an option for profile extrapolation at the top of the atmosphere: this may be useful if the input profile top is lower than the coefficient file pressure levels which reach 0.005hPa. 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 0.005hPa 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 realistic 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.

By default RTTOV checks the input profiles for unphysical values before carrying out the optical depth calculation. It is possible to turn off this internal profile checking by setting **opts%config%do_checkinput** to false. Note that as of RTTOV v11.2 the **opts%config%apply_reg_limits** functionality is independent of the setting of **opts%config%do_checkinput** so these two options can be switched on or off independently.

Users can also use the **rttov_user_profile_checkinput** subroutine (Annex N) 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 users should set **opts%config%do_checkinput** to false to turn off the internal RTTOV checkinput call.

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		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 flag .true.	Y
profiles(i) % co2(:)	CO_2 conc on levels	ppmv or kg/kg	N	If flag .true.	Y
profiles(i) % n2o(:)	N_2O conc on levels	ppmv or kg/kg	N	If flag .true.	Y
profiles(i) % co(:)	CO conc on levels	ppmv or kg/kg	N	If flag .true.	Y
profiles(i) % ch4(:)	CH_4 conc on levels	ppmv or kg/kg	N	If flag .true.	Y
profiles(i) % clw(:)	Microwave cloud liquid water treated as absorbing medium; do not use with RTTOV-SCATT.	kg/kg	N	MW clear-sky only	Y
profiles(i) % aerosols(:,:) indices are (aerosol_type, layers)	Aerosol components conc on layers	cm^{-3}	N	IR aerosol scatt unless user inputs optical params	Y

profiles(i) % cloud(:,:) indices are (cloud_type, layers)	Cloud water/ice content on layers	g.m^{-3}	N	IR cloud scatt unless user inputs optical params	Y
profiles(i) % cfrac(:,:)	Cloud fractional cover on layers	0-1	N	IR cloud scatt	Y
profiles(i) % icede(:)	Ice particle effective diameter	microns	N	Optional with IR cloud scatt	Y
profiles(i) % idg	Ice water content effective diameter scheme	1-4	N	IR cloud scatt unless user inputs optical params or selects baran scheme	N
profiles(i) % ish	Ice crystal shape (1, 2) or use Baran scheme (3, 4)	1-4	N	IR cloud scatt unless user inputs optical params	N
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	Y
profiles(i) % s2m % wfetc	Wind fetch	m	N	Sea surface BRDF model	Y
profiles(i) % skin % surftype	Surface type (land = 0, sea = 1, seaice = 2)	0-2	Y		N
profiles(i) % skin % watertype	Water type (fresh = 0, ocean = 1)	0-1	N	Surface BRDF model and 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	Y
profiles(i) % skin % foam_fraction	Ocean foam fraction	0-1	N	FASTEM if supply_foam_fraction flag .true.	Y
profiles(i) % skin % fastem(1:5)	FASTEM land/sea-ice parameters (see Table 20)		N	FASTEM for land/sea-ice surface types	Y
profiles(i) % ctp	Cloud top pressure for simple cloud	hPa	N	Simple cloud VIS/IR only	Y
profiles(i) % cfraction	Cloud fraction for simple cloud	0-1	N	Simple cloud VIS/IR only	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 refractivity or emis/BRDF atlases	N
profiles(i) % longitude	Longitude (0-360)	deg	N	Emis/BRDF atlases	N
profiles(i) % snow_frac	Surface snow cover fraction	0-1	N	IR emis atlas	N
profiles(i) % soil_moisture	Surface soil moisture	m^3/m^3	N	Not used	N
profiles(i) % elevation	Elevation	km	Y	For refractivity	N
profiles(i) % Be	Earth magnetic field strength	Gauss	N	Zeeman	N
profiles(i) % cosbk	Cosine of the angle between the		N	Zeeman	N

	Earth magnetic field and wave propagation direction				
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. The user should allocate a **chanprof(:)** array (derived type *rttov_chanprof*): this defines which channels are simulated for each profile. 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.

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 VIS/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-consective 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 (i.e. **opts%rt_ir%pc%addpc** is true), the user 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 described above 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 the user 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.

For MW instruments RTTOV has an internal emissivity model called FASTEM. There are now 6 versions of FASTEM available which can be chosen by setting **opts%rt_mw%fastem_version** to a value between 1 and 6 (with 5 being the default). If a value outside this range is supplied the FASTEM version is taken from the coefficient file: for v11 coefficient files, the default is again FASTEM-5. Section 8.4 provides more information about the FASTEM versions. FASTEM is primarily a sea-surface emissivity model, though it also provides values for land and sea-ice surface types.

For IR instruments RTTOV has a sea-surface emissivity model called ISEM. Over land and sea-ice, where **calcemis(:)** is true, emissivities are assigned default values of 0.98 (land) and 0.99 (sea-ice).

For the tangent linear model the emissivity perturbation is calculated within RTTOV when using FASTEM or when calling PC-RTTOV for sea surfaces with **calcemis(:)** set to true. Otherwise you may optionally specify input perturbations in **emissivity_tl(:)%emis_in** or set this array to zero. 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.

RTTOV also 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. The IR and MW emissivity atlases supplied with RTTOV v11 provide monthly climatological land surface emissivity values. The IR atlas also calculates emissivity values for sea-ice surfaces and for land surfaces with a non-zero snow cover fraction. The IR atlas can optionally return estimated errors (standard deviations) for each emissivity, and also, optionally, a quality control flag. New in v11.3, the IR atlas also optionally provides a zenith angle correction for the emissivities. TELSEM, the MW atlas and interpolator, can optionally return estimated errors (standard deviations) for each emissivity, or alternatively a full error covariance matrix for emissivities in all channels. An alternative MW atlas (called the CNRM MW atlas) is also provided for AMSU-A, AMSU-B and MHS. The user is referred to the atlas documentation Borbas *et al.* (2010) for the IR atlas, Aires *et al.* (2010) for TELSEM, and Karbou *et al.* (2006) for the CNRM MW atlas which provide more details, including these various outputs. The data files for these atlases are available from the NWP SAF website.

A common interface is provided to both IR and MW atlases: the atlas used depends on the type of instrument defined in the **coefs** structure. There are three subroutines the user must call to use the atlases:

- **rttov_setup_emis_atlas** : this allocates the necessary arrays and reads the atlas data appropriate to the sensor into them. This should be called after the **coefs** structure has been initialised.
- **rttov_get_emis** : this returns surface emissivity values at a given latitude/longitude in the required channels. This should be called when populating the **emissivity(:)%emis_in** array for input to RTTOV.
- **rttov_deallocate_emis_atlas** : this carries out deallocation of arrays and should be called once the atlas is no longer required.

Each atlas uses a number of variables from the input profile. Table 15 lists the profile variables used by each of the emissivity and BRDF atlases.

The interfaces of these subroutines are described in Annex F. Note that if called for a sea surface type or if the atlas has no data for the given latitude/longitude (i.e. the location is not land according to the atlas), zero or negative values will be returned. The user should check the emissivities to ensure that spurious values are not inadvertently passed in to RTTOV. An example of using the atlas subroutines with a forward model call can be found in the program `src/brdf_atlas/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} / sea=ε_{PC} for PC-RTTOV	No ∂ε calculated, except for PC-RTTOV where sea ∂ε, computed from ∂u, ∂v about ε_{PC}.
	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}	Land/sea-ice ∂ε about ε_{FASTEM} sea ∂ε, computed from ∂u, ∂v, ∂sst about ε_{FASTEM}
	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 gradient 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. Over sea-surfaces RTTOV calculates the reflectance due to sun-glint as described in Matricardi (2003). For land/sea-ice surfaces the BRDF is calculated as (1-*emissivity*)/ π if a valid emissivity has been supplied to or calculated by RTTOV, otherwise the surface reflectance is set to 0.3/π or 0.8/π for land and sea-ice surface types respectively.

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.

For channels where **calcrefl(:)** is true, the tangent linear model surface reflectance perturbation is calculated within RTTOV for sea surfaces and for other surface types when a valid **emissivity_tl** was calculated. Otherwise you may optionally specify input perturbations in **reflectance_tl(:)%refl_in** or set this array to zero. The reflectance perturbations used in the TL calculations are output in **reflectance_tl(:)%refl_out**.

RTTOV provides a BRDF atlas (Vidot and Borbas, 2013) which is similar in many ways to the IR emissivity atlas. It provides monthly climatological land surface BRDF values based on the MODIS BRDF kernel product. For water surfaces the atlas can also provide BRDF values away from sun glint-affected regions. The BRDF atlas can also optionally return the bi-hemispherical (black-sky) albedo and a quality flag. 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.

The interface to the BRDF atlas is similar to the emissivity atlases:

- **rttov_setup_brdf_atlas** : this allocates the necessary arrays and reads the atlas data into them. This should be called after the **coefs** structure has been initialised.
- **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** : this carries out deallocation of arrays and should be called once the atlas is no longer required.

The BRDF uses a number of variables from the input profile. Table 15 lists the profile variables used by each of the emissivity and BRDF atlases.

The interfaces of these subroutines are described in Annex G. For a limited number of locations there is missing data in the underlying BRDF dataset due to persistent cloud cover. In locations where the atlas has no values a negative BRDF

is returned. The user should check the BRDFs to ensure spurious values are not passed into RTTOV. An example of using the atlas subroutines with a forward model call can be found in the program `src/brdf_atlas/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.

TELSEM MW atlas	latitude, longitude, zenangle, skin%surftype
CNRM MW atlas	latitude, longitude, zenangle, skin%surftype
IR emissivity atlas	latitude, longitude, skin%surftype, snow_frac 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 for the user 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 described in Annex D. The trajectory structure consists of a number of structures used internally by RTTOV. The user 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.

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. It is recommended that users who are interested in minimising run-time carry out a test on their 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 v11

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,
radiancedata, radiancedata2, 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 **radiancedata** of the derived types *transmission_type* and *radiance_type* should be declared. Optionally an instance **radiancedata2** of type *radiance2_type* may be declared. The **radiancedata** structure contains the primary radiance, BT and reflectance outputs (the output reflectance quantity is defined in section 8.2) and is mandatory, while the **radiancedata2** 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 16 and 17 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 standard pressure levels defined by the user *except* for the layer containing the surface where the value is the radiance calculated from/to the 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 IR aerosol or PC simulations. Also note that in the case of IR aerosol simulations, the "clear-sky" outputs (such as **rad%clear** and **rad%bt_clear**) contain the aerosol-affected radiances.

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, the user 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.

Radiance_Type 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 given black cloud at the level bounding the bottom of each layer. For solar 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 IR aerosol simulations.

Radiance_Type 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)

Radiance_Type 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).

Transmission_Type 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 16. Main RTTOV v11 output arrays. The green rows are those most commonly used. See Annex O for more details about the output transmittances.

Radiance2_Type 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 atmsosphere for layers down to the level bounding the bottom of each layer.
real	down(nlayer,nchanprof)	Summed downwelling atmospheric emission term at bottom of atmsosphere 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 17. Secondary RTTOV v11 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 usual atmospheric profile and surface parameters as input
- 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.

For Principal Components calculations over sea where **calcemis(:)** is TRUE and where the FASTEM model is used the surface emissivity is calculated from the input profile and so RTTOV will calculate the surface emissivity perturbation from variables in **profiles_tl**. In all other cases, the surface emissivity perturbation may be specified manually in **emissivity_tl(:)%emis_in** if required. The emissivity perturbation used in the TL calculation is written to **emissivity_tl(:)%emis_out**.

Similarly, for solar calculations where **calcrefl** is true, RTTOV will calculate the surface BRDF perturbation from variables in **profiles_tl** for sea surfaces. Over land and sea-ice surfaces the BRDF perturbation will be calculated from the emissivity TL if there is a valid input emissivity. In all other cases the BRDF perturbation may be specified in **reflectance_tl(:)%refl_in**. 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 **radiancedata_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 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 **radiancedata_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 **radiancedata_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 **radiancedata_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 **radiancedata_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 **radiancedata_ad/k%total(:)** (in units of radiance), otherwise the input perturbations must be specified in **radiancedata_ad/k%bt(:)** (in Kelvin). For "solar" 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)
- **radiancedata_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** structure (set both **emis_in(:)** and **emis_out(:)** member arrays to zero)
- **reflectance** structure (set both **refl_in(:)** and **refl_out(:)** member arrays to zero)

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

On output the **emissivity_ad/k(:)%emis_in** and **reflectance_ad/k(:)%refl_in** arrays contain the AD/Jacobians for the surface emissivity and BRDF respectively.

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 **switchrad** flag is always true and the input perturbations are always in terms of brightness temperature. In general 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 **radiancedata_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.

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.

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. Summary of steps for running RTTOV v11

You need to ensure the following in your program which calls RTTOV v11:

- 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). This can be achieved with the allocation routines described in Annex D. If the emissivity and/or BRDF atlases are required these should be initialised now (see Annexes F and G). See Annex Q for example code detailing the input variables required for RTTOV v11.
- 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 the land surface emissivity atlas), in which case you have to set **calcemis(j)** to false for the desired channels. Alternatively you may let the code compute it by the use of the models ISEM (IR over ocean) and FASTEM (MW) 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 should 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).
- All user-level RTTOV routines 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/near-IR/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 simuilations, 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. 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

New in RTTOV v11 is the capability to carry out simulations for visible and near-infrared 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 84°. Figure 4 illustrates the definition of azimuth angle.

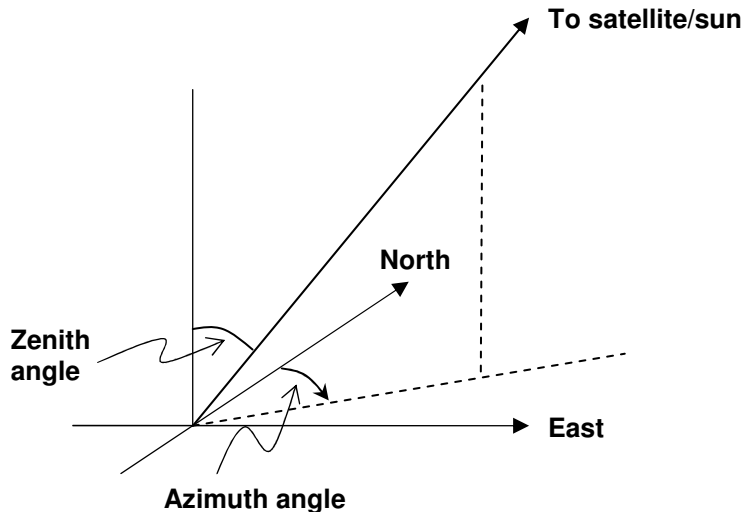

Figure 4. Azimuth angles are defined as the angle in the clockwise direction between North and the projection of the view-path or sun-surface path onto the horizontal plane at the surface.

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 users 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 described in Matricardi (2003). To make use of this 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.

If `calcrefl(:)` is set to true for land or sea-ice surfaces a default BRDF of 0.3/π or 0.8/π (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-*emissivity*)/π.

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 τ_{Cld} (ν,θ) 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 parameterisation has 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 `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 model

RTTOV incorporates the FASTEM emissivity model for MW surface emissivities. There are now 6 versions of FASTEM. FASTEM-6 has a much improved azimuthal dependence and is now the generally recommended option, although FASTEM-5 remains the default. The FASTEM options are summarised in Table 18.

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 ***(default)***	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	As FASTEM-5, but with an improved azimuthal dependence (Kazumori and English, 2014). NB FASTEM-6 extrapolates the azimuthal dependence for frequencies above 90GHz so should be used with some caution for these channels. Also, no azimuthal dependence is simulated for the 3rd and 4th Stokes channels on Windsat.

Table 18. Summary of FASTEM versions.

The version of FASTEM to use (1 to 6) is set in the `opts%rt_mw%fastem_version` variable. If this is set to a value other 1-6 the FASTEM version will be taken from the instrument coefficient file. Note that FASTEM should not be used with channels on new instruments like MetopSG ICI which lie beyond the range of frequencies over which FASTEM has been trained (in particular channels above 200 GHz).

The FASTEM models all compute a surface emissivity for the channel of interest at the given zenith angle. Using FASTEM requires the 10m wind speed to be provided in the state vector. FASTEM versions 4-6 also make use of the input `profiles(:)%skin%salinity` variable.

FASTEM also by default calculates the fractional ocean foam cover which is used in the emissivity calculation. Since v11.3, RTTOV allows you to specify the input foam fraction manually in the `profiles(:)%skin%foam_fraction` variable. To enable this the `opts%rt_mw%supply_foam_fraction` flag must be set to true (it is false by default).

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, TMI, AMSR, Windsat
4	Horizontal **H**	SSM/I, SSMIS, TMI, AMSR, Windsat
5	+ 45 minus −45 (3rd stokes vector) **S3**	Windsat
6	Left circular - right circular (4th stokes vector) **S4**	Windsat

Table 19. 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 20. Values for FASTEM parameters for various land and sea-ice surface types. Taken from English and Hewison (1998).

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 19. 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).

FASTEM can also provide 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 20.

IR sea surface emissivities

RTTOV uses the ISEM model (Sherlock, 1999) to compute IR emissivities for sea surfaces. This model parameterises the emissivity by zenith angle only.

PC-RTTOV uses a more physically-based emissivity model than ISEM for sea surface emissivities. This parameterisation is based on the Wu and Smith (1997) model and requires the zenith angle and 10m wind-speed. This model should always be used for PC-RTTOV simulations using the older sea-only PC coefficients, and is recommended (though not strictly mandatory) with the new global "landsea" PC coefficients..

Emissivity atlases

Over land and sea-ice surface types where `calcemis(:)` is true, default values are provided for the surface emissivity in the infrared (0.98 over land, 0.99 over sea-ice) and for the microwave the FASTEM model is used (as described above). For more accurate emissivity estimates over land RTTOV v11 also provides access to IR and MW land surface emissivity atlases. As 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.

For IR instruments the atlas described in Borbas *et al.* (2010) takes as input the latitude, longitude and month, and provides climatological emissivity values for the specified instrument channels for land surfaces. This atlas can optionally provide an estimate of the error in the surface emissivity. The IR atlas can also return emissivity values for land surfaces with fractional snow cover and for sea-ice surfaces. As of v11.3 the atlas also optionally provides a correction for varying zenith angle. This is selected when the atlas is initialised. For more information see Borbas (2014). The new PC-RTTOV "landsea" coefficients were trained using this atlas and as such it is recommended (though not strictly mandatory) to use the IR atlas for input emissivities for land and sea-ice surface types with the new coefficient files.

For MW instruments, the TELSEM atlas and interpolator (Aires *et al.* 2010) provides land surface emissivities and, optionally, a full error covariance matrix for the specified channels. The TELSEM atlas datasets are specifically intended for use with the TELSEM interpolator through the appropriate RTTOV subroutine (`rttov_get_emis`), rather than for stand-alone use. The TELSEM interpolator is designed for frequencies between 19 and 85 GHz, but has been found to be beneficial for frequencies between 10 and 190 GHz (Aires *et al.* 2010). Note that TELSEM should not be used with channels on new instruments like MetopSG ICI which lie beyond the range of frequencies over which TELSEM has been trained (above 200GHz).

A second emissivity atlas is available for selected MW instruments. The CNRM MW atlas provides land surface emissivity values for AMSU-A, AMSU-B and MHS. It is described in Karbou *et al.* (2006).

8.5. Simulation of cloudy radiances

This 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 a new multiple scattering radiance simulation code within RTTOV has been developed and is described in the RTTOV v9 science and validation plan. Note this is different from the multiple scattering code developed for microwave precipitation described in section 8.7. This internal multiple scattering code for the infrared uses a different approach, in that scattering effects are parametrised rather than treated explicitly, and it is currently only intended for simulating cloud-affected radiances for infrared sensors such as SEVIRI, AIRS and IASI. The scattering parameterisation may also be applied to visible and

near-infrared channels with `opts%rt_ir%addsolar` set to true, but users should be aware that errors will be large for solar scattering simulations when there is strong scattering as is the case with water clouds for example.
RTTOV v11 provides two options for invoking the multiple scattering scheme:

Method 1: specify number density profiles for the pre-defined particle types listed in Table 22. For ice clouds users have a choice regarding the parameterisation of the optical properties of the ice particles (this is similar to RTTOV v10).

Method 2: supply profiles of the cloud scattering optical properties for each instrument channel directly. This provides greater flexibility as the user is not limited to the pre-defined particle types, but the calling sequence is slightly more complicated.

For both methods 1 and 2

The user should supply a cloud fraction profile in `profiles(j)%cfrac(:)` for each profile `j` (note the cloud fraction is specified on layers rather than levels and the same applies to other cloud inputs). 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 number density. The remaining fraction of the layer is assumed to be clear.

When running the tangent linear (TL), adjoint (AD) or K models, users 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 users 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.

Method 1 details

Invoking the multiple scattering scheme within RTTOV requires additional inputs to RTTOV v11 as detailed below. The cloud profile arrays `profiles%cloud(i,j)` are 2 dimensional (index, layers) where the index refers to 6 different cloud types as defined in Table 22. The first 5 are water clouds and the sixth is ice cloud. The user must fill the required cloud type column with liquid water/ice concentration values in units of g m^{-3}. A non-zero concentration can be given for any combination of cloud types in each layer. Note that the cloud profiles are defined on layers: layer *n* lies between levels *n* and *n+1*. As noted above, the `cfrac` array specifies the combined cloud fraction for all cloud types present in the layer. See above for some caveats on the specification of cloud fraction for the TL, AD and K models. An example input cloud and cloud fraction profile is given in file:
`rttov_test/test_example_cld_file_fwd.1/cld_prof.dat`.

For water clouds optical parameters are available for five size distributions corresponding to five different cloud types. There are a number of options for ice clouds (also summarised in Table 21):

- Select a parameterisation for the effective diameter of the size distribution in `profiles%idg` (see Table 21) to be used in the calculation of the optical parameters. Coefficients are available for two ice crystal shapes: the shape should be specified by setting `profiles%ish` to 1 or 2 for hexagonal or aggregate crystals respectively.
- Specify the ice effective diameter explicitly in `profiles%icede(:)` (units: microns) for the calculation of the optical parameters. For layers where ice cloud is present in `profiles%cloud(6,:)`, the effective diameter specified in `icede(:)` will be used for that layer where this value is greater than zero. If `icede(:)` is zero for the layer, the parameterisation specified by `profiles%idg` is used.
- Use the new Baran parameterisation for the optical parameters by setting `profiles%ish` to 3 or 4. This calculates the optical parameters using the Baran scheme which parameterises the particle properties in terms of temperature and ice water content. This is described in the RTTOV v11 science and validation report; `profiles%idg` is ignored in this case. RTTOV v11.2 and later include an improved parameterisation which is selected using `ish`=4.

Users should be aware that the coefficients for the ice particle parameterisations have been trained over specific ranges of ice water content (IWC) and effective diameter (for hexagonal and aggregate crystals) and over specific ranges of

IWC and temperature (for the Baran scheme). Input values beyond these ranges can result in unphysical values for the particle optical properties. Therefore 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 hexagonal/aggregate crystals: 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. The parameter limits are given in Table 21.

The coefficients for calculating the optical parameters for each particle type are contained in cloud coefficient files whose filenames begin "sccldcoef_" by convention. It is important to note that the channels represented in the cloud coefficient file **must** be the same as the channels represented in the optical depth ("*rtcoef_*") coefficient file. Cloud coefficient files are not available for visible/near-IR channels so it is not possible to use the cloud coefficient files with the "v9 predictor" optical depth coefficient files for instruments with visible and near-IR channels. If this is desired, it is necessary to extract just the IR channels from the optical depth coefficient file first using the `rttov_conv_coef.exe` executable described in Annex A.

As detailed in the scientific and validation report, the computation of cloud affected radiances is performed by dividing the computed atmospheric path into a number of independent streams. The number of streams used for the scattering calculation is computed internally in `rttov_cldstr`. It is possible to reduce the number of streams and save time by considering only those streams whose weight is larger than the variable `opts%rt_ir%cldstr_threshold`. By setting `cldstr_threshold` to a negative number, all the streams will be processed (the default). This feature should be used with caution. Since the sum of the weights of all streams (including the clear one) must be equal to 1, if some 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. The user should select the value of `cldstr_threshold` in order to remove only the streams with a very small weight. These 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 tangent linear, adjoint or Jacobian output.

To compute cloudy radiances via method 1 the user must:

- Set `opts%rt_iraddclouds` 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_meteosat_5_mviri.dat* where MVIRI on Meteosat-5 is the sensor in this case.
- Populate the input `profiles(:)%cloud(:,:)` array with layer mean liquid or ice water/ice concentration in $g.m^{-3}$ 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 ice particles according to the three options in Table 21.

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 the users own programs.

Baran scheme	**Explicitly specify ice effective diameter**	**Use a parameterisation to convert ice water content to effective diameter.**
Set *profiles(:)%ish* = 3 or 4 **3 =>** original Baran parameterisation (RTTOV v11.1) **4 =>** improved Baran parameterisation (RTTOV v11.2 and later)	Set *profiles(:)%ish* 1 = Hexagonal 2 = Aggregates Set *profiles(:)%icede(:)* for the ice effective diameter in microns for each layer.	Set *profiles(:)%ish* 1 = Hexagonal 2 = Aggregates Set *profiles(:)%idg* 1 = Ou and Liou (1995) 2 = Wyser (1998) 3 = Boudala et al (2002) 4 = McFarquar et al (2003) Ensure *profiles(:)%icede(:)* is zero.
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}	Hexagonal: - Min eff. diameter = 12.2 µm - Max eff. diameter = 118.29µm - Min IWC = 0.000608 g m^{-3} - Max IWC = 0.254639 g m^{-3} Only effective diameter limits are imposed in the code.	Aggregates: - Min eff. diameter = 5.61 µm - Max eff. diameter = 166.46 µm - Min IWC = 0.000235 g m^{-3} - Max IWC = 0.489046 g m^{-3}

Table 21. 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:	Cirrus	(CIRR)

Table 22. Cloud types available in RTTOV v11.

Method 2 details

It is also possible to supply vertical profiles of the cloud particle optical parameters used by the scattering parameterisation directly to RTTOV for each channel. In this case the `profiles%cloud` array is not required and instead users must supply the `cld_opt_param` argument (of derived type `rttov_opt_param`) when calling RTTOV. This comprises arrays for each scattering parameter required by RTTOV for each layer and channel. The members of the `rttov_opt_param` structure are:

- `abs(:,:)` : absorption coefficient (units: km^{-1}), dimensions (nchannels,nlayers).
- `sca(:,:)` : scattering coefficient (units: km^{-1}), dimensions (nchannels,nlayers).
- `bpr(:,:)` : "*b* parameter" (no units) – this represents the fraction of back-scattered radiation from each layer and is calculated from the phase functions via a supplied subroutine (see below), dimensions (nchannels,nlayers).
- `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.
- `pha(:,:,:)` : phase function – these are required for the calculation of the *b* parameter, and for solar simulations the phase functions are required in the scattering calculations for solar-affected channels, dimensions (nchannels,nlayers,nphangle). Phase functions should be normalised such that the integral over all scattering angles is 4π.

A variable `cld_opt_param` of type `rttov_opt_param` should be declared and then allocated by calling `rttov_alloc_opt_param` (see Annex E). (This subroutine should also be called to deallocate the structure at the end of your program). The absorption and scattering coefficients, phase angles and phase functions should be populated in the `abs`, `sca`, `phangle` and `pha` member arrays respectively. The pre-defined RTTOV particle types use phase functions defined over a fixed set of 208 angles. These angles can be found in the `phangle(:)` array in `rttov_const.F90` and users may find it convenient to use this set of angles for their own programs. Alternatively users can specify their own set of phase angles.

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. Users may find it beneficial to calculate the *b* parameters off-line and store them for future use if this is practical for their application. To generate the values users should:

- 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

The interfaces for these subroutines are described in Annex E.
Note that if users have pre-prepared values for `bpr` and 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, the user must call the `rttov_init_opt_param` subroutine (see Annex E) 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 the user must:

- 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 the phase function and phase angles
- Populate `cld_opt_param` with the *b* parameters either from pre-calculated data or by calling `rttov_bpr_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`)
- When finished with RTTOV call `rttov_alloc_opt_param` again to deallocate `cld_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 the users own programs. The procedure is identical for cloudy simulations except that the cloud fraction must also be specified in the cloudy case.

Method 2 must be used for scattering calculations in visible/near-IR channels, but coefficients are available for solar calculations in short-wave IR channels of the hi-res sounders.

8.6. Simulation of aerosol affected radiances

The multiple scattering parameterisation is capable of simulating the effects of aerosols at infrared, near-infrared and visible wavelengths. Note that the errors in the simulations are larger at shorter wavelengths as solar radiation violates the assumptions of the scattering scheme. Errors are particularly large in cases where the single-scattering albedo approaches 1.0 (strong scattering) and in forward scattering geometries (i.e. where the satellite and solar zenith angles are similar in value and the relative azimuth is close to zero). The RTTOV v11 science and validation report provides more details.

As with cloud scattering, there are two ways to invoke the aerosol scattering calculations in RTTOV v11. It is important to note that due to the way in which the aerosol-affected radiances are calculated, the "clear-sky"outputs in the *radiance_type* structure (e.g. `radiance%clear`, `radiance%bt_clear`, `radiance%refl_clear`) include the effects of aerosol when these calculations are performed. This is true for both methods of invoking the aerosol scattering.

Method 1

The first option is to specify number density profiles of pre-defined aerosol particle types (as in RTTOV v10). The methodology is described in the RTTOV v9 science and validation plan. Aerosol simulations require `opts%rt_ir%addaerosl` to be set to true and the `profiles%aerosols(:,:)` array to be populated with number density profiles for each particle type: the first index of the array is the aerosol particle type (1-13, see Table 23) and the second index is the profile layer (*not* level). The mixing of the various aerosol components can be defined by the user or climatological profiles with predefined mixing can be supplied. To include an aerosol component in the radiative transfer, the user must assign the layer mean density (in units of cm^{-3}) for that component. An example input aerosol profile is given in file: `rttov_test/test_example_aer_file_fwd.1/aer_prof.dat`.

Alternatively if a climatological profile is desired (see RTTOV v9 science plan for more details on profiles) the program `src/other/create_aer_clim_prof.F90` (see Annex N) can be used to generate climatological aerosol profiles. 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. Users may wish to re-run the executable to generate their 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.

Column 1:	Insoluble	INSO
Column 2:	Water soluble	WASO
Column 3:	Soot	SOOT
Column 4:	Sea salt (acc mode)	SSAM
Column 5:	Sea salt (coa mode)	SSCM
Column 6:	Mineral (nuc mode)	MINM
Column 7:	Mineral (acc mode)	MIAM
Column 8:	Mineral (coa mode)	MICM
Column 9:	Mineral transported	MITR
Column 10:	Sulphated droplets	SUSO
Column 11:	Volcanic ash	VOLA
Column 12:	New volcanic ash	VAPO
Column 13:	Asian dust	ASDU

Table 23. Aerosol components available in RTTOV v11.

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).

The coefficients for calculating the optical parameters for each particle type are contained in aerosol coefficient files whose filenames begin "scaercoef_" by convention. It is important to note that the channels represented in the aerosol coefficient file **must** be the same as the channels represented in the optical depth ("*rtcoef_*") coefficient file. Aerosol coefficient files are not available for visible/near-IR channels so it is not possible to use the aerosol coefficient files with the "v9 predictor" optical depth coefficient files for instruments with visible and near-IR channels. If this is desired, it is necessary to extract just the IR channels from the optical depth coefficient file first using the `rttov_conv_coef.exe` executable described in Annex A.

To compute aerosol-affected radiances via method 1 the user must:

- 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_meteosat_5_mviri.dat* where MVIRI on Meteosat-5 is the sensor in this case.
- Populate the input `profiles(:)%aerosol(:,:)` array with mean layer aerosol concentration in cm^{-3} 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 the users own programs.

Method 2

The second method is to supply profiles of the aerosol scattering optical parameters for each channel directly. The same parameters are required as for the cloud scattering and the *b* parameter can be generated in exactly the same way (see section 8.5). To use this method both `opts%rt_ir%addaerosl` and `opts%rt_ir%aer_opt_param` should be set to true and the `aer_opt_param` argument should be supplied when calling RTTOV. No aerosol coefficient file is required in this case and the `profiles%aerosol(:,:)` array does not need populating.

To compute aerosol-affected radiances via method 2 the user must:

- 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 the phase function and phase angles
- Populate `aer_opt_param` with the *b* parameters either from pre-calculated data or by calling `rttov_bpr_calc`
- If performing solar calculations call `rttov_init_opt_param`
- Pass the `aer_opt_param` argument into `rttov_direct` (or `_tl/ad/k`)
- 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 the users own programs.

Method 2 must be used for scattering calculations in visible/near-IR channels, but coefficients are available for solar calculations in short-wave IR channels of the hi-res sounders.

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 package described in section 8.5 is completely different, in that scattering effects are parametrised rather than treated explicitly, and it is for the moment only intended for simulating cloud-affected 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 science and validation report.

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:

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

Here, C is the effective cloud fraction in the vertical profile and T_B is brightness temperature. The clear-air RTTOV is called from within RTTOV-SCATT and returns the brightness temperature of the clear sky column, T_B^{Clear}, and the profile of clear sky transmittances. RTTOV-SCATT then computes the cloudy or rainy brightness temperature, T_B^{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 brightness temperature T_B^{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. `profile_type`; Table 12 and section 7.3) but additional information is required, principally hydrometeor profiles, supplied in `profile_cloud_type` and listed in Table 24. The `use_totalice` logical variable in `profile_cloud_type` should be set to false for separate ice and snow and to true for total ice. These two options are mutually exclusive.

New in RTTOV v11 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]. Within the `profile_cloud_type` 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 and the top half level is the top of the atmosphere. Full pressure levels are those supplied in `profile_type` (in `profiles(:)%p(:)`), but the half level pressures need to be supplied in `profile_cloud_type`. Figure 5 shows the arrangement of full and half levels.

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 24. RTTOV-SCATT profile variables for `profile_cloud_type`

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., the user can supply their 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.

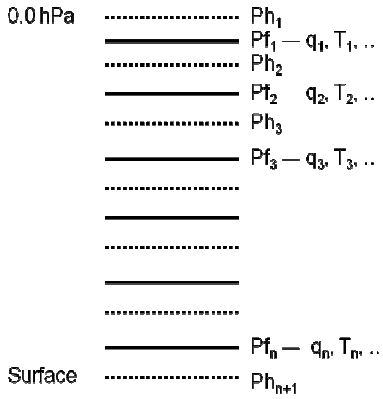

Figure 5. RTTOV-SCATT full and half levels, showing half and full level pressure (Ph,Pf) and examples of the variables specified on full levels (e.g. q, T, but also cloud and hydrometeors)

Mie coefficients

RTTOV-SCATT relies on both clear-sky coefficent files (e.g. `rtcoef_noaa_15_amsua.dat`) and precomputed tables of Mie scattering parameters (e.g. `mietable_noaa_amsua.dat`). Due to their large size, the Mie files for microwave sensors are not supplied in the tar file, but are provided on the RTTOV web page 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.

The user may also create their 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 scores

A principal component (PC) based version of RTTOV is available for the simulation of the full IASI and AIRS radiance spectrum 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 v11 a logical flag `opts%rt_ir%pc%addpc` is set to true.

The RTTOV optical depth regression coefficients must be compatible with PC-RTTOV: this is currently true only of the 101-level v9 predictor coefficient files. If an incompatible coefficient file is used an error will result. The format of the PC coefficient files has been updated for v11 so RTTOV v10 files are not compatible. PC coefficient files have filenames beginning "`pccoef_`" and are available for download from the RTTOV v11 website. They are read in at the same time as the optical depth coefficient file in the call to `rttov_read_coefs`.

The RTTOV v9 predictor optical depth coefficient files allow for the variation of CO_2, N_2O, CO and CH_4. However, in the LBL computations for PC-RTTOV the concentration of these species is fixed and consequently the same fixed values must be used when calling PC-RTTOV. For this reason, the CO_2, N_2O, CO and CH_4 values specified in the RTTOV input state vector are replaced by the constant values stored in the principal components coefficient file. This means that the RTTOV K model returns a value of the gradient of the radiances with respect to the state vector of these trace species equal to zero. Finally, the computation of the predictors mandates the use of the linear-in-tau approximation with the inclusion of the effects due to the presence of atmospheric refraction. Therefore refraction is accounted for regardless of the setting of `opts%rt_all%addrefrac`.

PC coefficient files available prior to the release of RTTOV v11.3 were trained only for sea surfaces and as such must only be used for such profiles with `calcemis(:)` set to true. As of RTTOV v11.3 new global 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 and to use the IR land surface emissivity atlas (see section 7.5) for land and sea-ice surface types as this was the configuration 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 v11.3 carries out no checks on how surface emissivity is specified for PC-RTTOV: you must ensure that you do not use the older sea-only PC coefficients for land surfaces.

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 choice of predictor set is selected in `opts%rt_ir%pc%ipcreg`. Table 25 gives the valid values for the PC-RTTOV coefficients available at the time of the RTTOV v11.3 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 from the PC coefficient structure in `coefs%coef_pccomp%pcreg`. An example of this can be seen in `src/test/example_pc_fwd.F90` which can be used as a model for your own code. An alternative way of obtaining the channel list is via the subroutine `rttov_get_pc_predictindex` (Annex H). 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 for the new "landsea" PC-RTTOV coefficients are different to those for the old "sea-only" 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.

Coefficient file	`opts%rt_ir%pc%ipcbnd`	`opts%rt_ir%pc%ipcreg`	`Max npcscores`
IASI – sea-only and global	1 => full spectrum (all channels)	1, 2, 3 or 4 => 300, 400, 500 or 600 predictors respectively	400
AIRS – sea-only and 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 25. Available options for band and predictor sets forPC coefficient files available at the time of the RTTOV v11.3 release.

The number of simulated principal components can vary from 1 to 400 (IASI, AIRS) or 1 to 600 (IASI-NG). This is 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.

The computed PC scores are stored in the `pccomp` structure (see Annex O). 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 must be passed in the call to `rttov_alloc_pccomp` (Annex D) and the reconstructed channel list `channels_rec(:)` may be supplied to `rttov_direct` (or to the TL, AD or K model; Annex I,J,K,L). The reconstructed radiances are also stored in the `pccomp` structure. The `channels_rec(:)` input array is mandatory if `opts%rt_ir%pc%addradrec` is true.

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 NLTE bias correction, 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.

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.

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 25. The files contain the regression coefficients for the PC-RTTOV sea surface emissivity model (activated by setting `calcemis(:)` to true which is mandatory for older sea-only PC coefficients, and is recommended – but not mandatory – for sea profiles with the new "landsea" coefficients). 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.

8.9. Inclusion of non-local thermodynamic equilibrium effects

RTTOV v11 is now able to estimate the effect that so-called 'solar pumping' has on observed TOA radiances around the CO_2 ν2 band (around 4.3 μm). Coefficient files which include the correction are available for IASI, AIRS and CrIS. NLTE coefficients for other hyperspectral IR sounders can be requested via the NWP SAF helpdesk.

To invoke the NLTE correction it is necessary to:

- Use a coefficient file that contains the NLTE coefficients for the regression.
- Set `opts%rt_ir%do_nlte_correction = .true.`
- Not be using the PC functionality of RTTOV.

The correction is valid (and is applied to channels) between 2225 cm^{-1} – 2400 cm^{-1} which corresponds to:

- IASI channels 6421 – 7021 (601 channels)
- AIRS channels 1939 – 2121 (183 channels)
- CrIS channels 1175 – 1245 (71 channels)
- CrIS-FSR channels 1651 – 1971 (321 channels)

The radiance correction scheme is very similar to the one documented in Chen *et al* (2013) and consists of two predictors; the average temperature between 0.005 hPa and 0.2 hPa, *T1* and the average temperature between 0.2 hPa and 52 hPa, *T2*.

There are six fixed solar zenith angles and 11 satellite viewing angles at which reference calculations were performed to generate coefficients:

θ^{sol} = 0, 40, 60, 80, 85 and 90 degrees

sec(θ^{sat}) = 1, 1.125, 1.25, 1.375, 1.5, 1.75, 2, 2.25, 2.5, 2.75, and 3, corresponding to satellite viewing angles between 0 and ~70.5 degrees.

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

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

where ΔR_{ch}^{NLTE} is bilinearly interpolated from the four corrections (labelled ΔR_{ij}) derived from the closest tabulated satellite viewing angle and solar zenith angle pairs (indexed by *i* and *j* below),

$$\Delta R_{ij} = c_0^{ij} + c_1^{ij} T_1^{avg} + c_2^{ij} T_2^{avg} . \quad (10)$$

The user is responsible for ensuring that input values for these parameters are in range. 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 users should be aware that the NLTE correction is added to the final LTE cloudy/aerosol-affected radiances and so the NLTE radiance is not included in the scattering calculations.

A more detailed discussion of the science and the impact of including the non-LTE correction is available in the RTTOV v 11 science and validation report.

8.10. 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 ~55 deg to be used for the downward radiation. The option of Lambertian reflection over is now possible for land and sea-ice surface types in the MW and, new in v11.3, for all surface types in 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. Note that for MW sensors over sea the FASTEM models do their own non-specular correction and so this option is not applied over the sea if FASTEM is used. More details are provided in the RTTOV v11 science and validation report.

8.11. 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. The user must specify two input variables for the geomagnetic field in the `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. To include the Zeeman effect for a given sensor, the user must run RTTOV v11 with a Zeeman coefficient file.

The profile top for a Zeeman run should extend as far as possible towards the model-top used in generating the coefficients, which will appear as the top-most pressure in the reference profile section of the coefficient file. Where there is significant absorption *above* the user's model top, the interpolation of predicted optical depths back to user levels may return a value at the user model-top significantly different from zero, although zero would be the expected value at the space boundary. For channels with low absorption in the mesosphere, these departures will be negligible, and the user may impose a zero value for the high peaking channels by setting the `opts%interpolation%spacetop` flag to True. With `spacetop` set to False the predicted absorption in the layer of gas above the user's model top will be included automatically in the final integration, but no emission. With `spacetop` set to True, the space boundary will, in effect, be relocated at the user's model top. In terms of the final integration, there will be a uniform extrapolation of the user's top layer to the space boundary, bringing in the predicted absorption as before, but also making some representation of the emission. Note that the setting of `spacetop` affects calculations for all instruments, not just those with a Zeeman coefficient file.

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 the user runs 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 the user runs 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.

8.12. Simulation of SSU radiances

For SSU, which uses pressure modulated gas cells to define the channels, RTTOV v11 allows the user 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 provides 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 new RTTOV v11 pressure modulated cell (PMC) shift coefficients, the user 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. If the user supplies a CO_2 profile in place of the default, this will also feed into the cell pressure scheme.

9. Limitations of RTTOV v11

There are a number of scientific limitations of RTTOV v11 the user should be aware of. The main ones are listed here:

- RTTOV v11 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 v11 only allows for water vapour, ozone, carbon dioxide, nitrous oxide, methane and carbon monoxide to be variable gases with all others included in the mixed gases transmittance calculation.
- RTTOV v11 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 v11.
- 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).
- Principal Component computations are limited by the configuration of the coefficient training. More information is given in section 8.8.
- Scattering calculations involving solar radiation exhibit larger errors than for infrared channels. In particular, while solar simulations for water clouds are possible, the errors are typically too large for the results to be useful. The errors are large for strongly scattering particles (where the single-scattering albedo approaches 1.0) and in forward scattering geometries (where the satellite and solar zenith angles are very similar and the relative azimuth is close to 0 degrees).
- For scattering calculations where the user inputs the optical parameters, the input parameters have no TL/AD/K counterparts (i.e. they are treated as static variables in the model).

10.Reporting and known bugs for RTTOV v11

The procedure to report bugs or make comments on the code to the NWP SAF is as follows:

Send a bug report to nwpsaf@metoffice.gov.uk including the following information:

- RTTOV version number (i.e. 11.3)
- Platform and operating system you are running the code on (e.g. HP, Sun, LINUX PC)
- 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 with a description of the fix if appropriate. There is also a RTTOV v11 email list which you can subscribe to by sending an email to mailto:nwpsaf@metoffice.gov.uk where bugs are announced. If you request the code and sign a licence agreement you will be automatically included on this list.

The known issues in the version RTTOV v11.3 of the code are given below. Any further problems and corrections will be provided via the RTTOV v11 web page as they become known:
http://nwpsaf.eu/deliverables/rtm/rtm_rttov11.html.

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 v11 has not been tested on this architecture, but similar issues may be expected.

ii. RTTOV v11 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.

11.Frequently asked questions

This section has now been put on the RTTOV v11 web site to allow updating:
http://nwpsaf.eu/deliverables/rtm/rttov11_faq.html

12.Glossary

AD	RTTOV Adjoint model
AMSU	Advanced Microwave Sounding Unit
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
CNRM	Centre National de Recherches Météorologiques
ECMWF	European Centre for Medium-Range Weather Forecasts
EUMETSAT	European Organisation for the Exploitation of Meteorological Satellites
FASTEM	MW surface emissivity model
HDF5	Hierarchical Data Format version 5
IR	Infrared
ISEM	IR sea surface emissivity model
IWC	Ice Water Content
K	RTTOV Jacobian model
LBLRTM	Line-By-Line Radiative Transfer Model used to generate RTTOV coefficients for IR sensors
Liebe-89 MPM	Line-by-line model used to generate RTTOV coefficients for MW sensors
MODIS	Moderate resolution Imaging Spectroradiometer
MHS	Microwave Humidity Sounder
MW	Microwave
NetCDF	Network Common Data Form
NIR	Near-Infrared (see VIS below)
NLTE	Non Local Thermodynamic Equilibrium
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
TELSEM	A Tool to Estimate Land Surface Emissivities at Microwave frequencies
TL	RTTOV Tangent Linear model
SAF	Satellite Applications Facility
SEVIRI	Spinning Enhanced Visible and Infrared Imager
SSMIS	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.Royal. Meteorol. Soc.,* **137,** 690-699. doi: 10.1002/qj.803.

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

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.* http://nwpsaf.eu/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.* http://nwpsaf.eu/vs_reports/nwpsaf-mo-vs-050.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.

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

Deblonde, G., 2000. Evaluation of FASTEM and FASTEM2, *NWP SAF report.* NWPSAF-MO-VS 1. Available here: http://nwpsaf.eu/deliverables/rtm/rtm_reports.html

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

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.

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/media/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.*, vol. **132**, No. 620, Part A, pp. 2333-2355(23), doi :10.1256/qj.05.216

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

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.*

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

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.

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

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

Sherlock, V. 1999 ISEM-6: Infrared Surface Emissivity Model for RTTOV-6. *NWP SAF report.* http://nwpsaf.eu/deliverables/rtm/papers/isem6.pdf

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

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.

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_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
     --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 reals in HDF5 32 bits, default is the RTTOV 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 ...
```

3. 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.

4. RTTOV789_CONV_COEF.EXE

The program **rttov789_conv_coef.exe** (located in the `bin/` directory) can be used to convert v7-, 8- and 9-format coefficient files to the v11 format. The input file must be in ASCII format.

The usage is as follows:

```
$ rttov789_conv_coef.exe --coef-in ...  --coef-out ...
```

Argument	Description
--coef-in	Input ASCII v7/8/9 RTTOV coefficient file.
--coef-out	Output ASCII v11-compatible RTTOV coefficient file.

5. RTTOV789_CONV_COEF_11TO9.EXE

The program **rttov789_conv_coef_11to9.exe** (located in the `bin/` directory) can be used to convert v10/11-format coefficient files to v9 format. The input file must be in ASCII format.

The usage is as follows:

```
$ rttov789_conv_coef_11to9.exe --coef-in ...  --coef-out ...
```

Argument	Description
--coef-in	Input ASCII v10/11 RTTOV coefficient file.
--coef-out	Output ASCII v9-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 computers the standard error is 0, but for HP it is 7. The user should set the value according to his 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)
```

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, but note that all input files (optical depth, cloud, aerosol, PC) must be in the same format.

The user 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.

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.

When calling PC-RTTOV, 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).

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 IR aerosol scattering coefficient file.
Character	Intent(in), optional	form_sccld	Format of 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 IR aerosol scattering coefficient file.
Character	Intent(in), optional	file_sccld	Name of 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 IR aerosol scattering coefficient file.
Integer	Intent(in), optional	file_id_sccld	Logical unit of 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.

2. RTTOV_DEALLOC_COEFS interface

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

rttov_dealloc_coefs is called in the user's program 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.

3. RTTOV_READ_SCATTCOEFFS interface

```
call rttov_read_scattcoeffs (err, coef_rttov, coef_scatt, file_id)
```

This subroutine is called to read the RTTOV-SCATT Mietable coefficients file into the **rttov_scatt_coef** structure. The filename of the Mietable file is constructed from the name of the platform and instrument in the RTTOV coefficient file. It is possible to open the Mietable file before calling this subroutine and to pass the logical unit into **rttov_read_scattcoeffs**. Note that the **coef_rttov** argument requires the **coefs%coef** structure to be passed in (i.e. the RTTOV optical depth coefficients).

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_coef)	Intent(in)	coef_rttov	RTTOV optical depth coefficients structure (i.e. **coefs%coef**).
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.

4. RTTOV_DEALLOC_SCATTCOEFFS interface

```
call rttov_dealloc_scattcoeffs (coef_scatt)
```

rttov_dealloc_scattcoeffs is called in the user's program 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,
              radiancedata,
              radiancedata2,
              calcemis,
              emissivity,
              calcrefl,
              reflectance,
              aer_nphangle,
              aer_opt_param,
              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(rtov_chanprof)	**Pointer, optional**	chanprof(:)	chanprof array pointer: (de)allocated after call
Type(rttov_options)	Intent(in)	opts	RTTOV options structure
Type(profile_type)	**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(transmission_type)	Intent(inout), optional	transmission	Transmission structure to be (de)allocated
Type(radiance_type)	Intent(inout), optional	radiance	Radiance structure to be (de)allocated
Type(radiance2_type)	Intent(inout), optional	radiance2	Secondary radiance structure to be (de)allocated
Logical	**Pointer, optional**	calcemis(:)	calcemis array pointer: (de)allocated after call
Type(rtov_emissivity)	**Pointer, optional**	emissivity(:)	Surface emissivity array pointer: (de)allocated after call
Logical	**Pointer, optional**	calcrefl(:)	calcrefl array pointer: (de)allocated after call
Type(rtov_reflectance)	**Pointer, optional**	reflectance(:)	Surface reflectance array pointer: (de)allocated after call
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_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,
               radiancedata,
               radiancedata_tl,
               calcemis,
               emissivity,
               emissivity_tl,
               calcrefl,
               reflectance,
               reflectance_tl,
               aer_nphangle,
               aer_opt_param,
               cld_nphangle,
               cld_opt_param,
               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(profile_type)	**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(transmission_type)	Intent(inout), optional	transmission_tl	Transmission TL structure to be (de)allocated
Type(radiance_type)	Intent(inout), optional	radiance_tl	Radiance TL structure to be (de)allocated
Type(rtov_emissivity)	**Pointer, optional**	emissivity_tl(:)	Surface emissivity TL array pointer: (de)allocated after call
Type(rtov_reflectance)	**Pointer, optional**	reflectance_tl(:)	Surface reflectance TL array pointer: (de)allocated after call
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,
               radiancedata,
               radiancedata_ad,
               calcemis,
               emissivity,
               emissivity_ad,
               calcrefl,
               reflectance,
               reflectance_ad,
               aer_nphangle,
               aer_opt_param,
               cld_nphangle,
               cld_opt_param,
               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(profile_type)	**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(transmission_type)	Intent(inout), optional	transmission_ad	Transmission AD structure to be (de)allocated
Type(radiance_type)	Intent(inout), optional	radiance_ad	Radiance AD structure to be (de)allocated
Type(rtov_emissivity)	**Pointer, optional**	emissivity_ad(:)	Surface emissivity AD array pointer: (de)allocated after call
Type(rtov_reflectance)	**Pointer, optional**	reflectance_ad(:)	Surface reflectance AD array pointer: (de)allocated after call
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,
               radiancedata,
               radiancedata_k,
               calcemis,
               emissivity,
               emissivity_k,
               calcrefl,
               reflectance,
               reflectance_k,
               aer_nphangle,
               aer_opt_param,
               cld_nphangle,
               cld_opt_param,
               traj,
               traj_tl,
               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(profile_type)	**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(transmission_type)	Intent(inout), optional	transmission_k	Transmission K structure to be (de)allocated
Type(radiance_type)	Intent(inout), optional	radiance_k	Radiance K structure to be (de)allocated
Type(rtov_emissivity)	**Pointer, optional**	emissivity_k(:)	Surface emissivity K array pointer: (de)allocated after call
Type(rtov_reflectance)	**Pointer, optional**	reflectance_k(:)	Surface reflectance K array pointer: (de)allocated after call
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(profile_type)	**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(profile_type)	**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,
              nprof,
              profiles,
              nlevels,
              opts,
              asw,
              coefs,
              init)
```

rttov_alloc_prof is called in the user's program 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)	nprof	Number of profiles per call to RTTOV
Type(profile_type)	Intent(inout)	profiles(nprof)	Profiles structure to be created
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(profile_type)	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,
              nchannels,
              radiance,
              nlayers,
              asw,
              radiance2,
              init)
```

rttov_alloc_rad is called in the user's program 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)	nchannels	Number of channels (i.e. total number of radiances to compute)
Type(radiance_type)	Intent(inout)	radiance	Radiance structure to be created
Integer	Intent(in)	nlayers	Number of profile layers (i.e. nlevels-1)
Integer	Intent(in)	asw	Switch (1=allocate; 0=deallocate)
Type(radiance2_type)	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(radiance_type)	Intent(inout)	radiance	Radiance structure to be initialised
Type(radiance2_type)	Intent(inout), optional	radiance2	Secondary radiance structure to be initialised

9. RTTOV_ALLOC_TRANSMISSION interface

```
call rttov_alloc_transmission (
             err,
             transmission,
             nlayers,
             nchannels,
             asw,
             init)
```

rttov_alloc_transmission is called in the user's program to allocate or de-allocate the memory for the **transmission** structure.

Type	In/Out	Variable	Description
Integer	Intent(out)	err	Return code
Type(transmission_type)	Intent(inout)	transmission	Transmission structure to be created
Integer	Intent(in)	nlayers	Number of profile layers (i.e. nlevels-1)
Integer	Intent(in)	nchannels	Number of channels (i.e. total number of radiances to compute)
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(transmission_type)	Intent(inout)	transmission	Transmission structure to be initialised

11. RTTOV_ALLOC_PCCOMP interface

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

rttov_alloc_pccomp is called in the user's program to allocate or de-allocate the memory for the **pccomp** structure. This is only required for Principal Component calculations.

Type	In/Out	Variable	Description
Integer	Intent(out)	err	Return code
Type(rttov_pccomp)	Intent(inout)	pccomp	pccomp structure to be created
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** 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_TRAJ interface

```
call rttov_alloc_traj (err, nprofiles, nchannels, 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 users 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.

Once all radiances have been computed, the user 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)	nchannels	Maximum number of channels per call to RTTOV.
Type(rttov_options)	Intent(in)	opts	RTTOV options structure
Integer	Intent(in)	nlevels	Number of levels in input profiles.
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 initialised with a single call to **rttov_alloc_traj**.

14. RTTOV_ALLOC_SCATT_PROF interface

```
call rttov_alloc_scatt_prof (nprof, cld_profiles, nlev, use_totalice, asw,
                             init, mmr_snowrain)
```

rttov_alloc_scatt_prof is called in the user's program 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(in)	nprof	Number of profiles per call to RTTOV.
Type(profile_cloud_type)	Intent(inout)	cld_profiles(nprof)	RTTOV-SCATT cloud profile structure to be 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/m2/s; True => kg/kg (default)

15. 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(profile_cloud_type)	Intent(inout)	cld_profiles(nprof)	RTTOV-SCATT cloud profile structure to be initialised.

16. 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_coef)	Intent(in)	coef	RTTOV optical depth coefficients structure (i.e. **coefs%coef**).
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 subroutines

1. RTTOV_ALLOC_OPT_PARAM interface

```
call rttov_alloc_opt_param (
             err,
             opts,
             opt_param,
             nchanprof,
             nlayers,
             nphangle,
             asw)
```

rttov_alloc_opt_param is called in the user's program 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 allocated.
Type(rttov_chanprof)	Intent(in)	chanprof	RTTOV chanprof structure
Integer	Intent(in)	nlayers	Number of profile layers.
Integer	Intent(in)	nphangle	Number of phase angles over which phase functions are to be defined.
Integer	Intent(in)	asw	Switch (1=allocate; 0=deallocate)

2. RTTOV_INIT_OPT_PARAM interface

```
call rttov_alloc_opt_param (
             err,
             opts,
             opt_param)
```

rttov_init_opt_param is called in the user's program 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.

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 allocated.

3. RTTOV_BPR_INIT interface

```
call rttov_bpr_init (
             phangle,
             err)
```

rttov_bpr_init is called in the user's program 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
Real	Intent(in)	phangle(:)	Array of angles over which phase functions are defined.
Integer	Intent(out)	err	Return code

4. RTTOV_BPR_CALC interface

```
call rttov_bpr_calc (
              pha,
              phangle,
              bpr,
              err)
```

rttov_bpr_calc is called in the user's program 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.

Type	In/Out	Variable	Description
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(out)	err	Return code

5. RTTOV_BPR_DEALLOC interface

```
call rttov_bpr_dealloc (
              err)
```

rttov_bpr_dealloc is called in the user's program after *b* parameters have been calculated.

Type	In/Out	Variable	Description
Integer	Intent(out)	err	Return code

Annex F – Emissivity atlas subroutines

1. RTTOV_SETUP_EMIS_ATLAS interface

```
call rttov_setup_emis_atlas (
              err,
              opts,
              imonth,
              coefs,
              path,
              ir_atlas_read_std,
              ir_atlas_single_instrument,
              ir_atlas_ang_corr,
              ir_atlas_ver,
              mw_atlas_ver)
```

This routine initialises the emissivity atlas appropriate to the sensor defined in the **coefs** structure (IR or MW). Climatological emissivities are provided for each month of the year. An optional **path** argument allows you to specify where the atlas data is stored. The code has been designed to allow for multiple versions of each atlas to co-exist. Currently there are two alternative MW atlases, selected by supplying version numbers 100 (the default if no version number is supplied) and 200. The two alternatives are described below. It is not possible to initialise both MW atlases at the same time. There is only one IR atlas at present, so there is no need to specify the version number in this case. For the IR atlas and the TELSEM MW atlas, once initialised each atlas may be called for any IR or MW instrument respectively. There is a new option to initialise the IR atlas for a single instrument only: this results in significantly faster accesses to the atlas, but the atlas may then only be called for this instrument.

As of RTTOV v11.3 the IR atlas files are available in HDF5 format. If you compile RTTOV with the HDF5 library then you **must** use the HDF5 format atlas files. Otherwise you **must** use the NetCDF files. The HDF5 files are significantly smaller in size due to the use of internal HDF5 compression. If the IR atlas standard deviations are not required there is no need to download the standard deviation data files from the web site.

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_coefs)	Intent(in)	coefs	RTTOV instrument coefficients structure.
Character	Intent(in), optional	path	Path of directory containing emissivity atlas data files. Defaults to the current directory.
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. Default is false. Not applicable to MW atlases.
Logical	Intent(in), optional	ir_atlas_single_instrument	If true, initialise IR atlas for a single instrument only (the one specified in coefs). This speeds up access when calling rttov_get_emis. 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	ir_atlas_ver	Specify IR atlas version. Currently there is only one IR atlas version, so this argument should not be used.
Integer	Intent(in), optional	mw_atlas_ver	Specify MW atlas version. Valid values are 100 (the default) or 200 (see above and rttov_get_emis routine for more details).

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. 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:

```
call rttov_get_emis (
              err,
              opts,
              chanprof,
              profiles,
              coefs,
              emissivity,
              emis_std,
              emis_flag)
```

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. The IR atlas can optionally return an estimate of the emissivity errors (standard deviations). In addition, each surface point in the atlas has an associated flag which may be returned. The user may wish to use this flag as a form of quality control (see the IR atlas documentation Borbas *et al.*, 2010). 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.

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(profile_type)	Intent(in)	profiles(:)	Profiles structure.
Type(rttov_coefs)	Intent(in)	coefs	RTTOV instrument coefficient structure.
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.

Returning MW emissivities:

```
call rttov_get_emis (
              err,
              opts,
              chanprof,
              profiles,
              coefs,
              resolution,
              emissivity,
              emis_std,
              emis_cov)
```

TELSEM is loaded by default if no version is specified when calling **rttov_setup_emis_atlas** for a MW instrument It 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 degrees. If the **resolution** parameter 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 atlas, the TELSEM MW atlas can optionally return estimated emissivity errors. It can also optionally return an emissivity covariance matrix: 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(profile_type)	Intent(in)	profiles(:)	Profiles structure.
Type(rttov_coefs)	Intent(in)	coefs	RTTOV instrument coefficient structure.
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 degrees.
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.

Returning MW emissivities, CNRM atlas (version 200):

```
call rttov_get_emis (
              err,
              opts,
              chanprof,
              profiles,
              coefs,
              emissivity,
              pbats_veg)
```

The CNRM atlas has emissivity datasets **only** for specific instruments (AMSU-A, AMSU-B, MHS) and so does not need to carry out interpolation in frequency to instrument channels. It does not provide an estimate of emissivity error. To use this atlas, **rttov_ setup_emis_atlas** must be called with **mw_atlas_version=200**.

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(profile_type)	Intent(in)	profiles(:)	Profiles structure.
Type(rttov_coefs)	Intent(in)	coefs	RTTOV instrument coefficient structure.
Real	Intent(out)	emissivity(nchanprof)	Emissivity values.
Real	Intent(out), optional	pbats_veg	Vegetation type.

3. RTTOV_DEALLOCATE_EMIS_ATLAS interface

```
call rttov_deallocate_emis_atlas (coefs)
```

rttov_deallocate_emis_atlas is called in the user's program to de-allocate the memory for the emissivity atlas.

Type	In/Out	Variable	Description
Type(rttov_coefs)	Intent(in)	coefs	RTTOV instrument coefficient structure.

Annex G – BRDF atlas subroutines

1. RTTOV_SETUP_BRDF_ATLAS interface

```
call rttov_setup_brdf_atlas (
              err,
              opts,
              imonth,
              coefs,
              path,
              brdf_atlas_single_instrument,
              vn_atlas_ver)
```

The atlas is described in Vidot and Borbas (2013). This routine initialises the BRDF atlas. Climatological BRDFs are provided for each month of the year. An optional **path** argument allows the user to specify where the atlas data is stored. The code has been designed to allow for multiple versions of each atlas to co-exist. Currently there is only one BRDF atlas so the version argument should not be used. Once initialised the atlas may be called for any appropriate instrument. There is an option to initialise the BRDF atlas for a single instrument only: this results in significantly faster accesses to the atlas, but the atlas may then only be called for this instrument.

As of RTTOV v11.3 the BRDF atlas files are available in HDF5 format. If you compile RTTOV with the HDF5 library then you **must** use the HDF5 format atlas files. Otherwise you **must** use the NetCDF files. The HDF5 files are significantly smaller in size due to the use of internal HDF5 compression.

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_coefs)	Intent(in)	coefs	RTTOV instrument coefficients structure.
Character	Intent(in), optional	path	Path of directory containing BRDF atlas data files. Defaults to the current directory.
Logical	Intent(in), optional	brdf_atlas_single_instrument	If true, initialise BRDF atlas for a single instrument only (the one specified in coefs). This speeds up access when calling rttov_get_brdf. Default is false.
Integer	Intent(in), optional	vn_atlas_ver	Specify BRDF atlas version. Currently there is only one atlas, so this argument should not be used.

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. In the discussions below, **nchanprof** is the size of the **chanprof(:)** array.

```
call rttov_get_brdf (
              err,
              chanprof,
              profiles,
              coefs,
              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. The user 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.

Type	In/Out	Variable	Description
Integer	Intent(out)	err	Return code
Type(rttov_chanprof)	Intent(in)	chanprof(:)	Chanprof structure.
Type(profile_type)	Intent(in)	profiles(:)	Profiles structure.
Type(rttov_coefs)	Intent(in)	coefs	RTTOV instrument coefficient structure.
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 (coefs)
```

rttov_deallocate_brdf_atlas is called in the user's program to de-allocate the memory for the BRDF atlas.

Type	In/Out	Variable	Description
Type(rttov_coefs)	Intent(in)	coefs	RTTOV instrument coefficient structure.

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).
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,
                   radiancedata, radiancedata2, 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 **radiancedata** (and optionally **radiancedata2**) 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 the user wishes 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(profile_type)	Intent(in)	profiles(nprof)	Profiles structure.
Type(rttov_coefs)	Intent(in)	coefs	RTTOV coefficient structure.
Type(transmission_type)	Intent(inout)	transmission	Output transmittances (0-1).
Type(radiance_type)	Intent(inout)	radiancedata	Output radiances (mW/cm^{-1}/sr/m^2, degK, and BRF/unitless).
Type(radiance2_type)	Intent(inout), optional	radiancedata2	Secondary output radiances (mW/cm^{-1}/sr/m^2).
Logical	Intent(in), optional	calcemis(nchanprof)	.true. if RTTOV should calculate surface emissivity using ISEM/FASTEM or .false. if user is supplying an emissivity value (e.g. from the 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, radiancedata, radiancedata_k,
              calcemis, emissivity, emissivity_k, calcrefl, reflectance,
              reflectance_k, aer_opt_param, cld_opt_param, 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.

Note that in RTTOV v9 the **radiancedata_k** argument was optional, but in RTTOV v11 this argument is mandatory. To obtain the same behaviour as the default in RTTOV v9 you should set either **radiancedata_k%total(:)** or **radiancedata_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 **radiancedata_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(profile_type)	Intent(in)	profiles(nprof)	Profiles structure.
Type(profile_type)	Intent(inout)	profiles_k(nchanprof)	Jacobian on profile variables.
Type(rttov_coefs)	Intent(in)	coefs	RTTOV coefficient structure.
Type(transmission_type)	Intent(inout)	transmission	Output transmittances (0-1).
Type(transmission_type)	Intent(inout)	transmission_k	Jacobian of transmittances.
Type(radiance_type)	Intent(inout)	radiancedata	Direct model output radiances ($mW/cm^{-1}/sr/m^2$, degK, BRF/unitless).
Type(radiance_type)	Intent(inout)	radiancedata_k	Input radiance perturbations (see text).
Logical	Intent(in), optional	calcemis(nchanprof)	.true. if RTTOV should calculate surface emissivity using ISEM/FASTEM or .false. if user is supplying an emissivity value (e.g. from the atlas).
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 (e.g. from the atlas).
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(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.
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(profile_type)	Intent(inout). optional	profiles_k_pc(npcscores* nprofiles)	Jacobian on principal component scores. Only required if **opts%rt_ir%pc%addradrec** is .false.
Type(profile_type)	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, radiancedata, radiancedata_tl,
               calcemis, emissivity, emissivity_tl, calcrefl, reflectance,
               reflectance_tl, aer_opt_param, cld_opt_param, 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(profile_type)	Intent(in)	profiles(nprof)	Profiles structure.
Type(profile_type)	Intent(inout)	profiles_tl(nprof)	Input profile variable increments.
Type(rttov_coefs)	Intent(in)	coefs	RTTOV coefficient structure.
Type(transmission_type)	Intent(inout)	transmission	Output transmittances (0-1).
Type(transmission_type)	Intent(inout)	transmission_tl	TL of transmittances.
Type(radiance_type)	Intent(inout)	radiancedata	Direct model output radiances (mW/cm^{-1}/sr/m^2, degK, BRF/unitless).
Type(radiance_type)	Intent(inout)	radiancedata_tl	TL of radiances.
Logical	Intent(in), optional	calcemis(nchanprof)	.true. if RTTOV should calculate surface emissivity using ISEM/FASTEM or .false. if user is supplying an emissivity value (e.g. from the 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(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.
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, radiancedata, radiancedata_ad,
               calcemis, emissivity, emissivity_ad, calcrefl, reflectance,
               reflectance_ad, aer_opt_param, cld_opt_param, 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 **radiancedata_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(profile_type)	Intent(in)	profiles(nprof)	Profiles structure.
Type(profile_type)	Intent(inout)	profiles_ad(nprof)	AD on profile variables.
Type(rttov_coefs)	Intent(in)	coefs	RTTOV coefficient structure.
Type(transmission_type)	Intent(inout)	transmission	Output transmittances (0-1).
Type(transmission_type)	Intent(inout)	transmission_ad	AD of transmittances.
Type(radiance_type)	Intent(inout)	radiancedata	Direct model output radiances ($mW/cm^{-1}/sr/m^2$, degK, BRF/unitless).
Type(radiance_type)	Intent(inout)	radiancedata_ad	Input radiance perturbations (see text).
Logical	Intent(in), optional	calcemis(nchanprof)	.true. if RTTOV should calculate surface emissivity using ISEM/FASTEM or .false. if user is supplying an emissivity value (e.g. from the 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(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.
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, calcemiss,
                  emissivity, radiance, cfrac)
```

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 v11 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 to the user via the **emis_out** member.

The **frequencies(:)** and **chanprof(:)** arrays should be populated by calling **rttov_scatt_setupindex** (Annex D) before calling **rttov_scatt**.

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(profile_type)	Intent(in)	profiles(nprof)	Profiles structure.
Type(profile_cloud_type)	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)	calcemiss(nchanprof)	.true. if RTTOV should calculate surface emissivity using FASTEM 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(radiance_type)	Intent(inout)	radiance	Output radiances (mW/cm^{-1}/sr/m^2 & degK).
Real	Intent(out), optional	cfrac	Cloud fraction actually used (diagnostic).

```
call rttov_scatt_tl (errorstatus, opts_scatt, nlevels, chanprof, frequencies,
                     profiles, cld_profiles, coef_rttov, coef_scatt, calcemiss,
                     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(:)**.

In RTTOV v11 it is possible to control a limited number of the internal RTTOV options via the **opts_scatt** parameter (see Annex O).

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 to the user via the **emis_out** member. The emissivity TL values are treated as for **rttov_tl**.

See section 7.9 for more information about the RTTOV TL model.

The **frequencies(:)** and **chanprof(:)** arrays should be populated by calling **rttov_scatt_setupindex** (Annex D) before calling **rttov_scatt**.

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(profile_type)	Intent(in)	profiles(nprof)	Profiles structure.
Type(profile_cloud_type)	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)	calcemiss(nchanprof)	.true. if RTTOV should calculate surface emissivity using FASTEM 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(profile_type)	Intent(in)	profiles_tl(nprof)	Input profile variable increments.
Type(profile_cloud_type)	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(radiance_type)	Intent(inout)	radiance	Output radiances (mW/cm^{-1}/sr/m^2 & degK).
Type(radiance_type)	Intent(inout)	radiance_tl	TL of radiances.

```
call rttov_scatt_ad (errorstatus, opts_scatt, nlevels, chanprof, frequencies,
                     profiles, cld_profiles, coef_rttov, coef_scatt, calcemiss,
                     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**.

In RTTOV v11 it is possible to control a limited number of the internal RTTOV options via the **opts_scatt** parameter (see Annex O).

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 to the user via the **emis_out** member. 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.

The **frequencies(:)** and **chanprof(:)** arrays should be populated by calling **rttov_scatt_setupindex** (Annex D) before calling **rttov_scatt**.

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(profile_type)	Intent(in)	profiles(nprof)	Profiles structure.
Type(profile_cloud_type)	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)	calcemiss(nchanprof)	.true. if RTTOV should calculate surface emissivity using FASTEM 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(profile_type)	Intent(inout)	profiles_ad(nprof) OR profiles_ad(nchanprof)	AD or Jacobian on profile variables.
Type(profile_cloud_type)	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(radiance_type)	Intent(inout)	radiance	Output radiances (mW/cm^{-1}/sr/m^2 & degK).
Type(radiance_type)	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. The user 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(profile_type)	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_options (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_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.

5. 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(profile_type)	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.

6. CREATE_AER_CLIM_PROF.EXE

This executable may be used to generate profiles on the layers defined by the user's pressure levels for the first 10 RTTOV aerosol types (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 the user 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

7. RTTOV_AER_CLIM_PROF

This subroutine is called by `create_aer_clim_prof.exe` to generate the aerosol profiles. Users may wish to call this subroutine directly in their 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 (ppmv).
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 aerosol types (cm^{-3})

8. 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 v10 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.

9. 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.

It is expected that users would wish to modify the source code to suit their 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).

Annex O – RTTOV v11 derived types

Only derived types which can be used at the user's level are presented, see **rttov_types.F90** for the full description of all derived types used.

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. Most options have the same name as their RTTOV v10 equivalents.

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). Similar to RTTOV v10 *opts%verbose_checkinput_warnings*
General radiative transfer options: ***opts % rt_all***		
Logical	opts%rt_all%**switchrad**	Determines input perturbation for AD/K routines: if true *radiancedata_ad/k % bt* is used for channels with wavelengths > 3µm, otherwise *radiancedata_ad/k % total* is used (default = false)
Logical	opts%rt_all%**addrefrac**	If true RTTOV calculations account for atmospheric refraction (default = false)
Logical	opts%rt_all%**use_q2m**	If true activate use of surface humidity (default = true)
Logical	opts%rt_all%**do_lambertian**	If true activate treatment of surface as Lambertian instead of specular reflector for downwelling emitted radiance (default = false) (Since v11.3 applies to IR and MW instruments).
MW-only radiative transfer options: ***opts % rt_mw***		
Integer	opts%rt_mw%**fastem_version**	Select the version of the FASTEM model to use. Valid versions are 1-6; other values result in the version specified in the coefficient file being used (default = 5).
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. When using RTTOV-SCATT this option should not be set.
Logical	opts%rt_mw%**do_lambertian**	If true activate treatment of surface as Lambertian instead of specular reflector for downwelling emitted radiance (default = false) (Applies to MW instruments only; retained in v11.3 for backward compatibility; recommended to use *opts%rt_all%do_lambertian*).
Logical	opts%rt_mw%**supply_foam_fraction**	If true use the foam fraction value specified in profiles(:)%skin%foam_fraction in FASTEM.

Visible/IR-only radiative transfer options: ***opts % rt_ir***		
Logical	opts%rt_ir%**addsolar**	If true enable solar calculations for solar-affected channels (default = false)
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*. New in v11.
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). New in v11.
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 number density profiles for pre-defined particle types (default = false). New in v11.
Real	opts%rt_ir%**cldstr_threshold**	Threshold for stream 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%**ozone_data**	If true user is supplying ozone profiles (default = false)
Logical	opts%rt_ir%**co2_data**	If true user is supplying co2 profiles (default = false)
Logical	opts%rt_ir%**n2o_data**	If true user is supplying n2o 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 ch4 profiles (default = false)
Logical	opts%rt_ir%**do_lambertian**	If true activate treatment of surface as Lambertian instead of specular reflector for downwelling emitted radiance (default = false) (Applies to IR instruments only; introduced for consistency with MW option; recommended to use *opts%rt_all%do_lambertian*).

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).
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 v11.3. New in v11.
Integer	opts%rt_ir%pc%**ipcreg**	The index of the required set of PC predictors: 1-4 for IASI/IASI-NG (see table 25 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). New in v11.
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). New in v11.
Logical	opts%interpolation%**lgradp**	Allow TL/AD of user pressure levels if *addinterp* is true (default = false)
Logical	opts%interpolation%**spacetop**	If true treat user's model top as space boundary (default = true)

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 to the user.

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)
Integer	opts_scatt%**interp_mode**	Set the interpolation mode (see Table 9 in section 7.3). Valid values are 1-5 (default = 1).
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).
Integer	opts_scatt%**fastem_version**	Select the version of the FASTEM model to use. Valid versions are 1-6; other values result in the version specified in the coefficient file being used (default = 5).
Logical	opts_scatt %**supply_foam_fraction**	If true use the foam fraction value specified in profiles(:)%skin%foam_fraction in FASTEM.
Logical	opts_scatt%**use_q2m**	If true, activate use of surface humidity (default = true)
Logical	opts_scatt%**lusercfrac**	If true, the user should supply supply the effective cloud fraction in the *profile_cloud_type* (see below). If false, this is calculated internally in RTTOV-SCATT. (default = false).

Profile structure

The *profile_type* structure is composed of the atmospheric part and two other structures for 2 meters air and skin surface. If the user is not able to provide ozone, CO2, 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 *skin_type*		
Integer	surftype	0=land, 1=sea, 2=sea-ice
Integer	watertype	0=fresh water, 1=ocean water – used for solar BRDF only
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	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.
Surface 2m – type *s2m_type*		
Real	t	Temperature (K)
Real	q	Water vapour (ppmv) – *only used if opts%rt_all%use_q2m is true.*
Real	o	Ozone (ppmv) – *not currently used.*
Real	p	Surface pressure (hPa)
Real	u	U 10m wind component (m/s)
Real	v	V 10m wind component (m/s)
Real	wfetc	Wind fetch (m) (typically 100000) – *only used by sea surface*

		BRDF model (for visible/near-IR).
Atmospheric profile – type *profile_type*		
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.
Integer	nlayers	Number of atmospheric layers (i.e. nlevels-1).
Integer	gas_units	Units for gas abundances: 2 => ppmv over moist air 1=> kg/kg over moist air 0=> "v11.2 compatibility mode" (default) -1=> ppmv over dry air
Real	p(nlevels)	Pressure (hPa)
Real	t(nlevels)	Temperature (K)
Real	q(nlevels)	Water vapour (ppmv)
Real	o3(nlevels)	Ozone (ppmv)
Real	co2(nlevels)	CO2 (ppmv)
Real	n2o(nlevels)	N2O (ppmv)
Real	co(nlevels)	CO (ppmv)
Real	ch4(nlevels)	CH4 (ppmv)
Real	clw(nlevels)	Cloud liquid water (kg/kg) – *MW only, treats cloud as absorbing medium only in "clear-sky"; do not use with RTTOV-SCATT.*
Real	aerosols(naertyp,nlayers) *(naertyp is the number of aerosol types, currently 13)*	Aerosols (cm^{-3}) – *IR only.* Note: the number of aerosol types is defined in the IR scattering coefficient file. This is not used if opts%rt_ir%user_aer_opt_param is true.
Real	cloud(ncldtyp,nlayers) *(ncldtyp is currently 6 => 5 water cloud types plus ice cloud)*	Cloud water/ice (g/m^3) – *IR only.* Note: the number of water cloud types is defined in the IR scattering coefficient file. This is not used if opts%rt_ir%user_cld_opt_param is true.
Real	cfrac(nlayers)	Cloud fractional cover (0-1) – *IR only.*
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.* This is not used if opts%rt_ir%user_cld_opt_param is true or if the Baran ice particle scheme is used.
Integer	idg	Scheme for ice water content to effective diameter, Dg 1=Ou and Liou; 2=Wyser et al.; 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 particle scheme is used (ish=3 or 4) or in layers where profiles(:)%icede(:) is non-zero.
Integer	ish	Choose the shape of ice crystals, 1=Hexagonal ice crystals; 2=Ice aggregates; 3=Baran scheme (v11.1), 4=Baran scheme (v11.2/3). This is not used if opts%rt_ir%user_cld_opt_param is true.
Real	zenangle	Local satellite zenith angle (degrees), maximum valid value depends on coefficient file (see Table 4).
Real	azangle	Local satellite azimuth angle (0-360 deg; measured clockwise, east=90, see Figure 4)
Real	sunzenangle	Local solar zenith angle (degrees), solar radiation only included up to 84 degrees.
Real	sunazangle	Local solar azimuth angle (0-360 deg; measured clockwise, east=90, see Figure 4)
Real	elevation	Elevation (km)
Real	latitude	Latitude (deg) -90 to +90
Real	longitude	Longitude (deg) 0-360. *Used only by emissivity and BRDF atlases.*
Real	snow_frac	Surface snow coverage fraction (0-1). *Used only by IR emissivity*

		atlas.
Real	soil_moisture	Soil moisture (m^3/m^3) – *not currently used.*
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 *profile_cloud_type* 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
Logical	use_totalice	logical flag to switch between using separate ice and snow, or total ice hydrometeor types.
Logical	mmr_snowrain	logical flag to select units for snow and rain hydrometeors: True => kg/kg (default); False => kg/m2/s
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(nchannels,nlayers)	Absorption coefficients (km^{-1})
Real	sca(nchannels,nlayers)	Scattering coefficients (km^{-1})
Real	bpr(nchannels,nlayers)	"*b*" parameters (no units): represents the fraction of backscattered radiation at each layer. A subroutine is provided to calculate these from the phase functions.
Real	phangle(:)	Angles over which the phase function is defined (degrees).
Real	pha(nchannels,nlayers,nphangle)	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 for solar-affected channels.

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 *radiance_type* 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. Both radiances and brightness temperatures are computed so the user can decide which quantity he wants to use in his program.

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 given black cloud 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 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 or IR aerosol simulations.
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

Secondary radiance structure

The *radiance2_type* 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. They do not contain any solar contributions. 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.

Similarly to **rad%overcast**, for the layer j 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 temperatue.

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 atmsosphere for layers down to the level bounding the bottom of each layer.
Real	down(nlayer,nchanprof)	Summed downwelling atmospheric emission term at bottom of atmsosphere 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 *transmission_type* 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, 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.

PCCOMP structure

The *rttov_pccomp* structure contains the results of the Principal Component calculations. The **pcscores** member has size equal to the number of principal components being used. The output radiance and BT arrays are only required if **opts%rt_ir%pc%addradrec** is true. These are sized according to the number of channels for which reconstructed radiances are required.

Type	Variable	Description
Real	pcscores(npcscores)	Computed PC scores.
Real	total_pccomp(nchannels_rec)	Clear-sky/cloudy radiances constructed using principal components.
Real	bt_pccomp(nchannels_rec)	BTs equivalent to reconstructed radiances.

Annex P – Contents of rttov_const.F90

```
! Description:
!> @file
!!   Defines various constants used by RTTOV
!
!> @brief
!!   Defines various constants used by RTTOV
!
! Copyright:
!    This software was developed within the context of
!    the EUMETSAT Satellite Application Facility on
!    Numerical Weather Prediction (NWP SAF), under the
!    Cooperation Agreement dated 25 November 1998, 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, KNMI and MeteoFrance.
!
!    Copyright 2015, EUMETSAT, All Rights Reserved.
!
MODULE rttov_const

  USE parkind1, ONLY : jpim, jprb
  IMPLICIT NONE

  ! General
  ! -------
  ! Version number of the current code
  INTEGER(KIND=jpim), PARAMETER :: version = 11
  INTEGER(KIND=jpim), PARAMETER :: release = 3
  INTEGER(KIND=jpim), PARAMETER :: minor_version = 0

  ! Min/max version numbers of compatible coefficient files:
  !   coef files with "id_comp_lvl" outside range will be rejected
  INTEGER(KIND=jpim), PARAMETER :: version_compatible_min = 10
  INTEGER(KIND=jpim), PARAMETER :: version_compatible_max = 11

  CHARACTER(LEN=16),  PARAMETER :: rttov_magic_string = '%RTTOV_COEFF    '
  REAL(KIND=jprb),    PARAMETER :: rttov_magic_number = 1.2345E+12_jprb

  INTEGER(KIND=jpim), PARAMETER :: default_err_unit = 0  ! standard error unit number
                             ! standard error unit number is 7 for HPUX

  ! Precision and numerical constants
  ! ---------------------------------
  ! Try to ensure this is large enough to avoid overflows in reciprocals but small enough to not
affect the calculations.
  ! these parameters are defined at bottom of module (because they use
  REAL(jprb), PARAMETER :: max_exp_exponent = 50._jprb ! approx 1e22 which should be sufficiently
big for most purposes
  REAL(jprb), PARAMETER :: min_exponent     = 1e-16_jprb ! approx log_10(1+2^-52) - anything raised
to this power or smaller
                                                         ! should be approx equal to 1
! small_val is defined in rttov_transmit to avoid compiler incompatibility
!   ! small_val is used in rttov_transmit to ensure small values do not result in underflows. In
subsequent calculations
!   ! these values are multiplied together hence the exponent of 1/3.
!   REAL(jprb)            :: small_val = (tiny(min_exponent)) ** (0.333333_jprb) ! XLF doesn't like
1/3

  ! Physical constants
  ! ------------------
  ! Molecular weights  (g/mole) are calculated by adding NIST Standard Atomic Weights
  ! Molecular weight of dry air refers to US standard atmosphere 1976
  ! NIST  Standard Atomic Weight are:
  ! H    1.00794    (7)
  ! C   12.0107     (8)
  ! N   14.0067     (2)
  ! O   15.9994     (3)
  ! http://webbook.nist.gov/chemistry/form-ser.html
  REAL(KIND=jprb), PARAMETER :: mair   = 28.9644_jprb
  REAL(KIND=jprb), PARAMETER :: mh2o   = 18.01528_jprb
  REAL(KIND=jprb), PARAMETER :: mo3    = 47.9982_jprb
```

```
REAL(KIND=jprb), PARAMETER :: mco2   = 44.0095_jprb
REAL(KIND=jprb), PARAMETER :: mch4   = 16.04246_jprb
REAL(KIND=jprb), PARAMETER :: mn2o   = 44.0128_jprb
REAL(KIND=jprb), PARAMETER :: mco    = 28.0101_jprb
REAL(KIND=jprb), PARAMETER :: mso2   = 64.064_jprb
REAL(KIND=jprb), PARAMETER :: mo2    = 31.9988_jprb
REAL(KIND=jprb), PARAMETER :: mno    = 30.0061_jprb
REAL(KIND=jprb), PARAMETER :: mno2   = 46.0055_jprb
REAL(KIND=jprb), PARAMETER :: mhno3  = 63.0128_jprb
REAL(KIND=jprb), PARAMETER :: mocs   = 60.075_jprb
REAL(KIND=jprb), PARAMETER :: mn2    = 28.0134_jprb
REAL(KIND=jprb), PARAMETER :: mccl4  = 153.823_jprb
REAL(KIND=jprb), PARAMETER :: mcfc11 = 137.368_jprb ! CFCl3
REAL(KIND=jprb), PARAMETER :: mcfc12 = 120.914_jprb ! CF2Cl2
REAL(KIND=jprb), PARAMETER :: mcfc14 = 88.0043_jprb ! CF4


! NB Included in v11.3 for compatibility where other software uses these: to be removed in v12.
!    Simple units conversion from  mixing ratio to ppmv
REAL(KIND=jprb), PARAMETER :: q_mixratio_to_ppmv  = 1.60771704e+6_jprb
REAL(KIND=jprb), PARAMETER :: o3_mixratio_to_ppmv = 6.03504e+5_jprb
REAL(KIND=jprb), PARAMETER :: co2_mixratio_to_ppmv= 6.58114e+5_jprb
REAL(KIND=jprb), PARAMETER :: co_mixratio_to_ppmv = 1.0340699e+6_jprb
REAL(KIND=jprb), PARAMETER :: n2o_mixratio_to_ppmv= 6.58090e+5_jprb
REAL(KIND=jprb), PARAMETER :: ch4_mixratio_to_ppmv= 1.80548e+6_jprb


! Avogadro constant from NIST (mol-1)
REAL(KIND=jprb), PARAMETER :: na = 6.02214129E23_jprb

! Gravity from NIST 9.80665 ms-1 (exact)
REAL(KIND=jprb), PARAMETER :: gravity = 9.80665_jprb

! Fundamental constants taken from http://physics.nist.gov/cuu/index.html
! Barry N. Taylor (Fundamental Constants Data Center of NIST) and Peter J.
! Mohr (Atomic Physics Division of NIST). Values as of 01/12/2010.
! c1 = 2hc**2; c2 = hc/k; units are consistent with those used within RTTOV.
! NB Planck fn constant values (c1, c2) in core code are still taken from coef files.
!    Newer values are available from NIST: we will update in a future version.
REAL(KIND=jprb), PARAMETER :: c1 = .00001191042722_jprb   ! * mW/(m2 sr cm-4)
REAL(KIND=jprb), PARAMETER :: c2 = 1.4387752_jprb         ! cm K
REAL(KIND=jprb), PARAMETER :: speedl = 29979245800.0_jprb ! Speed of light cm s-1

!
! Kaye & Laby latest library edition is 16e 1995, and gives
! * standard value  g = 9.80665 ms-1 exactly (p.191)
! * earth mean radius r= 6371.00 km (p191)
!    [defined as [(r_equator)^2 (r_pole)]^1/3]
REAL(KIND=jprb), PARAMETER :: pi      = 3.1415926535_jprb
REAL(KIND=jprb), PARAMETER :: deg2rad = pi/180.0_jprb
REAL(KIND=jprb), PARAMETER :: earthradius = 6371.00_jprb
REAL(KIND=jprb), PARAMETER :: flatt       = 3.3528107E-3_jprb
REAL(KIND=jprb), PARAMETER :: omega       = 7292115E-11_jprb
REAL(KIND=jprb), PARAMETER :: eqrad       = 6378.137_jprb
REAL(KIND=jprb), PARAMETER :: grave       = 9.7803267715_jprb
REAL(KIND=jprb), PARAMETER :: z4pi_r      = 0.0795774715_jprb
REAL(KIND=jprb), PARAMETER :: pi_r        = 0.3183098862_jprb
REAL(KIND=jprb), PARAMETER :: sec_theta_eff = 1.743446796_jprb   ! theta_eff = 55 degrees

! The Cosmic Microwave Background Spectrum from the Full COBE FIRAS Data Set
! Fixsen D.J. et al Astrophysical Journal v.473, p.576 December 1996
! CMBR = 2.728 +- 0.004K
REAL(KIND=jprb), PARAMETER :: tcosmic     = 2.728_jprb
!  REAL(KIND=jprb), PARAMETER :: tcosmic     = 0.1_jprb !used for ECMWF tests

! Universal gas constant R = 8.314510 J/mol/K
REAL(KIND=jprb), PARAMETER :: rgp = 8.314510_jprb
REAL(KIND=jprb), PARAMETER :: rgc = 8.314472_jprb

! mean molar mass of dry air rm = 0.0289644 kg.mol^-1
REAL(KIND=jprb), PARAMETER :: rm = 0.0289644_jprb

! zero temperature(K)
REAL(KIND=jprb), PARAMETER :: t0 = 273.15_jprb
```

```
! standard pressure
REAL(KIND=jprb), PARAMETER :: p0 = 1013.25_jprb


! Satellite and instrument information
! -------------------------------------

! Platform ID codes
INTEGER(KIND=jpim), PARAMETER :: nplatforms = 45
! NB ID 22 was changed from "insat_3d" - use ID 40 for INSAT3
!    IDs 41 and 45 are for ground-based and airborne sensors (experimental)
INTEGER(KIND=jpim), PARAMETER :: &
     & platform_id_noaa      = 1,  platform_id_dmsp      = 2,  platform_id_meteosat  = 3,  &
     & platform_id_goes      = 4,  platform_id_gms       = 5,  platform_id_fy2       = 6,  &
     & platform_id_trmm      = 7,  platform_id_ers       = 8,  platform_id_eos       = 9,  &
     & platform_id_metop     = 10, platform_id_envisat   = 11, platform_id_msg       = 12, &
     & platform_id_fy1       = 13, platform_id_adeos     = 14, platform_id_mtsat     = 15, &
     & platform_id_coriolis  = 16, platform_id_jpss      = 17, platform_id_gifts     = 18, &
     & platform_id_sentinel3 = 19, platform_id_meghatr   = 20, platform_id_kalpana   = 21, &
     & platform_id_meteor    = 22, platform_id_fy3       = 23, platform_id_coms      = 24, &
     & platform_id_meteorm   = 25, platform_id_gosat     = 26, platform_id_calipso   = 27, &
     & platform_id_dummy     = 28, platform_id_gcomw     = 29, platform_id_nimbus    = 30, &
     & platform_id_himawari  = 31, platform_id_mtg       = 32, platform_id_saral     = 33, &
     & platform_id_metopsg   = 34, platform_id_landsat   = 35, platform_id_jason     = 36, &
     & platform_id_gpm       = 37, platform_id_insat1    = 38, platform_id_insat2    = 39, &
     & platform_id_insat3    = 40, platform_id_ground    = 41, platform_id_dscovr    = 42, &
     & platform_id_clarreo   = 43, platform_id_ticfire   = 44, platform_id_aircraft  = 45

! Platform names
CHARACTER(LEN=9), PARAMETER :: platform_name(nplatforms) = &
     & (/ 'noaa     ', 'dmsp     ', 'meteosat ', 'goes     ', 'gms      ', &
        & 'fy2      ', 'trmm     ', 'ers      ', 'eos      ', 'metop    ', &
        & 'envisat  ', 'msg      ', 'fy1      ', 'adeos    ', 'mtsat    ', &
        & 'coriolis ', 'jpss     ', 'gifts    ', 'sentinel3', 'meghatr  ', &
        & 'kalpana  ', 'meteor   ', 'fy3      ', 'coms     ', 'meteor-m ', &
        & 'gosat    ', 'calipso  ', 'dummy    ', 'gcom-w   ', 'nimbus   ', &
        & 'himawari ', 'mtg      ', 'saral    ', 'metopsg  ', 'landsat  ', &
        & 'jason    ', 'gpm      ', 'insat1   ', 'insat2   ', 'insat3   ', &
        & 'ground   ', 'dscovr   ', 'clarreo  ', 'ticfire  ', 'aircraft ' /)

! Instrument ID codes
INTEGER(KIND=jpim), PARAMETER :: &
     & inst_id_hirs   =  0, inst_id_msu      =  1, inst_id_ssu    =  2, inst_id_amsua   =  3, &
     & inst_id_amsub  =  4, inst_id_avhrr    =  5, inst_id_ssmi   =  6, inst_id_vtpr1   =  7, &
     & inst_id_vtpr2  =  8, inst_id_tmi      =  9, inst_id_ssmis  = 10, inst_id_airs    = 11, &
     & inst_id_hsb    = 12, inst_id_modis    = 13, inst_id_atsr   = 14, inst_id_mhs     = 15, &
     & inst_id_iasi   = 16, inst_id_amsre    = 17, inst_id_gmsim  = 18, inst_id_atms    = 19, &
     & inst_id_mviri  = 20, inst_id_seviri   = 21, inst_id_goesim = 22, inst_id_goessd  = 23, &
     & inst_id_mtsatim= 24, inst_id_vissr    = 25, inst_id_mvisr  = 26, inst_id_cris    = 27, &
     & inst_id_cmis   = 28, inst_id_viirs    = 29, inst_id_windsat= 30, inst_id_gifts   = 31, &
     & inst_id_ssmt1  = 32, inst_id_ssmt2    = 33, inst_id_saphir = 34, inst_id_madras  = 35, &
     & inst_id_ssmisz = 36, inst_id_vhrr     = 37, inst_id_insatim= 38, inst_id_insatsd = 39, &
     & inst_id_mwts   = 40, inst_id_mwhs     = 41, inst_id_iras   = 42, inst_id_mwri    = 43, &
     & inst_id_abi    = 44, inst_id_mi       = 45, inst_id_msumr  = 46, inst_id_tansofts= 47, &
     & inst_id_iir    = 48, inst_id_mwr      = 49, inst_id_dummyir= 50, inst_id_dummymw = 51, &
     & inst_id_dummyhi= 52, inst_id_dummypo  = 53, inst_id_scams  = 54, inst_id_smmr    = 55, &
     & inst_id_ahi    = 56, inst_id_irs      = 57, inst_id_altika = 58, inst_id_iasing  = 59, &
     & inst_id_tm     = 60, inst_id_fci      = 61, inst_id_amsr1  = 62, inst_id_amsr2   = 63, &
     & inst_id_vissr2 = 64, inst_id_slstr    = 65, inst_id_tirs   = 66, inst_id_amr     = 67, &
     & inst_id_oli    = 68, inst_id_iris     = 69, inst_id_ici    = 70, inst_id_gmi     = 71, &
     & inst_id_mwts2  = 72, inst_id_mwhs2    = 73, inst_id_aster  = 74, inst_id_hatpro  = 75, &
     & inst_id_mtvzagy= 76, inst_id_metimage = 77, inst_id_mws    = 78, inst_id_mwi     = 79, &
     & inst_id_epic   = 80, inst_id_mrir     = 81, inst_id_si     = 82, inst_id_mrfirs  = 83, &
     & inst_id_mbfiri = 84, inst_id_lhr      = 85, inst_id_ismar  = 86, inst_id_mersi1  = 87, &
     & inst_id_mersi2 = 88

INTEGER(KIND=jpim), PARAMETER :: ninst = 89
! List of instruments  !!!! HIRS is number 0
CHARACTER(LEN=8), PARAMETER :: inst_name(0:ninst-1) =       &
      & (/ 'hirs    ', 'msu     ', 'ssu     ', 'amsua   ', 'amsub   ',  &
         & 'avhrr   ', 'ssmi    ', 'vtpr1   ', 'vtpr2   ', 'tmi     ',  &
         & 'ssmis   ', 'airs    ', 'hsb     ', 'modis   ', 'atsr    ',  &
         & 'mhs     ', 'iasi    ', 'amsre   ', 'imager  ', 'atms    ',  &
         & 'mviri   ', 'seviri  ', 'imager  ', 'sounder ', 'imager  ',  &
```

```
          & 'vissr   ', 'mvisr   ', 'cris    ', 'cmis    ', 'viirs   ',  &
          & 'windsat ', 'gifts   ', 'ssmt1   ', 'ssmt2   ', 'saphir  ',  &
          & 'madras  ', 'ssmisz  ', 'vhrr    ', 'imager  ', 'sounder ',  &
          & 'mwts    ', 'mwhs    ', 'iras    ', 'mwri    ', 'abi     ',  &
          & 'mi      ', 'msumr   ', 'tansofts', 'iir     ', 'mwr     ',  &
          & 'dummyir ', 'dummymw ', 'dummyhi ', 'dummypo ', 'scams   ',  &
          & 'smmr    ', 'ahi     ', 'irs     ', 'altika  ', 'iasing  ',  &
          & 'tm      ', 'fci     ', 'amsr    ', 'amsr2   ', 'vissr   ',  &
          & 'slstr   ', 'tirs    ', 'amr     ', 'oli     ', 'iris    ',  &
          & 'ici     ', 'gmi     ', 'mwts2   ', 'mwhs2   ', 'aster   ',  &
          & 'hatpro  ', 'mtvzagy ', 'metimage', 'mws     ', 'mwi     ',  &
          & 'epic    ', 'mrir    ', 'si      ', 'mrfirs  ', 'mbfiri  ',  &
          & 'lhr     ', 'ismar   ', 'mersi1  ', 'mersi2  ' /)

! Sensor type ID codes
INTEGER(KIND=jpim), PARAMETER :: nsensors = 4
INTEGER(KIND=jpim), PARAMETER :: &
     & sensor_id_ir     = 1, &
     & sensor_id_mw     = 2, &
     & sensor_id_hi     = 3, &
     & sensor_id_po     = 4

! Sensor type names
CHARACTER(LEN=2), PARAMETER :: sensor_name(nsensors) = &
     & (/ 'ir', 'mw', 'hi', 'po' /)

! Sensor types corresponding to entries in the inst_name array
INTEGER(KIND=jpim), PARAMETER :: sensor_id(0:ninst-1) = (/ &
  sensor_id_ir, sensor_id_mw, sensor_id_ir, sensor_id_mw, sensor_id_mw,  &
  sensor_id_ir, sensor_id_mw, sensor_id_ir, sensor_id_ir, sensor_id_mw,  &
  sensor_id_mw, sensor_id_hi, sensor_id_mw, sensor_id_ir, sensor_id_ir,  &
  sensor_id_mw, sensor_id_hi, sensor_id_mw, sensor_id_ir, sensor_id_mw,  &
  sensor_id_ir, sensor_id_ir, sensor_id_ir, sensor_id_ir, sensor_id_ir,  &
  sensor_id_ir, sensor_id_ir, sensor_id_hi, sensor_id_mw, sensor_id_ir,  &
  sensor_id_po, sensor_id_hi, sensor_id_mw, sensor_id_mw, sensor_id_mw,  &
  sensor_id_mw, sensor_id_mw, sensor_id_ir, sensor_id_ir, sensor_id_ir,  &
  sensor_id_mw, sensor_id_mw, sensor_id_ir, sensor_id_mw, sensor_id_ir,  &
  sensor_id_ir, sensor_id_ir, sensor_id_hi, sensor_id_ir, sensor_id_mw,  &
  sensor_id_ir, sensor_id_mw, sensor_id_hi, sensor_id_po, sensor_id_mw,  &
  sensor_id_mw, sensor_id_ir, sensor_id_hi, sensor_id_mw, sensor_id_hi,  &
  sensor_id_ir, sensor_id_ir, sensor_id_mw, sensor_id_mw, sensor_id_ir,  &
  sensor_id_ir, sensor_id_ir, sensor_id_mw, sensor_id_ir, sensor_id_hi,  &
  sensor_id_mw, sensor_id_mw, sensor_id_mw, sensor_id_mw, sensor_id_ir,  &
  sensor_id_mw, sensor_id_mw, sensor_id_ir, sensor_id_mw, sensor_id_mw,  &
  sensor_id_ir, sensor_id_ir, sensor_id_hi, sensor_id_hi, sensor_id_ir,  &
  sensor_id_hi, sensor_id_mw, sensor_id_ir, sensor_id_ir /)


! Coefficient file Section names
! ------------------------------
INTEGER(KIND=jpim), PARAMETER :: nsections = 43
INTEGER(KIND=jpim), PARAMETER :: lensection = 34
CHARACTER(LEN=lensection), PARAMETER :: section_types(nsections) = &
  & (/ 'IDENTIFICATION                    ', 'LINE-BY-LINE                      ', &
     & 'FAST_MODEL_VARIABLES              ', 'FILTER_FUNCTIONS                  ', &
     & 'FUNDAMENTAL_CONSTANTS             ', 'SSIREM                            ', &
     & 'FASTEM                            ', 'REFERENCE_PROFILE                 ', &
     & 'PROFILE_LIMITS                    ', 'FAST_COEFFICIENTS                 ', &
     & 'COEF_SUB_FILES                    ', 'GAZ_UNITS                         ', &
     & 'DIMENSIONS                        ', 'FREQUENCIES                       ', &
     & 'HYDROMETEOR                       ', 'CONVERSIONS                       ', &
     & 'EXTINCTION                        ', 'ALBEDO                            ', &
     & 'ASYMMETRY                         ', 'GAS_SPECTRAL_INTERVAL             ', &
     & 'TRANSMITTANCE_TRESHOLD            ', 'SOLAR_SPECTRUM                    ', &
     & 'WATER_OPTICAL_CONSTANT            ', 'WAVE_SPECTRUM                     ', &
     & 'AEROSOLS_PARAMETERS               ', 'AEROSOLS_COMPONENTS               ', &
     & 'WATERCLOUD_TYPES                  ', 'WATERCLOUD_PARAMETERS             ', &
     & 'ICECLOUD_TYPES                    ', 'HEXAGONAL_PARAMETERS              ', &
     & 'AGGREGATE_PARAMETERS              ', 'PRINCOMP_PREDICTORS               ', &
     & 'PRINCOMP_EIGENVECTORS             ', 'PRINCOMP_COEFFICIENTS             ', &
     & 'EMISSIVITY_COEFFICIENTS           ', 'PC_REFERENCE_PROFILE              ', &
     & 'PC_PROFILE_LIMITS                 ', 'INSTRUMENT_NOISE                  ', &
     & 'PLANCK_WEIGHTED                   ', 'SOLAR_FAST_COEFFICIENTS           ', &
     & 'README_SPECTRAL_RESPONSE_FUNCTION ', 'NLTE_RADIANCE_COEFS               ', &
```

```
        & 'PRESSURE_MODULATED_CELL           '/)

  ! Gas ID codes
  INTEGER(KIND=jpim), PARAMETER :: ngases_max = 8
  INTEGER(KIND=jpim), PARAMETER :: &
        & gas_id_mixed       = 1,  &
        & gas_id_watervapour = 2,  &
        & gas_id_ozone       = 3,  &
        & gas_id_wvcont      = 4,  &
        & gas_id_co2         = 5,  &
        & gas_id_n2o         = 6,  &
        & gas_id_co          = 7,  &
        & gas_id_ch4         = 8!,  &
!         & gas_id_so2         = 9,  &
!         & gas_id_o2          = 10, &
!         & gas_id_no          = 11, &
!         & gas_id_no2         = 12, &
!         & gas_id_hno3        = 13, &
!         & gas_id_ocs         = 14, &
!         & gas_id_n2          = 15, &
!         & gas_id_ccl4        = 16, &
!         & gas_id_cfc11       = 17, &
!         & gas_id_cfc12       = 18, &
!         & gas_id_cfc14       = 19

  ! Gas names
  CHARACTER(LEN=12), PARAMETER :: gas_name(ngases_max) = &
        & (/ 'Mixed_gases ', &
           & 'Water_vapour', &
           & 'Ozone       ', &
           & 'WV_Continuum', &
           & 'CO2         ', &
           & 'N2O         ', &
           & 'CO          ', &
           & 'CH4         ' /) !, &
!            & 'SO2         ', &
!            & 'O2          ', &
!            & 'NO          ', &
!            & 'NO2         ', &
!            & 'HNO3        ', &
!            & 'OCS         ', &
!            & 'N2          ', &
!            & 'CCl4        ', &
!            & 'CFC-11      ', &
!            & 'CFC-12      ', &
!            & 'CFC-14      ' /)

  ! Convenient array of gas molecular masses
  REAL(KIND=jprb), PARAMETER :: gas_mass(ngases_max) = &
    & (/ mair, mh2o, mo3, mh2o, mco2, mn2o, mco, mch4 /) !, mso2, mo2, mno, &
!          mno2, mhno3, mocs, mn2, mccl4, mcfc11, mcfc12, mcfc14 /)

  ! Gas units
  INTEGER(KIND=jpim), PARAMETER :: ngases_unit = 2
  INTEGER(KIND=jpim), PARAMETER :: &
        & gas_unit_compatibility = 0, &   ! Used for input gas units to ensure behaviour like v11.2
and earlier
        & gas_unit_specconc  = 1, &     ! specific concentration (kg/kg over wet air)
        & gas_unit_ppmv      = 2        ! volume mixing ratio (ppmv over wet air)
  CHARACTER(LEN=12), PARAMETER :: gas_unit_name(ngases_unit) = &
        & (/ 'spec. concen', &
           & 'ppmv        '  /)

  ! Error reporting
  ! ---------------
  INTEGER(KIND=jpim), PARAMETER :: errorstatus_success = 0
  INTEGER(KIND=jpim), PARAMETER :: errorstatus_fatal   = 1

  ! Surface types
  ! -------------
  INTEGER(KIND=jpim), PARAMETER :: nsurftype = 2
  INTEGER(KIND=jpim), PARAMETER :: surftype_land = 0
  INTEGER(KIND=jpim), PARAMETER :: surftype_sea = 1
  INTEGER(KIND=jpim), PARAMETER :: surftype_seaice = 2
```

```
  ! Water types
  ! -----------
  INTEGER(KIND=jpim), PARAMETER :: nwatertype = 1
  INTEGER(KIND=jpim), PARAMETER :: watertype_fresh_water = 0
  INTEGER(KIND=jpim), PARAMETER :: watertype_ocean_water = 1

  ! Hard limits for control of input profile
  ! ----------------------------------------
  ! Temperature
  REAL(KIND=jprb), PARAMETER :: tmax   = 400.0_jprb       ! degK
  REAL(KIND=jprb), PARAMETER :: tmin   = 90.0_jprb        ! degK
  ! Water Vapour
  REAL(KIND=jprb), PARAMETER :: qmax   = 0.60E+06_jprb    ! ppmv 0.373_jprb kg/kg
  REAL(KIND=jprb), PARAMETER :: qmin   = 0.1E-10_jprb     ! ppmv
  ! Ozone
  REAL(KIND=jprb), PARAMETER :: o3max  = 1000.0_jprb      ! ppmv  1.657E-3_jprb kg/kg
  REAL(KIND=jprb), PARAMETER :: o3min  = 0.1E-10_jprb     ! ppmv
  ! CO2
  REAL(KIND=jprb), PARAMETER :: co2max = 1000.0_jprb      ! ppmv
  REAL(KIND=jprb), PARAMETER :: co2min = 0.1E-10_jprb     ! ppmv
  ! CO
  REAL(KIND=jprb), PARAMETER :: comax  = 10.0_jprb        ! ppmv
  REAL(KIND=jprb), PARAMETER :: comin  = 0.1E-10_jprb     ! ppmv
  ! N2O
  REAL(KIND=jprb), PARAMETER :: n2omax = 10.0_jprb        ! ppmv
  REAL(KIND=jprb), PARAMETER :: n2omin = 0.1E-10_jprb     ! ppmv
  ! CH4
  REAL(KIND=jprb), PARAMETER :: ch4max = 50.0_jprb        ! ppmv
  REAL(KIND=jprb), PARAMETER :: ch4min = 0.1E-10_jprb     ! ppmv
  ! Cloud Liquid Water
  REAL(KIND=jprb), PARAMETER :: clwmax = 1.0_jprb         ! kg/kg
  REAL(KIND=jprb), PARAMETER :: clwmin = 0.0_jprb         ! kg/kg
  ! Surface Pressure
  REAL(KIND=jprb), PARAMETER :: pmax   = 1100.0_jprb      ! surface pressure hPa
  REAL(KIND=jprb), PARAMETER :: pmin   = 400.0_jprb       ! hPa
  ! Surface Wind
  REAL(KIND=jprb), PARAMETER :: wmax   =  100.0_jprb      ! surface wind speed (m/s)
  ! Zenith Angle
  REAL(KIND=jprb), PARAMETER :: zenmax = 75.0_jprb        ! zenith angle (Deg) = secant 3.86_jprb
  REAL(KIND=jprb), PARAMETER :: zenmaxv9 = 85.3_jprb      ! larger zenmax for v9 predictors = secant
12
  ! Cloud Top Pressure
  REAL(KIND=jprb), PARAMETER :: ctpmax = 1100.0_jprb      ! (hPa)
  REAL(KIND=jprb), PARAMETER :: ctpmin =   50.0_jprb      ! (hPa)
  ! Magnetic field strength
  REAL(KIND=jprb), PARAMETER :: bemax = 0.7_jprb          ! (Gauss)
  REAL(KIND=jprb), PARAMETER :: bemin = 0.2_jprb          ! (Guass)
  ! Ice Crystal Diameter
  REAL(KIND=jprb), PARAMETER :: dgmin_hex =  12.2_jprb    ! (micron)
  REAL(KIND=jprb), PARAMETER :: dgmax_hex =  118.29_jprb  ! (micron)
  REAL(KIND=jprb), PARAMETER :: dgmin_agg =  5.61_jprb    ! (micron)
  REAL(KIND=jprb), PARAMETER :: dgmax_agg =  166.46_jprb  ! (micron)
  ! Ice Water Content
  REAL(KIND=jprb), PARAMETER :: iwcmin_hex =  0.000608_jprb ! (g.m-3)
  REAL(KIND=jprb), PARAMETER :: iwcmax_hex =  0.254639_jprb ! (g.m-3)
  REAL(KIND=jprb), PARAMETER :: iwcmin_agg =  0.000235_jprb ! (g.m-3)
  REAL(KIND=jprb), PARAMETER :: iwcmax_agg =  0.489046_jprb ! (g.m-3)


  ! Min/max optical depth and transmittance values
  ! ----------------------------------------------
  ! maximum value of optical depth for transmittance calculation
  ! e(-30) -> 10**-14
  ! e(-50) -> 10**-22
  REAL(KIND=jprb), PARAMETER :: max_optical_depth = 50._jprb
  REAL(KIND=jprb), PARAMETER :: min_tau = 1.0e-8_jprb
  REAL(KIND=jprb), PARAMETER :: min_od  = 1.0e-5_jprb


  ! Maximum solar zenith angle for which to apply solar calculation
  ! ---------------------------------------------------------------
  REAL(KIND=jprb), PARAMETER  :: max_sol_zen = 85.3_jprb ! = secant 12
```

```
! Auxiliary parameters
! --------------------
INTEGER(KIND=jpim), PARAMETER :: max_fastem_version = 6 ! Highest FASTEM version number available
INTEGER(KIND=jpim), PARAMETER :: fastem_sp = 5          ! Number of fastem surface parameters
REAL(KIND=jprb), PARAMETER    :: mwcldtp = 322.0_jprb   ! Upper pressure level (HPa) for lwp calcs
REAL(KIND=jprb), PARAMETER    :: dcoeff(8) = &          ! Debye coefs
       (/ 17.1252_jprb, 134.2450_jprb, 310.2125_jprb,  5.667_jprb, &
         188.7979_jprb,  80.5419_jprb,   0.1157_jprb,  4.8417_jprb/)

! Pressure [hPa] top of radiatively insignificant "low cloud" in simple IR cloud streams approach
REAL(KIND=jprb), PARAMETER :: cldstr_low_cloud_top = 750.0_jprb


! Polarisation definitions
! ------------------------
! == pol_id +1
!   1 average of vertical and horizontal
!   2 nominal vertical at nadir, rotating
!      with view angle
!   3 nominal horizontal at nadir, rotating
!      with view angle
!   4 vertical
!   5 horizontal
!   6 + 45 minus -45 (3rd stokes vector)
!   7 left circular - right circular (4th stokes vector)
INTEGER(KIND=jpim), PARAMETER :: npolar_compute(7) = &
 & (/ 2, 2, 2, 1, 1, 2, 4/)
INTEGER(KIND=jpim), PARAMETER :: npolar_return(7) = &
 & (/ 1, 1, 1, 1, 1, 2, 4/)

! pol_v and pol_h give proportion of v and h pol to use in emissivity calculation
! pol_s3 adds the 3rd/4th stokes vectors
REAL(KIND=jprb), PARAMETER :: pol_v(3,7) = RESHAPE( &
  & (/ 0.5_jprb, 0.0_jprb, 0.0_jprb, &
     & 0.0_jprb, 0.0_jprb, 1.0_jprb, &
     & 0.0_jprb, 1.0_jprb, 0.0_jprb, &
     & 1.0_jprb, 0.0_jprb, 0.0_jprb, &
     & 0.0_jprb, 0.0_jprb, 0.0_jprb, &
     & 0.0_jprb, 0.0_jprb, 0.0_jprb, &
     & 0.0_jprb, 0.0_jprb, 0.0_jprb  /), (/3,7/) )
REAL(KIND=jprb), PARAMETER :: pol_h(3,7) = RESHAPE( &
  & (/ 0.5_jprb, 0.0_jprb, 0.0_jprb, &
     & 0.0_jprb, 1.0_jprb, 0.0_jprb, &
     & 0.0_jprb, 0.0_jprb, 1.0_jprb, &
     & 0.0_jprb, 0.0_jprb, 0.0_jprb, &
     & 1.0_jprb, 0.0_jprb, 0.0_jprb, &
     & 0.0_jprb, 0.0_jprb, 0.0_jprb, &
     & 0.0_jprb, 0.0_jprb, 0.0_jprb  /), (/3,7/) )
REAL(KIND=jprb), PARAMETER :: pol_s3(0:1,7) = RESHAPE( &
  & (/ 0.0_jprb, 0.0_jprb, &
     & 0.0_jprb, 0.0_jprb, &
     & 0.0_jprb, 0.0_jprb, &
     & 0.0_jprb, 0.0_jprb, &
     & 0.0_jprb, 0.0_jprb, &
     & 1.0_jprb, 0.0_jprb, &
     & 0.0_jprb, 1.0_jprb  /), (/2,7/) )

! RTTOVSCATT aux parameters
! ------------------------
! Pressure of top level for line by line calculations (hPa)
REAL(KIND=jprb), PARAMETER :: pressure_top = 0.004985_jprb
! Minimum cloud cover processed by rttov_scatt
REAL(KIND=jprb), PARAMETER :: ccthres = 0.05_jprb
! Minimum single scattering albedo processed by rttov_scatt
REAL(KIND=jprb), PARAMETER :: min_ssa = 1.0E-03_jprb
! Rain density (g.cm-3)
REAL(KIND=jprb), PARAMETER :: rho_rain = 1.0_jprb
! Snow density (g.cm-3)
REAL(KIND=jprb), PARAMETER :: rho_snow = 0.1_jprb

! Flags to identify function in shared K/Adjoint routines
INTEGER(KIND=jpim), PARAMETER :: adk_adjoint = 0
INTEGER(KIND=jpim), PARAMETER :: adk_k       = 1
```

```
! Parameters to compute refractive index of air
! ----------------------------------------------
REAL(KIND=jprb), PARAMETER :: D1   =8341.87_jprb
REAL(KIND=jprb), PARAMETER :: D2   =2405955.0_jprb
REAL(KIND=jprb), PARAMETER :: D3   =130.0_jprb
REAL(KIND=jprb), PARAMETER :: D4   =15996.0_jprb
REAL(KIND=jprb), PARAMETER :: D5   =38.9_jprb
REAL(KIND=jprb), PARAMETER :: DCO2 =0.540_jprb
REAL(KIND=jprb), PARAMETER :: ED1  =96095.43_jprb
REAL(KIND=jprb), PARAMETER :: ED2  =0.601_jprb
REAL(KIND=jprb), PARAMETER :: ED3  =0.00972_jprb
REAL(KIND=jprb), PARAMETER :: ED4  =0.003661_jprb
REAL(KIND=jprb), PARAMETER :: EW1  =3.7345_jprb
REAL(KIND=jprb), PARAMETER :: EW2  =0.0401_jprb
REAL(KIND=jprb), PARAMETER :: HTOP =100.0_jprb
REAL(KIND=jprb), PARAMETER :: CTOM =1.0E-4_jprb
REAL(KIND=jprb), PARAMETER :: WAVER=1700.0_jprb

! CO2 concentration assumed for atmospheric refractivity calculation
REAL(KIND=jprb), PARAMETER :: co2_conc = 376._jprb


! IR scattering parameters
! ------------------------

! Aerosols
INTEGER(KIND=jpim), PARAMETER :: naer_max = 13

INTEGER(KIND=jpim), PARAMETER :: &
      & aer_id_inso       = 1, &
      & aer_id_waso       = 2, &
      & aer_id_soot       = 3, &
      & aer_id_ssam       = 4, &
      & aer_id_sscm       = 5, &
      & aer_id_minm       = 6, &
      & aer_id_miam       = 7, &
      & aer_id_micm       = 8, &
      & aer_id_mitr       = 9, &
      & aer_id_suso       =10, &
      & aer_id_vola       =11, &
      & aer_id_vapo       =12, &
      & aer_id_asdu       =13

CHARACTER(LEN=4), PARAMETER :: aer_name(naer_max) = &
      & (/ 'inso', &
         & 'waso', &
         & 'soot', &
         & 'ssam', &
         & 'sscm', &
         & 'minm', &
         & 'miam', &
         & 'micm', &
         & 'mitr', &
         & 'suso', &
         & 'vola', &
         & 'vapo', &
         & 'asdu'  /)

! Constants for relative humidity calculation
REAL(KIND=jprb), PARAMETER :: E00 = 611.21_jprb
REAL(KIND=jprb), PARAMETER :: T00 = 273.16_jprb
REAL(KIND=jprb), PARAMETER :: TI  = T00 - 23.0_jprb

! Phase functions
INTEGER(KIND=jpim), PARAMETER :: nphangle = 208

REAL(KIND=jprb), PARAMETER :: phangle(nphangle) = &
           (/   0.0_jprb,   0.1_jprb,   0.2_jprb,   0.3_jprb,   0.4_jprb,   0.5_jprb,   0.6_jprb,&
            &   0.7_jprb,   0.8_jprb,   0.9_jprb,   1.0_jprb,   1.1_jprb,   1.2_jprb,   1.3_jprb,&
            &   1.4_jprb,   1.5_jprb,   1.6_jprb,   1.7_jprb,   1.8_jprb,   1.9_jprb,   2.0_jprb,&
            &   2.1_jprb,   2.2_jprb,   2.3_jprb,   2.4_jprb,   2.5_jprb,   2.6_jprb,   2.7_jprb,&
            &   2.8_jprb,   2.9_jprb,   3.0_jprb,   4.0_jprb,   5.0_jprb,   6.0_jprb,   7.0_jprb,&
            &   8.0_jprb,   9.0_jprb,  10.0_jprb,  11.0_jprb,  12.0_jprb,  13.0_jprb,  14.0_jprb,&
```

```
      &  15.0_jprb,  16.0_jprb,  17.0_jprb,  18.0_jprb,  19.0_jprb,  20.0_jprb,  21.0_jprb,&
      &  22.0_jprb,  23.0_jprb,  24.0_jprb,  25.0_jprb,  26.0_jprb,  27.0_jprb,  28.0_jprb,&
      &  29.0_jprb,  30.0_jprb,  31.0_jprb,  32.0_jprb,  33.0_jprb,  34.0_jprb,  35.0_jprb,&
      &  36.0_jprb,  37.0_jprb,  38.0_jprb,  39.0_jprb,  40.0_jprb,  41.0_jprb,  42.0_jprb,&
      &  43.0_jprb,  44.0_jprb,  45.0_jprb,  46.0_jprb,  47.0_jprb,  48.0_jprb,  49.0_jprb,&
      &  50.0_jprb,  51.0_jprb,  52.0_jprb,  53.0_jprb,  54.0_jprb,  55.0_jprb,  56.0_jprb,&
      &  57.0_jprb,  58.0_jprb,  59.0_jprb,  60.0_jprb,  61.0_jprb,  62.0_jprb,  63.0_jprb,&
      &  64.0_jprb,  65.0_jprb,  66.0_jprb,  67.0_jprb,  68.0_jprb,  69.0_jprb,  70.0_jprb,&
      &  71.0_jprb,  72.0_jprb,  73.0_jprb,  74.0_jprb,  75.0_jprb,  76.0_jprb,  77.0_jprb,&
      &  78.0_jprb,  79.0_jprb,  80.0_jprb,  81.0_jprb,  82.0_jprb,  83.0_jprb,  84.0_jprb,&
      &  85.0_jprb,  86.0_jprb,  87.0_jprb,  88.0_jprb,  89.0_jprb,  90.0_jprb,  91.0_jprb,&
      &  92.0_jprb,  93.0_jprb,  94.0_jprb,  95.0_jprb,  96.0_jprb,  97.0_jprb,  98.0_jprb,&
      &  99.0_jprb, 100.0_jprb, 101.0_jprb, 102.0_jprb, 103.0_jprb, 104.0_jprb, 105.0_jprb,&
      & 106.0_jprb, 107.0_jprb, 108.0_jprb, 109.0_jprb, 110.0_jprb, 111.0_jprb, 112.0_jprb,&
      & 113.0_jprb, 114.0_jprb, 115.0_jprb, 116.0_jprb, 117.0_jprb, 118.0_jprb, 119.0_jprb,&
      & 120.0_jprb, 121.0_jprb, 122.0_jprb, 123.0_jprb, 124.0_jprb, 125.0_jprb, 126.0_jprb,&
      & 127.0_jprb, 128.0_jprb, 129.0_jprb, 130.0_jprb, 131.0_jprb, 132.0_jprb, 133.0_jprb,&
      & 134.0_jprb, 135.0_jprb, 136.0_jprb, 137.0_jprb, 138.0_jprb, 139.0_jprb, 140.0_jprb,&
      & 141.0_jprb, 142.0_jprb, 143.0_jprb, 144.0_jprb, 145.0_jprb, 146.0_jprb, 147.0_jprb,&
      & 148.0_jprb, 149.0_jprb, 150.0_jprb, 151.0_jprb, 152.0_jprb, 153.0_jprb, 154.0_jprb,&
      & 155.0_jprb, 156.0_jprb, 157.0_jprb, 158.0_jprb, 159.0_jprb, 160.0_jprb, 161.0_jprb,&
      & 162.0_jprb, 163.0_jprb, 164.0_jprb, 165.0_jprb, 166.0_jprb, 167.0_jprb, 168.0_jprb,&
      & 169.0_jprb, 170.0_jprb, 171.0_jprb, 172.0_jprb, 173.0_jprb, 174.0_jprb, 175.0_jprb,&
      & 176.0_jprb, 177.0_jprb, 178.0_jprb, 179.0_jprb, 180.0_jprb /)

  ! Water clouds
  INTEGER(KIND=jpim), PARAMETER :: nwcl_max = 5

  INTEGER(KIND=jpim), PARAMETER :: &
        & wcl_id_stco       = 1, &
        & wcl_id_stma       = 2, &
        & wcl_id_cucc       = 3, &
        & wcl_id_cucp       = 4, &
        & wcl_id_cuma       = 5

  CHARACTER(LEN=4), PARAMETER :: wcl_name(nwcl_max) = &
        & (/ 'stco', &
           & 'stma', &
           & 'cucc', &
           & 'cucp', &
           & 'cuma' /)

  INTEGER(KIND=jpim), PARAMETER:: ncldtyp = 6

  ! Ice clouds
  INTEGER(KIND=jpim), PARAMETER :: nish = 4 ! Max valid value for ish
  INTEGER(KIND=jpim), PARAMETER :: nidg = 4 ! Max valid value for idg


  ! Wavenumber intervals for RTTOV9 predictors
  ! ------------------------------------------
  REAL(KIND=jprb), PARAMETER :: rttov9_wv0690_50 =  690.50_jprb, &
                                rttov9_wv1050_00 = 1050.00_jprb, &
                                rttov9_wv1095_25 = 1095.25_jprb, &
                                rttov9_wv1100_25 = 1100.25_jprb, &
                                rttov9_wv1350_25 = 1350.25_jprb, &
                                rttov9_wv1750_25 = 1750.25_jprb, &
                                rttov9_wv1900_25 = 1900.25_jprb, &
                                rttov9_wv1995_00 = 1995.00_jprb, &
                                rttov9_wv2000_00 = 2000.00_jprb, &
                                rttov9_wv2250_00 = 2250.00_jprb, &
                                rttov9_wv2295_25 = 2295.25_jprb, &
                                rttov9_wv2360_00 = 2360.00_jprb, &
                                rttov9_wv2380_25 = 2380.25_jprb, &
                                rttov9_wv2660_25 = 2660.25_jprb, &
                                rttov9_wv2760_25 = 2760.25_jprb


  ! Parameters for solar overcast radiance calculation
  ! --------------------------------------------------
  REAL(KIND=jprb), PARAMETER :: overcast_albedo_wvn = 10000._jprb ! Wavenumber (cm-1) at which
albedo changes
  REAL(KIND=jprb), PARAMETER :: overcast_albedo1    = 0.7_jprb    ! Overcast albedo for wvn > limit
  REAL(KIND=jprb), PARAMETER :: overcast_albedo2    = 0.6_jprb    ! Overcast albedo for wvn <= limit
```

```
  ! Parameters for Rayleigh cross-section parameterization taken from Bucholzt 1995
  ! ---------------------------------------------------------------------------
  REAL(KIND=jprb), PARAMETER :: ray_min_wvn = 5000.0_jprb,     & ! Min wavenumber (cm-1) for which
Rayleigh is calculated
                                ray_scs_wlm = 0.5_jprb,        & ! Wavelength limit: below 0.5um the
                                ray_scs_a1 = 3.01577E-28_jprb, & !   first set of parameters a1-d1
are used
                                ray_scs_b1 = -3.55212_jprb,    & !   while above this a2-d2 are
used.
                                ray_scs_c1 = -1.35579_jprb,    &
                                ray_scs_d1 = -0.11563_jprb,    &
                                ray_scs_a2 = 4.01061E-28_jprb, &
                                ray_scs_b2 = -3.99668_jprb,    &
                                ray_scs_c2 = -1.10298E-3_jprb, &
                                ray_scs_d2 = -2.71393E-2_jprb

  ! Interpolation modes
  ! -------------------
  !       MODE                     USER->COEF LEVEL          COEF->USER LEVEL
  !                               (profile interpolation)   (optical depth/weighting fn interpolation)
  ! interp_rochon:                      Rochon                  Rochon,     op dep
  ! interp_loglinear:                   Log-linear              Log-linear, op dep
  ! interp_rochon_loglinear:            Rochon                  Log-linear, op dep
  ! interp_rochon_wfn:                  Rochon                  Rochon,     weighting fns
  ! interp_rochon_loglinear_wfn:        Rochon                  Log-linear, weighting fns
  INTEGER(KIND=jpim), PARAMETER :: ninterp_modes              = 5_jpim ! Number of valid
interpolation options
  INTEGER(KIND=jpim), PARAMETER :: interp_rochon              = 1_jpim
  INTEGER(KIND=jpim), PARAMETER :: interp_loglinear           = 2_jpim
  INTEGER(KIND=jpim), PARAMETER :: interp_rochon_loglinear    = 3_jpim
  INTEGER(KIND=jpim), PARAMETER :: interp_rochon_wfn          = 4_jpim
  INTEGER(KIND=jpim), PARAMETER :: interp_rochon_loglinear_wfn = 5_jpim

END MODULE rttov_const
```

Annex Q – Example user interface program to run RTTOV

```
PROGRAM example_fwd
  !
  ! Copyright:
  !    This software was developed within the context of
  !    the EUMETSAT Satellite Application Facility on
  !    Numerical Weather Prediction (NWP SAF), under the
  !    Cooperation Agreement dated 25 November 1998, 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, KNMI and MeteoFrance.
  !
  !    Copyright 2010, EUMETSAT, All Rights Reserved.
  !
  !     ************************************************************
  !
  !     TEST PROGRAM FOR RTTOV FORWARD MODEL
  !          RTTOV VERSION 11
  !
  ! To run this program you must have the following files
  ! either resident in the same directory or set up as a
  ! symbolic link:
  !   the file containing input profiles (e.g. prof.dat)
  !   the RTTOV coefficient file
  !
  ! The script run_example_fwd.sh may be used to run this program.
  ! The output is generated in a file called example_fwd_output.dat.
  !
  !
  ! To adapt the code to their own applications users should
  ! edit the code highlighted like this:
  !     !================================
  !     !======Read =====start===========
  !          code to be modified
  !     !======Read ===== end ===========
  !     !================================
  !
  ! Current Code Owner: SAF NWP
  !
  ! Code Description:
  !   Language:          Fortran 90.
  !   Software Standards: "European Standards for Writing and
  !     Documenting Exchangeable Fortran 90 Code".
  !

  ! rttov_const contains useful RTTOV constants
  USE rttov_const, ONLY :     &
       & errorstatus_success, &
       & errorstatus_fatal,   &
       & platform_name,       &
       & inst_name

  ! rttov_types contains definitions of all RTTOV data types
  USE rttov_types, ONLY :     &
       & rttov_options,       &
       & rttov_coefs,         &
       & profile_type,        &
       & transmission_type,   &
       & radiance_type,       &
       & rttov_chanprof,      &
       & rttov_emissivity,    &
       & rttov_reflectance

  ! jpim, jprb and jplm are the RTTOV integer, real and logical KINDs
  USE parkind1, ONLY : jpim, jprb, jplm

  USE rttov_unix_env, ONLY : rttov_exit

  IMPLICIT NONE

#include "rttov_direct.interface"
#include "rttov_parallel_direct.interface"
```

```
#include "rttov_read_coefs.interface"
#include "rttov_dealloc_coefs.interface"
#include "rttov_alloc_direct.interface"
#include "rttov_user_options_checkinput.interface"
#include "rttov_print_opts.interface"
#include "rttov_print_profile.interface"
#include "rttov_skipcommentline.interface"

  !-------------------------
  !
  INTEGER(KIND=jpim), PARAMETER :: iup   = 20   ! unit for input profile file
  INTEGER(KIND=jpim), PARAMETER :: ioout = 21   ! unit for output

  ! RTTOV variables/structures
  !====================
  TYPE(rttov_options)              :: opts                     ! Options structure
  TYPE(rttov_coefs)                :: coefs                    ! Coefficients structure
  TYPE(rttov_chanprof),    POINTER :: chanprof(:)    => NULL() ! Input channel/profile list
  LOGICAL(KIND=jplm),      POINTER :: calcemis(:)    => NULL() ! Flag to indicate calculation of
emissivity within RTTOV
  TYPE(rttov_emissivity),  POINTER :: emissivity(:)  => NULL() ! Input/output surface emissivity
  LOGICAL(KIND=jplm),      POINTER :: calcrefl(:)    => NULL() ! Flag to indicate calculation of
BRDF within RTTOV
  TYPE(rttov_reflectance), POINTER :: reflectance(:) => NULL() ! Input/output surface BRDF
  TYPE(profile_type),      POINTER :: profiles(:)    => NULL() ! Input profiles
  TYPE(transmission_type)          :: transmission             ! Output transmittances
  TYPE(radiance_type)              :: radiance                 ! Output radiances

  INTEGER(KIND=jpim)               :: errorstatus              ! Return error status of RTTOV
subroutine calls

  INTEGER(KIND=jpim) :: alloc_status
  CHARACTER(LEN=11)  :: NameOfRoutine = 'example_fwd'

  ! variables for input
  !====================
  CHARACTER(LEN=256) :: coef_filename
  CHARACTER(LEN=256) :: prof_filename
  INTEGER(KIND=jpim) :: nthreads
  INTEGER(KIND=jpim) :: dosolar
  INTEGER(KIND=jpim) :: nlevels
  INTEGER(KIND=jpim) :: nprof
  INTEGER(KIND=jpim) :: nchannels
  INTEGER(KIND=jpim) :: nchanprof
  INTEGER(KIND=jpim), ALLOCATABLE :: channel_list(:)
  REAL(KIND=jprb)    :: trans_out(10)
  ! loop variables
  INTEGER(KIND=jpim) :: j, jch
  INTEGER(KIND=jpim) :: np, nch
  INTEGER(KIND=jpim) :: ilev, nprint
  INTEGER(KIND=jpim) :: iprof, joff
  INTEGER            :: ios

  !- End of header ------------------------------------------------------

  ! The usual steps to take when running RTTOV are as follows:
  !   1. Specify required RTTOV options
  !   2. Read coefficients
  !   3. Allocate RTTOV input and output structures
  !   4. Set up the chanprof array with the channels/profiles to simulate
  !   5. Read input profile(s)
  !   6. Set up surface emissivity and/or reflectance
  !   7. Call rttov_direct and store results
  !   8. Deallocate all structures and arrays

  ! If nthreads is greater than 1 the parallel RTTOV interface is used.
  ! To take advantage of multi-threaded execution you must have compiled
  ! RTTOV with openmp enabled. See the user guide and the compiler flags.

  errorstatus = 0_jpim

  !=====================================================
  !========== Interactive inputs == start ==============
```

```
WRITE(0,*) 'enter name of coefficient file (in current directory)'
READ(*,*) coef_filename
WRITE(0,*) 'enter name of file containing profile data (in current directory)'
READ(*,*) prof_filename
WRITE(0,*) 'enter number of profiles'
READ(*,*) nprof
WRITE(0,*) 'enter number of profile levels'
READ(*,*) nlevels
WRITE(0,*) 'turn on solar simulations? (0=no, 1=yes)'
READ(*,*) dosolar
WRITE(0,*) 'enter number of channels to simulate per profile'
READ(*,*) nchannels
ALLOCATE(channel_list(nchannels))
WRITE(0,*) 'enter space-separated channel list'
READ(*,*,iostat=ios) channel_list(:)
WRITE(0,*) 'enter number of threads to use'
READ(*,*) nthreads


! ----------------------------------------------------------------------
! 1. Initialise RTTOV options structure
! ----------------------------------------------------------------------

IF (dosolar == 1) THEN
  opts % rt_ir % addsolar = .TRUE.           ! Include solar radiation
ELSE
  opts % rt_ir % addsolar = .FALSE.          ! Do not include solar radiation
ENDIF
opts % interpolation % addinterp   = .TRUE.  ! Allow interpolation of input profile
opts % interpolation % interp_mode = 1       ! Set interpolation method
opts % rt_all % addrefrac          = .TRUE.  ! Include refraction in path calc
opts % rt_ir % addclouds           = .FALSE. ! Don't include cloud effects
opts % rt_ir % addaerosl           = .FALSE. ! Don't include aerosol effects

opts % rt_ir % ozone_data          = .FALSE. ! Set the relevant flag to .TRUE.
opts % rt_ir % co2_data            = .FALSE. !   when supplying a profile of the
opts % rt_ir % n2o_data            = .FALSE. !   given trace gas (ensure the
opts % rt_ir % ch4_data            = .FALSE. !   coef file supports the gas)
opts % rt_ir % co_data             = .FALSE. !
opts % rt_mw % clw_data            = .FALSE. !

opts % config % verbose            = .TRUE.  ! Enable printing of warnings

!========== Interactive inputs == end ==============
!===================================================


! ----------------------------------------------------------------------
! 2. Read coefficients
! ----------------------------------------------------------------------
CALL rttov_read_coefs(errorstatus, coefs, opts, file_coef=coef_filename)
IF (errorstatus /= errorstatus_success) THEN
  WRITE(*,*) 'fatal error reading coefficients'
  CALL rttov_exit(errorstatus)
ENDIF

! Ensure input number of channels is not higher than number stored in coefficient file
IF (nchannels > coefs % coef % fmv_chn) THEN
  nchannels = coefs % coef % fmv_chn
ENDIF

! Ensure the options and coefficients are consistent
CALL rttov_user_options_checkinput(errorstatus, opts, coefs)
IF (errorstatus /= errorstatus_success) THEN
  WRITE(*,*) 'error in rttov options'
  CALL rttov_exit(errorstatus)
ENDIF


! ----------------------------------------------------------------------
! 3. Allocate RTTOV input and output structures
! ----------------------------------------------------------------------

! Determine the total number of radiances to simulate (nchanprof).
```

```
  ! In this example we simulate all specified channels for each profile, but
  ! in general one can simulate a different number of channels for each profile.

  nchanprof = nchannels * nprof

  ! Allocate structures for rttov_direct
  CALL rttov_alloc_direct( &
      & errorstatus,             &
      & 1_jpim,                  &  ! 1 => allocate
      & nprof,                   &
      & nchanprof,               &
      & nlevels,                 &
      & chanprof,                &
      & opts,                    &
      & profiles,                &
      & coefs,                   &
      & transmission,            &
      & radiance,                &
      & calcemis=calcemis,       &
      & emissivity=emissivity,   &
      & calcrefl=calcrefl,       &
      & reflectance=reflectance, &
      & init=.TRUE._jplm)
  IF (errorstatus /= errorstatus_success) THEN
    WRITE(*,*) 'allocation error for rttov_direct structures'
    CALL rttov_exit(errorstatus)
  ENDIF


  ! ------------------------------------------------------------------------
  ! 4. Build the list of profile/channel indices in chanprof
  ! ------------------------------------------------------------------------

  nch = 0_jpim
  DO j = 1, nprof
    DO jch = 1, nchannels
      nch = nch + 1_jpim
      chanprof(nch)%prof = j
      chanprof(nch)%chan = channel_list(jch)
    ENDDO
  ENDDO


  ! ------------------------------------------------------------------------
  ! 5. Read profile data
  ! ------------------------------------------------------------------------

  !================================================
  !========== Read profiles == start =============

  OPEN(iup, file=TRIM(prof_filename), status='old', iostat=ios)
  IF (ios /= 0) THEN
    WRITE(*,*) 'error opening profile file ios= ', ios
    CALL rttov_exit(errorstatus_fatal)
  ENDIF
  CALL rttov_skipcommentline(iup, errorstatus)

  ! Read gas units for profiles
  READ(iup,*) profiles(1) % gas_units
  profiles(:) % gas_units = profiles(1) % gas_units
  CALL rttov_skipcommentline(iup, errorstatus)

  ! Loop over all profiles and read data for each one
  DO iprof = 1, nprof

    ! Read pressure (hPa), temp (K), WV, O3 (gas units ppmv or kg/kg - as read above)
    READ(iup,*) profiles(iprof) % p(:)
    CALL rttov_skipcommentline(iup, errorstatus)
    READ(iup,*) profiles(iprof) % t(:)
    CALL rttov_skipcommentline(iup, errorstatus)
    READ(iup,*) profiles(iprof) % q(:)
    CALL rttov_skipcommentline(iup, errorstatus)
    ! Ozone profile is commented out in input profile data
!     READ(iup,*) profiles(iprof) % o3(:)
```

```
!     CALL rttov_skipcommentline(iup, errorstatus)

    ! 2 meter air variables
    READ(iup,*) profiles(iprof) % s2m % t, &
              & profiles(iprof) % s2m % q, &
              & profiles(iprof) % s2m % p, &
              & profiles(iprof) % s2m % u, &
              & profiles(iprof) % s2m % v, &
              & profiles(iprof) % s2m % wfetc
    CALL rttov_skipcommentline(iup, errorstatus)

    ! Skin variables
    READ(iup,*) profiles(iprof) % skin % t, &
              & profiles(iprof) % skin % fastem   ! FASTEM only applies to MW instruments
    CALL rttov_skipcommentline(iup, errorstatus)

    ! Surface type and water type
    READ(iup,*) profiles(iprof) % skin % surftype, &
              & profiles(iprof) % skin % watertype
    CALL rttov_skipcommentline(iup, errorstatus)

    ! Elevation, latitude and longitude
    READ(iup,*) profiles(iprof) % elevation, &
              & profiles(iprof) % latitude,  &
              & profiles(iprof) % longitude
    CALL rttov_skipcommentline(iup, errorstatus)

    ! Satellite and solar angles
    READ(iup,*) profiles(iprof) % zenangle,    &
              & profiles(iprof) % azangle,     &
              & profiles(iprof) % sunzenangle, &
              & profiles(iprof) % sunazangle
    CALL rttov_skipcommentline(iup, errorstatus)

    ! Cloud variables for simple cloud scheme, set cfraction to 0. to turn this off (VIS/IR only)
    READ(iup,*) profiles(iprof) % ctp, &
              & profiles(iprof) % cfraction
    CALL rttov_skipcommentline(iup, errorstatus)

  ENDDO
  CLOSE(iup)

  !========== READ profiles == end =============
  !=============================================


  ! ----------------------------------------------------------------------------
  ! 6. Specify surface emissivity and reflectance
  ! ----------------------------------------------------------------------------

  ! In this example we have no values for input emissivities
  emissivity(:) % emis_in = 0._jprb

  ! Calculate emissivity within RTTOV where the input emissivity value is
  ! zero or less (all channels in this case)
  calcemis(:) = (emissivity(:) % emis_in <= 0._jprb)

  ! In this example we have no values for input reflectances
  reflectance(:) % refl_in = 0._jprb

  ! Calculate BRDF within RTTOV where the input BRDF value is zero or less
  ! (all channels in this case)
  calcrefl(:) = (reflectance(:) % refl_in <= 0._jprb)

  ! Use default cloud top BRDF for simple cloud in VIS/NIR channels
  reflectance(:) % refl_cloud_top = 0._jprb


  ! ----------------------------------------------------------------------------
  ! 7. Call RTTOV forward model
  ! ----------------------------------------------------------------------------
  IF (nthreads <= 1) THEN
    CALL rttov_direct(               &
          & errorstatus,             &! out   error flag
```

```
          & chanprof,                &! in    channel and profile index structure
          & opts,                    &! in    options structure
          & profiles,                &! in    profile array
          & coefs,                   &! in    coefficients structure
          & transmission,            &! inout computed transmittances
          & radiance,                &! inout computed radiances
          & calcemis    = calcemis,  &! in    flag for internal emissivity calcs
          & emissivity  = emissivity,&! inout input/output emissivities per channel
          & calcrefl    = calcrefl,  &! in    flag for internal BRDF calcs
          & reflectance = reflectance) ! inout input/output BRDFs per channel
  ELSE
    CALL rttov_parallel_direct(     &
          & errorstatus,             &! out   error flag
          & chanprof,                &! in    channel and profile index structure
          & opts,                    &! in    options structure
          & profiles,                &! in    profile array
          & coefs,                   &! in    coefficients structure
          & transmission,            &! inout computed transmittances
          & radiance,                &! inout computed radiances
          & calcemis    = calcemis,  &! in    flag for internal emissivity calcs
          & emissivity  = emissivity,&! inout input/output emissivities per channel
          & calcrefl    = calcrefl,  &! in    flag for internal BRDF calcs
          & reflectance = reflectance,&! inout input/output BRDFs per channel
          & nthreads    = nthreads)   ! in    number of threads to use
  ENDIF

  IF (errorstatus /= errorstatus_success) THEN
    WRITE (*,*) 'rttov_direct error'
    CALL rttov_exit(errorstatus)
  ENDIF


  ! --- Output the results ------------------------------------------------

  ! Open output file where results are written
  OPEN(ioout, file='output_'//NameOfRoutine//'.dat', status='unknown', form='formatted', iostat=ios)
  IF (ios /= 0) THEN
    WRITE(*,*) 'error opening the output file ios= ', ios
    CALL rttov_exit(errorstatus_fatal)
  ENDIF

  WRITE(ioout,*)' -----------------'
  WRITE(ioout,*)' Instrument ', inst_name(coefs % coef % id_inst)
  WRITE(ioout,*)' -----------------'
  WRITE(ioout,*)' '
  CALL rttov_print_opts(opts, lu=ioout)

  DO iprof = 1, nprof

    joff = (iprof-1_jpim) * nchannels

    nprint = 1 + INT((nchannels-1)/10)
    WRITE(ioout,*)' '
    WRITE(ioout,*)' Profile ', iprof

    CALL rttov_print_profile(profiles(iprof), lu=ioout)

    WRITE(ioout,777)'CHANNELS PROCESSED FOR SAT ', platform_name(coefs % coef % id_platform), coefs
% coef % id_sat
    WRITE(ioout,111) (chanprof(j) % chan, j = 1+joff, nchannels+joff)
    WRITE(ioout,*)' '
    WRITE(ioout,*)'CALCULATED BRIGHTNESS TEMPERATURES (K):'
    WRITE(ioout,222) (radiance % bt(j), j = 1+joff, nchannels+joff)
    IF (opts % rt_ir % addsolar) THEN
      WRITE(ioout,*)' '
      WRITE(ioout,*)'CALCULATED SATELLITE REFLECTANCES (BRF):'
      WRITE(ioout,444) (radiance % refl(j), j = 1+joff, nchannels+joff)
    ENDIF
    WRITE(ioout,*)' '
    WRITE(ioout,*)'CALCULATED RADIANCES (mW/m2/sr/cm-1):'
    WRITE(ioout,222) (radiance % total(j), j = 1+joff, nchannels+joff)
    WRITE(ioout,*)' '
    WRITE(ioout,*)'CALCULATED OVERCAST RADIANCES:'
    WRITE(ioout,222) (radiance % cloudy(j), j = 1+joff, nchannels+joff)
```

```
  WRITE(ioout,*)' '
  WRITE(ioout,*)'CALCULATED SURFACE TO SPACE TRANSMITTANCE:'
  WRITE(ioout,4444) (transmission % tau_total(j), j = 1+joff, nchannels+joff)
  WRITE(ioout,*)' '
  WRITE(ioout,*)'CALCULATED SURFACE EMISSIVITIES:'
  WRITE(ioout,444) (emissivity(j) % emis_out, j = 1+joff, nchannels+joff)
  IF (opts % rt_ir % addsolar) THEN
    WRITE(ioout,*)' '
    WRITE(ioout,*)'CALCULATED SURFACE BRDF:'
    WRITE(ioout,444) (reflectance(j) % refl_out, j = 1+joff, nchannels+joff)
  ENDIF

  IF (nchannels <= 20) THEN
    DO np = 1, nprint
        WRITE(ioout,*)' '
        WRITE(ioout,*)'Level to space transmittances for channels'
        WRITE(ioout,1115) (chanprof(j+joff) % chan, &
                & j = 1+(np-1)*10, MIN(np*10, nchannels))
        DO ilev = 1, nlevels
          DO j = 1 + (np-1)*10, MIN(np*10, nchannels)
            ! Select transmittance based on channel type (VIS/NIR or IR)
            IF (coefs % coef % ss_val_chn(chanprof(j+joff) % chan) == 2) THEN
              trans_out(j - (np-1)*10) = transmission % tausun_levels_path1(ilev,j+joff)
            ELSE
              trans_out(j - (np-1)*10) = transmission % tau_levels(ilev,j+joff)
            ENDIF
          ENDDO
          WRITE(ioout,4445) ilev, trans_out(1:j-1-(np-1)*10)
        ENDDO
        WRITE(ioout,1115) (chanprof(j+joff) % chan, &
                & j = 1+(np-1)*10, MIN(np*10, nchannels))
    ENDDO
  ENDIF
ENDDO

! Close output file
CLOSE(ioout, iostat=ios)
IF (ios /= 0) THEN
  WRITE(*,*) 'error closing the output file ios= ', ios
  CALL rttov_exit(errorstatus_fatal)
ENDIF

! --- End of output section ---------------------------------------------


! ----------------------------------------------------------------------
! 8. Deallocate all RTTOV arrays and structures
! ----------------------------------------------------------------------
DEALLOCATE (channel_list, stat=alloc_status)
IF (alloc_status /= 0) THEN
  WRITE(*,*) 'mem dellocation error'
ENDIF

! Deallocate structures for rttov_direct
CALL rttov_alloc_direct( &
    & errorstatus,             &
    & 0_jpim,                  &  ! 0 => deallocate
    & nprof,                   &
    & nchanprof,               &
    & nlevels,                 &
    & chanprof,                &
    & opts,                    &
    & profiles,                &
    & coefs,                   &
    & transmission,            &
    & radiance,                &
    & calcemis=calcemis,       &
    & emissivity=emissivity,   &
    & calcrefl=calcrefl,       &
    & reflectance=reflectance)
IF (errorstatus /= errorstatus_success) THEN
  WRITE(*,*) 'deallocation error for rttov_direct structures'
  CALL rttov_exit(errorstatus)
ENDIF
```

```
  CALL rttov_dealloc_coefs(errorstatus, coefs)
  IF (errorstatus /= errorstatus_success) THEN
    WRITE(*,*) 'coefs deallocation error'
  ENDIF


! Format definitions for output
111  FORMAT(1X,10I8)
1115 FORMAT(3X,10I8)
222  FORMAT(1X,10F8.2)
444  FORMAT(1X,10F8.3)
4444 FORMAT(1X,10F8.4)
4445 FORMAT(1X,I2,10F8.4)
777  FORMAT(/,A,A8,I3)

END PROGRAM example_fwd
```

End of User Guide