RTTOV v10 Users Guide

James Hocking, Peter Rayer and Roger Saunders
Met Office, Exeter, UK
&
Marco Matricardi and Alan Geer
ECMWF
&
Pascal Brunel
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	04/06/10	R. Saunders	Initial draft
0.2	06/07/10	RS/JH/PR/MM/AG/	Modifications to include RTTOV v10 changes
0.3	16/07/10	JH/RS	Beta test version
0.4	12/11/10	JH	Incorporate comments and changes from beta test phase
0.5	15/11/10	R. Saunders	Comments on JH draft
1.0	22/11/10	RS/PJR/JH	Final changes from review comments
1.1	09/12/10	JH	Response to DRI comments
1.2	22/12/10	JH	Updates following DRI teleconference
1.3	19/01/11	JH	Minor updates before release
1.4	19/12/11	JH	Updated for RTTOV v10.2
1.5	12/01/11	JH	Minor updates after comments.

TABLE OF CONTENTS

1. Introduction and Scope

This document gives an overview of the RTTOV v10 fast radiative transfer model (in section 2), limitations in section 3, the differences from RTTOV v9 (in section 4), how to install the RTTOV v10 code on a UNIX/LINUX platform and run it (section 5) and how to apply it to the user's particular application (section 6). The procedure for reporting bugs or learning about known bugs is given in section 7. Finally a frequently asked questions (FAQ) section is provided in section 8. This document relates to version 10 of the RTTOV code and all its sub-versions (10.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 v10 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 v10 code, go to http://research.metoffice.gov.uk/research/interproj/nwpsaf/request_forms/index.html and complete a licence agreement form on-line. You will then be given access to the code via FTP or sent a CD containing the code.

The old RTTOV v7, 8 and 9 codes are still available in FORTRAN 90 but cannot be guaranteed to be upgraded for new instruments and capability. Coefficient files for RTTOV v7, RTTOV v8 and RTTOV v9 will continue to be made available from the NWP SAF web site but note that they will not work with the new RTTOV v10 code. RTTOV v10 is a rewrite of RTTOV v9 adding many more features as documented here.

The RTTOV v10 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://research.metoffice.gov.uk/research/interproj/nwpsaf/rtm/ in pdf format on the RTTOV v10 page. There is also a RTTOV v10 performance report which documents the run times of RTTOV v10 on a few platforms and compares these to the equivalent RTTOV v9 run times.

2. Overview of RTTOV v10

This section gives a brief overview of the RTTOV v10 model. More details can be found in the references given in this section. RTTOV v10 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 v8, 9 and 10 are the latest versions. 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 v10 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 v10 is given in Annex V. Not all parameters have to be supplied as RTTOV can assume default values. RTTOV v10 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 v10 is mainly based on the 91L 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).

The spectral range of the RTTOV v10 model in the infrared is 3-20 microns (500 – 3000 cm^{-1}), governed by the range of the GENLN2 or kCARTA or 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 even in monotonic wavelength order. It is planned to improve this aspect eventually but users so far are reluctant to change the current historical order. 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 each of the RTTOV versions.

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	1.400	0.014	0.296
2	0.01	250.40	151.24	5.81	1.03	1.410	0.054	0.314
3	0.10	292.17	188.94	8.32	1.72	1.900	0.210	0.675
4	0.20	313.63	203.72	8.56	1.93	2.290	0.319	0.993
5	0.50	340.17	200.98	8.52	2.73	2.810	0.654	1.650
6	0.80	342.35	195.20	8.23	3.12	4.120	0.738	2.240
7	1.20	339.36	185.90	8.04	3.36	5.840	0.707	3.010
8	1.60	335.28	179.94	7.91	3.24	7.130	0.653	3.660
9	2.20	327.76	177.24	7.78	2.97	8.710	0.528	4.490
10	2.70	319.36	175.70	7.74	2.90	9.480	0.490	5.080
11	3.50	313.42	175.28	7.70	2.85	10.200	0.677	5.840
12	4.20	309.30	174.27	7.66	2.76	11.300	1.030	6.330
13	5.00	304.91	173.51	7.60	2.72	12.200	1.560	6.710
14	6.95	296.13	168.79	7.54	2.56	12.900	1.870	7.100
15	10.37	292.52	165.71	7.35	2.45	12.700	1.200	7.000
16	14.81	283.88	162.46	7.12	2.40	11.500	0.549	6.240
17	20.40	281.75	161.31	6.87	1.82	10.400	0.358	5.060
18	27.26	282.24	161.97	6.49	1.56	8.920	0.188	4.040
19	35.51	280.24	162.40	6.19	1.31	7.440	0.108	3.170
20	45.29	272.90	164.73	5.91	1.35	6.780	0.054	2.460
21	56.73	265.49	166.31	6.47	1.29	5.580	0.048	1.850
22	69.97	263.98	167.71	11.50	1.15	4.670	0.051	1.360
23	85.18	262.26	158.53	18.20	0.03	4.260	0.025	0.930
24	102.05	261.36	164.30	24.00	0.01	3.400	0.016	0.682
25	122.04	259.53	169.36	50.20	0.01	2.840	0.016	0.567
26	143.84	259.10	169.76	160.00	0.01	2.510	0.016	0.445
27	167.95	261.07	169.49	436.00	0.01	2.190	0.010	0.340
28	194.36	263.46	172.66	1060.00	0.01	1.730	0.011	0.236
29	222.94	266.68	174.80	1900.00	0.01	1.300	0.016	0.165
30	253.71	274.71	180.34	3530.00	0.01	0.865	0.016	0.124
31	286.60	282.36	183.54	5650.00	0.01	0.684	0.015	0.094
32	321.50	290.55	187.08	8360.00	1.22	0.578	0.016	0.077
33	358.28	298.44	188.49	11400.00	1.46	0.484	0.016	0.066
34	396.81	302.01	193.18	14500.00	1.78	0.393	0.015	0.059
35	436.95	303.77	197.14	17600.00	2.34	0.313	0.015	0.055
36	478.54	306.31	200.84	20800.00	2.70	0.259	0.015	0.052
37	521.46	310.12	202.54	23900.00	3.43	0.217	0.015	0.051
38	565.54	314.90	202.22	26800.00	3.89	0.193	0.012	0.050
39	610.60	317.35	189.96	29800.00	6.67	0.174	0.010	0.049
40	656.43	321.49	189.96	32500.00	6.17	0.132	0.009	0.048
41	702.73	327.82	189.96	35300.00	6.72	0.124	0.009	0.047
42	749.12	334.25	189.96	37900.00	8.67	0.117	0.008	0.046
43	795.09	336.94	189.96	40500.00	8.17	0.115	0.008	0.044
44	839.95	339.93	189.96	43000.00	7.74	0.113	0.008	0.042
45	882.80	344.77	189.96	45300.00	7.36	0.110	0.007	0.041
46	922.46	348.39	189.96	47300.00	7.04	0.104	0.006	0.038
47	957.44	349.86	189.96	51300.00	6.79	0.100	0.006	0.035
48	985.88	350.08	189.96	50100.00	6.59	0.100	0.006	0.031
49	1005.43	350.08	189.96	48000.00	6.46	0.096	0.006	0.029
50	1025.00	350.08	189.96	47500.00	6.34	0.094	0.006	0.028
51	1050.00	350.08	189.96	47600.00	6.19	0.094	0.006	0.027

Table 1. Pressure levels adopted for RTTOV v10 51 level coefficients and profile limits within which the transmittance calculations are valid. The default ozone profile is also given in the right hand column.

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 $\boldsymbol{x_0}$.

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

Where 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 (sec. 2.1), but there are options for infrared cloudy and aerosol-affected radiances (secs. 2.5, 2.6) and for cloud and precipitation affected microwave radiances (sec. 2.7).

Platform	RTTOV id	Sat id range
NOAA¶	1	1 to 19
DMSP	2	8 to 18
Meteosat	3	1 to 7
GOES	4	4 to 16
GMS	5	5
FY2	6	2 to 4
TRMM	7	1
ERS	8	1 to 2
EOS	9	1 to 2
METOP	10	2
ENVISAT	11	1
MSG	12	1 to 3
FY1	13	3 to 4
ADEOS	*14*	*1 to 2*
MTSAT	15	1 to 2
CORIOLIS	16	1
JPSS/NPP	17	0
GIFTS	*18*	*1*
Sentinel	19	1
MeghaTropique	20	1
Kalpana	*21*	*1*
Reserved	22	
FY3	23	1
COMS	24	1
METEOR-M	25	1
GOSAT	26	1
CALIPSO	27	1
Reserved	28	
GCOM-W	29	1

¶ Includes TIROS-N

Table 2. Platforms supported by RTTOV as at December 2011. Platforms in italics are not yet supported in the RTTOV v10 distribution but can be requested.

2.1. Simulation of clear air radiances

If *N,* the cloud cover parameter, is set to zero and the liquid water concentration profile vector is set to zero both the infrared and microwave radiances computed are for clear air with the second right hand term of equation 6 being zero. $L^{Clr}(\nu,\theta)$ 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 (now only used for SSU) and those given in Matricardi (2005) or the RTTOV v9 science plan for RTTOV v9 predictors (only used for advanced IR sounders such as AIRS and IASI). The code supports any of these predictor sets with the selection being made according to the coefficient file supplied to the program. 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 v10.

Sensor	RTTOV id	Sensor Chans	RTTOV v10 Chans
HIRS	0	1 to 19	1 to 19
MSU	1	1 to 4	1 to 4
SSU**	2	1 to 3	1 to 3
AMSU-A	3	1 to 15	1 to 15
AMSU-B	4	1 to 5	1 to 5
AVHRR**	5	3b to 5	1 to 3
SSMI	6	1 to 7	1 to 7
*VTPR1***	*7*	*1 to 8*	*1 to 8*
*VTPR2***	*8*	*1 to 8*	*1 to 8*
TMI	9	1 to 9	1 to 9
SSMIS***	10	1 to 24*	1 to 24*
AIRS	11	1 to 2378	1 to 2378
HSB	12	1 to 4	1 to 4
MODIS**	13	1 to 16	1 to 16
ATSR/SLSTR	14	1 to 3/7 to 9	1 to 3
MHS	15	1 to 5	1 to 5
IASI	16	1 to 8461	1 to 8461
AMSR-E/AMSR2	17	1 to 12/1 to 14	1 to 12/1 to 14
Reserved	18		
ATMS	19	1 to 22	1 to 22
MVIRI**	20	1 to 2	1 to 2
SEVIRI**	21	4 to 11	1 to 8
GOES-Imager**	22	1 to 4	1 to 4
GOES-Sounder	23	1 to 18	1 to 18
GMS/MTSAT imager***	24	1 to 3/1 to 4	1 to 3/1 to 4
FY2-VISSR**	25	1 to 2/4	1 to 2/4
FY1-MVISR**	26	1 to 3	1 to 3
CrIS	27	1 to 1305	1 to 1305
VIIRS	29	16 to 22	1 to 7
WINDSAT	30	1 to 16	1 to 16
GIFTS	*31*	*TBD*	*TBD*
SSM-T1	*32*	*1 to 7*	*1 to 7*
SSM-T2	*33*	*1 to 5*	*1 to 5*
SAPHIR	34	1 to 6	1 to 6
MADRAS	35	1 to 9	1 to 9
Spare	36		
*Kalpana Imager***	*37*	*1 to 2*	*1 to 2*
Reserved	38-39		
FY3 MWTS	40	1 to 4	1 to 4
FY3 MWHS	41	1 to 5	1 to 5
FY3 IRAS	42	1 to 20	1 to 20
FY3 MWRI	43	1 to 10	1 to 10
GOES-R ABI**	44	1 to 3	1 to 3
COMS MI**	45	1 to 4	1 to 4
MSUMR	46	1 to 3	1 to 3
Reserved	47		

Calipso IIR	48	1 to 3	1 to 3
ESA MWR	*49*	*1 to 2*	*1 to 2*
Reserved	50-53	-	-

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

Table 3. Instruments supported by RTTOV v10 at Dec 2011. Sensors in italics are not yet supported in the RTTOV v10 distribution but can be requested.

If reflected solar radiation is required to be included in the SWIR channels (i.e. in the range 2000-2760 cm^{-1}) then the logical flag `opts%addsolar` must be set to true and a number of additional profile variables need to be specified which are, solar zenith and azimuth angles `profiles(:)%sunzenangle`, `profiles(:)%sunazangle`, the satellite azimuth angle, `profiles(:)%azangle` specification of fresh or salt water `profiles(:)%skin%watertype` and surface wind and wind fetch in `profiles(:)%s2m%u`, `profiles(:)%s2m%v`, `profiles(:)%s2m%wfetc`. The computation is only performed if the solar zenith angle is less than 84^o. The satellite azimuth angle is the azimuth angle of the direction formed by the projection on the X-Y plane of the vector pointing towards the receiver. The X-Y plane coincides with the mean sea level and the Z-axis points towards the zenith. The X-axis coincides with the direction of the local parallel on the Earth's surface. It is positive if directed eastward, negative if directed westward. The Y-axis coincides with the direction of the local meridian on the Earth's surface. It is positive if directed northward, negative if directed southward. The azimuth angle is counted counter-clockwise from the X-axis. Note that reflected solar radiation can only be included for the SWIR channels of IASI and AIRS at present since regression coefficients are not yet available for other sensors.

2.2. Simple cloud

More complicated treatments of cloud and precipitation are available (see sections 2.5 and 2.7), but the simplest cloud approach is described here. If a black opaque cloud is assumed at a single level, the top of the atmosphere upwelling radiance, $L(v,\theta)$, at a frequency v and viewing angle θ from zenith at the surface, neglecting scattering effects, is written as:

$$L(v,\theta) = (1-N)L^{Clr}(v,\theta) + NL^{Cld}(v,\theta) \qquad (6)$$

where $L^{Clr}(v,\theta)$ and $L^{Cld}(v,\theta)$ are the clear sky and fully cloudy top of atmosphere upwelling radiances and N is the effective fractional cloud cover.

2.3. 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 U). 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` must lie in the range 0.2-0.7 gauss, and certainly should not be set to zero, as it is a divisor in the predictors.

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 v10 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%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.

2.4. Definition of surface emissivity

To compute ε_s over water there are fast surface emissivity routines for both the infrared, ISEM, (Sherlock, 1999) and for the microwave, which has now been upgraded to FASTEM-4 (Liu et al, 2010). Some improvements to FASTEM-4 have been made since the release of RTTOV v10.1 and this alternative model has been named FASTEM-5 in v10.2. The updates made for FASTEM-5 are as follows:

1. A constraint has been added to the reflectance fitting equations, ensuring the same vertically and horizontally polarised reflectance at nadir. (V- and H-pol nadir reflectances have differed significantly in previous FASTEM versions).

2. The foam coverage model used in FASTEM-4 (Tang, 1974) has been found to differ significantly from values derived from microwave measurements, so FASTEM-5 uses the same foam coverage model as FASTEM-3 (Monahan *et al.*, 1986).

These two changes result in the analytic equation used for the large-scale reflectance calculation giving a better fit in the regression than for FASTEM-4.

The FASTEM models all compute a surface emissivity for the channel of interest at the given viewing angle θ. Using FASTEM requires the 10m wind-speed to be provided in the state vector. The version of FASTEM to use (1 to 5) can be set in the `opts%fastem_version` variable. If this is set to a value other 1-5, the FASTEM version will be taken from the instrument coefficient file: this is the default setting and all microwave v10 coefficient files specify FASTEM-4. For FASTEM it is also important to define the polarisation status of each channel which is given in each coefficient file as a series of numbers in the FASTEM section and are defined in Table 4. Different polarisations are defined for cross-track scanners, conical scanners and polarimetric instruments. Users shouldn't need to worry about this unless they want to change the polarisation of a particular microwave channel in which case they should edit the coefficient file and modify the polarisation IDs.

Fastem Pol_ID in coeff file	Definition	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 4. Definition of polarisation status for FASTEM-2/3/4/5.

Over the land and sea-ice surfaces, approximate default values are provided for the surface emissivity in both the infrared (0.98 over land, 0.99 over sea-ice) and microwave (based on a parameterised model, see refs above for details and Table 5) . For more accurate emissivity estimates over land RTTOV v10 also provides access to IR and MW land surface emissivity atlases.

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.

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).

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).

As in previous versions of RTTOV the user has the option of providing their own estimate of surface emissivity to the model (see Table 5 for input options). The user is now able to obtain emissivity values from the appropriate atlas and pass these to the model if desired, in which case `calcemis` should be set to false. Note that for Principal Component calculations (which are only valid over sea surfaces), surface emissivities are calculated within RTTOV using the RTIASI emissivity model, which is a modified version of the Masuda et al (1988) model and the user should ensure `calcemis` is set to true for PC calculations.

Table 5 also shows the tangent linear emissivity output. For IR instruments, no emissivity tangent linear is calculated at all, except in the case of Principal Components calculations. For all instruments, the input `emissivity_tl` should be zero when `calcemis` is true.

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_{RTIASI} for PC-RTTOV	No ∂ε calculated, except for PC-RTTOV where sea ∂ε, computed from ∂u, ∂v about ε_{RTIASI}.
False	$\varepsilon_{atlas/user}$	$\varepsilon_{atlas/user}$ sea=ε_{RTIASI} for PC-RTTOV	No ∂ε calculated, except for PC-RTTOV where sea ∂ε, computed from ∂u, ∂v about ε_{RTIASI}.
		MICROWAVE CHANNELS	
True	0	Land/sea-ice computed from coefs in `prof % skin % fastem(1:5)` sea= $\varepsilon_{FASTEM4}$	Land/sea-ice ∂ε about $\varepsilon_{FASTEM4}$ sea ∂ε, computed from ∂u, ∂v, ∂sst about $\varepsilon_{FASTEM4}$
False	$\varepsilon_{atlas/user}$	$\varepsilon_{atlas/user}$	No ∂ε calculated

Table 5. Input and output values of ε and ∂ε arrays for infrared and microwave channels for forward and gradient surface emissivity routines. The version of FASTEM can be set in the opts%fastem_version variable. If this is set to a value other than 1-5, the version is taken from the coefficient file (FASTEM-4 for v10 coeff files).

2.5. Simulation of cloudy radiances

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:

$$L^{Cld}(\nu,\theta) = \tau_{Cld}(\nu,\theta)\, B(\nu,T_{Cld}) + \int_{\tau_{Cld}}^{1} B(\nu,T)\, d\tau \,, \qquad (7)$$

where $\tau_{Cld}(\nu,\theta)$ is the cloud top to space transmittance and T_{Cld} the cloud top temperature specified by the cloud top pressure in the input state vector. The emissivity of the cloud top is assumed to be unity which is a tolerable assumption for optically thick water cloud at infrared radiances but not valid for optically thin cloud and all cloud at microwave frequencies. For partially cloud filled fields of view, equation 6 is used to combine the clear sky radiance (equation 5) and cloudy radiance (equation 7) using the fractional cloud cover *N* provided as input.

This simple cloud calculation (equation 6) can be used for infrared channels and 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 2.7. 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.

Invoking the multiple scattering scheme within RTTOV requires additional inputs to RTTOV v10 as detailed in Table 6. Note that the cloud profiles are defined on layers: layer *n* lies between levels *n* and *n+1*. If the user only wants to compute clear sky radiances or simple cloudy radiances as defined above the cloud and aerosol profile variables/flags need to be set to zero/.false. 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 7. 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 this is an enhancement over RTTOV v9). In addition to the concentration array the fractional cloud cover array also must be provided `profiles(:)%cfrac(i,j)` where 0 is no cloud and 1 is overcast at layer *j*. Note that although `cfrac` is a two-dimensional array, it must contain ***at most one*** non-zero value per layer. This specifies the combined cloud fraction for all cloud types present in the layer. Example profiles are given in Tables 8 and 9. As of RTTOV v10.2 there is no restriction on the number of layers which may contain non-zero cloud amount.

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 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 restrictions on the values in `cfrac(:,:)` do not apply when running the direct model alone.

Options	**Set logical flags and fill profile arrays**	**Define options to convert IWC to effective diameter**	**Define ice crystal shape**
Cloudy simulation	*opts%addclouds=.true.* *profiles(:)%cloud(i,j)*=layer mean liquid or ice water content in units of $g.m^{-3}$. *profiles(:)%cfrac(i,j)*= fractional cloud cover for each layer (0-1) where *i* is the index of the cloud type (see Table 7) and *j* is the layer index.	Set *profiles(:)%idg* 1 = Ou and Liou (1995) 2 = Wyser (1998) 3 = Boudala et al (2002) 4 = McFarquar et al (2003) Optionally set *profiles(:)%icede(j)* non-zero to specify the ice effective diameter in microns instead of using a parameterisation where *j* is the layer index.	Set *profiles(:)%ish* 1 = Hexagonal 2 = Aggregates
	Set logical flags and fill profile arrays	**Define climatological profile**	
Aerosol simulation	*opts%addaerosl=.true.* *profiles(:)%aerosols(i,j)*=layer mean number density in units of cm^{-3} where *i* is the index of the aerosol component (see Table 11) and *j* is the layer index.	0 = User defined profile Profiles available in file *prof_aerosl_cl.dat* to read into *profiles(:)%aerosols(i,j)* 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	

Table 6. Inputs required for cloud and aerosol options.

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 7. Cloud types available in RTTOV v10.

STCO	STMA	CUCC	CUCP	CUMA	CIRR
0	0	0	0	0	0.026
0	0	0	0	0	0
0	0	0	0	0	0
0.14	0	0.26	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0.28	0	0	0	0	0
0	0	0	0	0	0

Table 8. An example of a cloud liquid/ice water content input profile in g.m^{-3} for some atmospheric layers.

STCO	STMA	CUCC	CUCP	CUMA	CIRR
0	0	0	0	0	0.8
0	0	0	0	0	0
0	0	0	0	0	0
0.5	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0.3	0	0	0	0	0
0	0	0	0	0	0

Table 9. An example of a cloud fractional coverage input profile for some atmospheric layers. ***NB at most one*** *non-zero value may be specified per layer. In practice it does not matter with which cloud type it is associated.*

For water clouds optical parameters are available for five size distributions corresponding to five different cloud types whereas for ice clouds the optical parameters are parameterized as a function of the effective diameter of the size distribution. Consequently, for ice clouds the user must choose which assumption to use to parameterise the effective diameter and must also specify which shape is to be used for the ice crystals since optical parameters are available for hexagonal ice crystals and ice aggregates as defined in Table 6, `profiles(:)%idg`.

As of RTTOV v10.2 a new profile variable, `profiles(:)%icede(:)`, is available which may optionally be used to specify the ice particle effective diameter. 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 non-zero. If `icede(:)` is zero for the layer, the parameterisation specified by `profiles(:)%idg` is used.

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%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, as noted above, by default `cldstr_threshold` is set to a negative number and this is the recommended setting, particularly 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 IR radiances in addition to filling the input profile structure with cloud water concentration and fractional cover for each cloud type (`cloud(:,:)` in g.m^{-3}and `cfrac(:,:)` from 0-1) the user must ensure the IR scattering coefficient file is linked to the main directory in the same way as the other coefficient files. The naming convention for the file is: *sccldcoef_meteosat_5_mviri.dat* – optical parameters for water and ice cloud types where Meteosat-5 is the sensor in this case.

2.6. Simulation of aerosol affected radiances

Using the new multiple scattering code it is also possible to simulate the effects of aerosols at infrared wavelengths. The methodology is described in the RTTOV v9 science and validation plan. Again additional inputs prescribed in Table 6 are required for aerosol simulations, and note that the input aerosol profiles are defined on layers rather than levels as was the case in RTTOV v9. The mixing of the various aerosol components can be defined by the user or climatological profiles with predefined mixing can be supplied. The input profile `profiles(:)%aerosols(i,j)` where the first index is the aerosol component (currently 1-11 as defined in Table 11) and the second index is the layer number. 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 of an input profile for a few layers is given in Table 10. Alternatively if a climatological profile is chosen (see Table 6 for options and RTTOV v9 science plan for more details on profiles) the user can input the climatological profile file *data/prof_aerosl_cl.dat* and the layer mean number densities can be scaled by a factor if required. This was generated for a specific input latitude of zero, elevation of zero, surface level (2) and scale factor (1.0) using the standard RTTOV 101 pressure levels with T and q profiles taken from the RTTOV v9 file prof_101lev.dat. The aerosol profiles supplied are all on 100 layers. If the user requires these profiles to be on a different set of layers there is a program *aer_clim_prof* provided in *src/other* which will write aerosol climatological profiles on user defined levels. The use of this program is detailed in Annex U.

INSO	WASO	SOOT	SSAM	SSCM	MINM	MIAM	MICM	MITR	SUSO	VOLA
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
1350	0.9	0	0	0	0	0	0	0	0	0
1600	0.11	0	0	0	0	0	0	0	0	0
1800	0.12	0	0	0	0	0	0	0	0	0
2000	0.13	0	0	0	0	0	0	0	0	0
2400	0.14	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0

Table 10. An example of part of the input profile for aerosol components for a few layers units are number density cm^{-3}.

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

Table 11. Aerosol components available in RTTOV v10.

It is important to note that due to the way in which the aerosol-affected radiances are calculated, the output "clear-sky" radiances in the *radiance_type* structure (e.g. `radiance%clear`, `radiance%bt_clear`) include the effects of aerosol when these calculations are performed.

To compute aerosol-affected IR radiances, in addition to filling the input profile structure in aerosol number density for each aerosol type (`aerosols(:,:)` in cm^{-3}), the user must ensure the IR aerosol scattering coefficient file is linked to the main directory. The naming convention for the file is: *scaercoef_meteosat_5_mviri.dat* – optical parameters for aerosol types where MVIRI on Meteosat-5 is the sensor in this case.

2.7. 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 2.5 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 function 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 14 and section 6.3) but additional information is required, principally hydrometeor profiles, supplied in `profile_cloud_type` and listed in Table 12. 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.

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.
cfrac	***Optional:*** *if 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 (kg/(m^2)/s)
sp(:)	nlevels of solid precipitation flux (kg/(m^2)/s)

Table 12. RTTOV-SCATT profile variables for `profile_cloud_type`

RTTOV-SCATT uses a slightly different level definition (compared to RTTOV v9) for the cloudy/rainy column, in which 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`, but the half level pressures need to be supplied in `profile_cloud_type`. Figure 1 shows the arrangement of full and half levels.

An example of how to use RTTOV-SCATT in forward mode is `example_rttovscatt.F90` provided in the `src/mw_scatt` directory. This reads an atmospheric profile from a data file and prints the simulated brightness temperatures to screen. The test programs `rttovscatt_test.F90` (top level, driven by `rttovscatt_test.sh`) and `rttovscatt_test_one.F90` contain examples of how to use the tangent-linear, adjoint and K functionality.

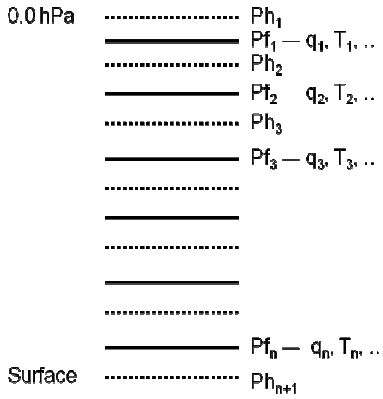

Figure 1. RTTOV-SCATT full and half levels, showing half and full level pressure (Ph,Pf) and examples of the variables specified on full levels (e.g. q, T, but also cloud and hydrometeors)

There are two logical input variables for RTTOV-SCATT, both of which are fortran `optional`, and hence do not need to be supplied. First, by default, the new cloud overlap (Geer et al., 2009a) is used. However, if lnewcld = .false., the original cloud overlap (the default up to RTTOV v9) will be used. Second, if lusercfrac = .true., the user can supply their own effective cloud fraction (see Geer et al., 2009b). By default, the effective cloud fraction will be calculated internally in RTTOV-SCATT.

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 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. To build the necessary executable for coefficient generation, RTTOV must be compiled with the `mw_scatt_coef` build target (see section 5.2). 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.

2.8. Simulation of IASI and AIRS radiances using principal components scores

A principal component (PC) based version of RTTOV is available for the simulation of the full IASI and AIRS radiance spectrum as PCs. 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 v10, a logical flag `opts%addpc` is set. A number of restrictions apply to the PC calculations: they may only be performed for AIRS and IASI over ocean surfaces for a clear atmosphere. 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. Surface emissivities are always calculated within PC-RTTOV and as such `calcemis(:)` should always be set to true.

As the computation of the principal component scores coefficients has been carried out using RTTOV v9 predictors these allow for the variation of CO_2, N_2O, CO and CH_4. However, in the LBL computations the concentration of these species is fixed and consequently the same fixed values must be used in the RTTOV computations. 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 RTTOV 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 is mandatory with the inclusion of the effects due to the presence of atmospheric refraction. Therefore refraction is accounted for regardless of the setting of `opts%addrefrac`.

The user can choose how many predictors are to be used for the prediction of the principal component scores and how many principal components are needed. The number of predictors is specified via `opts%ipcreg`; the following choices are available:

1) IASI: 300, 400, 500, or 600 predictors – set `opts%ipcreg` = 1, 2, 3 or 4, respectively
2) AIRS: 200, 300, or 400 predictors – set `opts%ipcreg` = 1, 2 or 3, respectively

Note that the number of channels per profile is defined by `opts%ipcreg` as described above. This also defines the input channel list to RTTOV (in the `chanprof` structure). The appropriate channel list can be obtained from the PC coefficient structure: `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 the user's code. An alternative way of obtaining the channel list is via the subroutine `rttov_get_pc_predictindex` (Annex O). If the input channel list does not match the predictor channel list, RTTOV will report an error.

The number of principal components can vary from 1 to 400 irrespective of the instrument. This is defined when calling the `rttov_alloc_pccomp` subroutine (see Annex I) to allocate the `pccomp` structure (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. The user can vary these parameters and consequently change the computational efficiency and the accuracy of the model.

The use of the PC-based version of RTTOV requires additionally the coefficient files `pccoef_metop_2_iasi.dat` and `pccoef_eos_2_airs.dat`: these coefficient files contain the regression coefficients used to predict the principal component scores and up to 400 eigenvectors to reconstruct the radiances. For the IASI instrument, regression coefficients are available based on 300, 400, 500 and 600 predictors whereas for the AIRS instrument regression coefficients are available based on 200, 300 and 400 predictors. Since the computation of the polychromatic predictors require the mandatory use of the same emissivities and trace gas profiles used in the LBLRTM computations on which the principal component based version of RTTOV is trained, the new coefficient files also store the fixed profiles of CO_2, N_2O, CO and CH_4 and the regression coefficients used for the emissivity model utilized in the LBLRTM computations.

The computed PC scores are stored in the `pccomp` structure (see Annex V). It is possible to reconstruct radiances from the PC scores by setting `opts%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 I) and the reconstructed channel list `channels_rec(:)` may be supplied to `rttov_direct` (or to the TL, AD or K model; Annex P,Q,R,S). The reconstructed radiances are also stored in the `pccomp` structure. If the optional `channels_rec(:)` input array is not supplied, radiances for all channels will be computed.

As of RTTOV v10.2, PC-RTTOV can be run via the parallel RTTOV routines.

3. Limitations of RTTOV v10

There are a number of scientific limitations of RTTOV v10 the user should be aware of. The main ones are listed here:

- RTTOV v10 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 v10 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 v10 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 3 microns can be simulated with RTTOV v10.
- 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 available for all sensors where this is a problem resulting in much lower biases.
- Principal Component computations can only be performed for clear-sky profiles over sea surfaces.

4. Changes from RTTOV v9

The main differences between the RTTOV v9 code and RTTOV v10 not documented above are listed here.

Treatment of the model-top
In RTTOV v9 the profile provided by the user did not include the values of the variables for the top level of the profile. The profile top pressure was fixed at 0.005 hPa in the code, and in the calculation of the predictors, the top variables were given the same values as the level below, the top-most level of the input profile. In RTTOV v10, by contrast, the profile top is treated like any other level, the pressure and the variables at the model-top being supplied by the user as part of the input profile. This simplifies the code and allows the user to provide a more representative top layer for the profile. It also improves the simulations for high peaking channels.

The user may wish to check for back-compatibility with RTTOV v9 for comparison or testing purposes. Exact duplication will be difficult to achieve in all circumstances, but there are steps the user should take if RTTOV v10 results are to be brought as close as possible to those for RTTOV v9. For this, the user should ensure that the following conditions are satisfied:

In the reference and limits profiles in the coefficient file
1- top pressure is 0.005 hPa
2- the rest of the pressure grid agrees with RTTOV v9
3- top temperature and abundance in the reference profile agrees (in ppmv) with level below
In the user input profile
4- top pressure is 0.005 hPa
5- the rest of the pressure grid agrees with RTTOV v9
6- top temperature and abundance passed to predictor calculation agrees (in ppmv) with level below

Unless conditions 4 and 5 are satisfied, the interpolator will be called, and the essential problem is that, when unit conversions or the interpolator intervene between the profile input and the calculation of the predictors, it is difficult for RTTOV v10 to ensure that condition 6 is satisfied, leading to differences in the predicted optical depths.

Note that the old RTTOV v9 (or 7 and 8) coefficient files will not work with RTTOV v10, because the header structure is different. Also, predictors involving the top layer may have been assigned different values in the generation of coefficients for RTTOV v9 and RTTOV v10, leading to a difference in the coefficients.

Some users may wish to use RTTOV v9 input profiles, particularly while undertaking back-compatibility checks. In that case, the model-top variables will not, initially, form part of the profile and must be added to the profile arrays by the user before RTTOV is called. However, if conditions 4-6 are followed, this procedure may be automated.

Changes to the interface
All the logical flags are now in one common structure `rttov_options` or `opts%` as defined in Annex V. The `rttov_coefs` structure contains all coefficients (i.e. optical depth, IR scattering, and PC coefficients). Similarly the profile variables are all in the `profile_type` structure defined in Annex V. All profiles in an RTTOV call should have the same gas combination. The `errorstatus` argument to the core routines is now a scalar rather than an array. The lists of channels and corresponding profiles for which to compute radiances have been combined into a single `chanprof` structure (see Annex V). The `transmittance` structure now contains only the total surface-to-satellite transmittance, and the transmittance from each level to the satellite (see Annex V).

New Code
The code was rewritten to be more efficient and clearer. The test suite for testing of the code has been rewritten to be more flexible.

Other important changes from RTTOV v9 are listed below so the user is aware of them. Most of these are documented elsewhere in this document.

- IR and MW surface emissivity atlases included for improved simulations of window channels over land
- Improved MW emissivities over ocean using FASTEM-4 and, since RTTOV v10.2, FASTEM-5.
- Option of computation of PCs instead of radiances for IASI and AIRS
- Any combination of cloud types now allowed per layer
- Inclusion of explicit treatment of Zeeman effect for AMSU-A and SSMIS channels

- A new set of coefficients which only work for RTTOV v10 created
- Updated IR sensor coefficients based on latest spectroscopy (LBLRTMv11)
- Updated SSU coefficients based on newer spectroscopy
- The code has been organised into separate functional units. Compilation of the code now results in several libraries being built which the user should link to, rather than a single library.

5. FORTRAN-90 UNIX/LINUX installation instructions

RTTOV v10 is designed for UNIX/Linux systems. The software is now successfully tested on Intel, IBM, NEC, Sun 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 v10:

- UNIX or Linux operating system
- Fortran 90 compiler
- make utilities
- gzip and gunzip
- about 100 MBytes of free disk space is the minimum required (3.5 GBytes if you require AIRS and IASI coefficient files)
- Typically 10 Mbytes of memory are required to run the code.
- In order to use the emissivity atlases, the NetCDF library (v3.6 or higher) is required.

Some basic information on installing the RTTOV v10 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 from ECMWF Data Services. The file name should be *rttov10.tar.gz* and be copied to your 'top' RTTOV directory (e.g. ~user/rttov10) from which subdirectories will be created.

RTTOV v10 will not work with older versions of some compilers. The following list gives the versions of several common compilers known to be compatible:

- gfortran – v4.4.0 and later (the test suite does not work with earlier versions; OpenMP requires v4.4.5 or later)
- g95 – v0.93
- ifort – v9.1 and later
- NAG – v5.1 and later (note v5.1 has separate compiler flags to v5.2 and later)
- pgf90 – v8 and later (the test suite does not work with earlier versions; OpenMP only tested successfully for v11.7).
- IBM – xlf95 v12.1 and v13.1
- NEC – FORTRAN90/SX Version 2.0

5.1 Unpacking the code

This is achieved using the command:

```
$ tar -zxvf rttov102.tar.gz
```

The following subdirectories are created:

`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 (see below)
`src/`	The RTTOV source code
`rtcoef_rttov10/`	RTTOV v10 coefficient files
`rttov_test/`	test scripts, input profiles for tests, and reference output for tests

Most coefficient files are based on RTTOV v7 predictors. SSU coefficients use RTTOV v8 predictors. Files based on RTTOV v9 predictors are available for AIRS and IASI. MW files are configured for FASTEM-4. Zeeman files are included for AMSU-A and SSMI/S. PC coefficients are available for IASI and AIRS. IR scattering coefficients are available for most IR sensors. MW scattering coefficients are available for a range of MW instruments.

To reduce the size of the source distribution, the emissivity data, the IR and MW scattering coefficient files, the PC coefficient files, and the AIRS and IASI coefficient files are not included. These may be downloaded from the RTTOV v10 website. The header format for coefficient files has changed in RTTOV v10 so that older coefficient files cannot be used with RTTOV v10 without modification. The supplied coefficient files are the recommended ones for use with RTTOV v10. However, a program is available to convert v7/8/9 coefficient files to be compatible with RTTOV v10 (see Annex A).

The `rtcoef_rttov10/` directory contains sub-directories for each kind of coefficient file:

`rttov7pred51L/`	v7 predictor files on 51 levels (most optical depth predictor coefficient files)
`rttov7pred101L/`	v7 predictor files on 101 levels (for AIRS, IASI and CrIS files)
`rttov8pred44L/`	v8 predictor files on 44 levels (for SSU only)
`rttov9pred101L/`	v9 predictor files on 101 levels (for the v9 AIRS and IASI files)
`cldaer/`	IR cloud and aerosol scattering coefficient files
`mietable/`	MW scattering coefficient files
`pc/`	Principal Components coefficient files

For the purposes of running the test suite, the user should ensure coefficient files are placed in the appropriate directories.

5.2 Compiling the code

The user may compile RTTOV v10 simply by navigating to the `src/` directory of the RTTOV distribution and typing:

```
$ make
```

By default this builds just the core RTTOV libraries and the test executables using the gfortran compiler. Note a separate library is created for each folder in the `src/` directory, and the user should link all required libraries (at the very least *librttov10.2.0_main.a* and *librttov10.2.0_coef_io.a*) when using RTTOV in their own systems.

Users would usually wish to compile for a specific architecture: the directory `build/arch/` contains a list of architectures which have been tested by the project team. The user should choose the one which appears most appropriate for their target machine. The user should navigate to the `src/` directory and type:

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

This builds RTTOV for the `myarch` architecture: `myarch` must exist in the `build/arch/` directory. By default the build process creates a number of subdirectories containing object code (`obj/`), modules (`mod/`), interfaces (`include/`), executables (`bin/`) and libraries (`lib/`) in the RTTOV distribution directory. If the optional `INSTALLDIR` is specified, these subdirectories are placed in `mydir/` (relative to the top-level RTTOV distribution directory) which can be useful when compiling RTTOV with different flags or different compilers on the same machine. Note that `mydir/` will be created within the RTTOV distribution directory. However, once the code has been compiled and tested, the user is free to move it where he likes.

If `target` is not specified, only the core RTTOV libraries and the test executables are built. If the `all` target is specified, the full RTTOV code is built including RTTOV_SCATT and the new land surface emissivity atlases (NB see section on external libraries below). If the `clean` target is specified, all object files, modules, interfaces, binaries and libraries generated by the Makefile are deleted.

It is also possible to build the specific parts of the RTTOV code the user is interested in. The code has been organised in a modular fashion within the `src/` directory. To build just the RTTOV_SCATT code, for example, type:

```
$ make ARCH=ifort INSTALLDIR=install/ifort mw_scatt
```

This builds just the code in the `src/mw_scatt/` directory and its dependencies. Other targets exist, for example:

```
$ make ARCH=ifort INSTALLDIR=install/ifort mw_scatt/lib

$ make ARCH=ifort INSTALLDIR=install/ifort mw_scatt/clean
```

It may occur that the Makefiles used by RTTOV need to be recreated or updated. This will happen, for example, if there is a change in the dependencies within the RTTOV source code, or if source files are added or removed from the `src/` directory. In this case the user should navigate to the `src/` directory and type:

```
$ ../build/Makefile.PL
$ make ARCH=myarch [INSTALLDIR=mydir] clean
```

The `Makefile.PL` script analyses the dependencies between the source files in `src/` and automatically generates the necessary Makefiles. The code can then be rebuilt as described above.

To make use of the parallel routines, RTTOV v10 must be compiled with OpenMP. This is typically a case of supplying a suitable flag to an appropriate compiler. There are compiler flag files in `build/arch/` for compiling with OpenMP support with gfortran, pgf90 and ifort.

IMPORTANT NOTE: Using external libraries

Building the emissivity atlases requires the NetCDF library (v3.6 or later). To build the code with any `target` which includes the emissivity atlases (such as `all` or `emis_atlas`), the user must first edit the file `build/Makefile.local` to point to their NetCDF installation.

It is possible to use other external libraries as well, for example DrHook. This can be achieved by editing `build/Makefile.local` to include the flags required by these libraries. An example for DrHook is included in the file (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 user wants to run with DrHook enabled).

The user can specify their own `Makefile.local` on the command line as follows:

```
$ make ARCH=myarch [INSTALLDIR=mydir] MAKEFILE_LOCAL=path_to_makefile_local all
```

As an aside, note that if the emissivity atlases are not required at all, the user may safely delete the `src/emis_atlas` directory. They should then execute:

```
$ ../build/Makefile.PL
```

from the `src/` directory to update the Makefiles. All build targets will then work without requiring changes to the file `build/Makefile.local`.

Creating an architecture configuration file

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

`FC` : the name of the user's Fortran 90 compiler.
`FC77` : the name of the user's Fortran 77 compiler; this might be the Fortran 90 compiler, possibly with some special options.
`LDFLAGS_ARCH` : specific flags to pass to the linker.
`FFLAGS_ARCH` : specific flags for the Fortran compiler.

AR : the command to create a library from object files.

NB The Fortran 77 compiler is used to compile specific source files (src/main/lapack.f). However, the RTTOV v10 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 units; defining FFLAGS_ARCH_a will force the build system to compile unit a.F90 with these specific flags.

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

5.3 Running the test code

Users 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.

Core RTTOV testing

NB Users compiling with the Intel Fortran compiler on Linux may need to execute the following command before all tests will run correctly:

```
$ ulimit -s unlimited
```

The simplest way to test the installation of RTTOV is to type:

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

This will run through a series of tests of the core RTTOV code (note that the tests can take several minutes to run). If the INSTALLDIR parameter was supplied to the Makefile, the BIN parameter must be specified here giving the location of the directory containing binary executables relative to the top-level RTTOV distribution directory (e.g. if the user specified INSTALLDIR=install/gfortran when building RTTOV then he should use BIN=install/gfortran/bin). 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 machine rounding errors: these will be reported as differences, but are not cause for concern.

A full description of the RTTOV v10 test suite may be found in docs/rttov-test-v10.2.pdf (under the top-level RTTOV directory). A brief overview is given here.

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/01,avhrr/01 DIRECT=1
```

The ARCH and BIN parameters have been 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 all aspects of RTTOV

to be tested. The full list of parameters may be listed by typing (note ARCH must either be supplied on the command line or be defined as an environment variable):

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

The `rttov_test/` directory contains the following shell scripts which use `rttov_test.pl` to run particular types of tests for a range of instruments:

`test_fwd.sh`	tests the forward model for a wide range of instruments
`test_rttov10.sh`	tests the full code (direct/TL/AD/K) for a range of instruments except AIRS/IASI
`test_rttov10_hires.sh`	tests the full code for AIRS and IASI
`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_cpu.sh`	for performance testing (this test has no reference data)

The `test_core.sh` script simply runs these shell scripts one by one. However, the user can run the individual test scripts by supplying the `ARCH` and `BIN` parameters as described above. For example, to test the forward model:

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

To run all of the tests, the necessary coefficient files must be placed in the `rtcoef_rttov10/` directory. This includes, for example, the AIRS and IASI optical depth coefficient files and the PC coefficient files. These larger files are not included with the distribution and may be downloaded from the RTTOV v10 web page.

Note that a number of the test scripts supply the COEF_EXTRACT=1 argument which causes coefficients to be extracted to the `coefs.1.myarch/` directory for efficiency. Repeated runs of the tests will be faster because these extracted coefficients will be used. However, if the user makes any changes in the `rtcoef_rttov10/` directory, then `coefs.1.myarch/` should be deleted to ensure the updated files are used by the tests.

RTTOV_SCATT testing and example code

The RTTOV_SCATT code is not compiled by default, so a target which includes this code (e.g. `all` or `mw_scatt`) must have been specified when building RTTOV. The `test_rttovscatt.sh` shell script may be used to verify the RTTOV_SCATT code. The user 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_rttov10/rttov7pred/` and `rtcoef_rttov10/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.rttov10_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/`. The user 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 is not compiled by default, so a target which includes this code (e.g. `all` or `emis_atlas`) must have been specified when building RTTOV. 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. The user 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_rttov10/rttov7pred/`), 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). 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 script also writes 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.

Running a simple example of code calling RTTOV v10

A simple example of running the forward model is given in `src/test/example_fwd.F90`. This code may be used as a starting point for the user's own applications. The program may be run using the `run_example_fwd.sh` script. The user may need to edit the first few lines of this script to specify the location of the coefficient files (by default assumed to be in `rtcoef_rttov10/`). The script is run by typing:

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

Test reference output is in `test_example_fwd.2/`. Input files for the script are in the `test_example_fwd.1/` directory, and this is also where the test output is written. The output consists of a file named `output_example_fwd.dat.myarch`, and a diff file named `diff_example_fwd.myarch` showing the differences between the test output and the reference output. The diff file should typically have zero size.

Running an example of code calling PC-RTTOV

An example of running the Principal Components forward model is given in `src/test/example_pc_fwd.F90`. This code may be used as a starting point for the user's own applications. The program may be run using the `run_example_pc_fwd.sh` script. The user may need to edit the first few lines of this script to specify the location of the coefficient files (by default assumed to be in `rtcoef_rttov10/v7pred/` and `rtcoef_rttov10/pc/`). The script is run by typing:

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

Test reference output is in `test_example_pc_fwd.2/`. Test output is written to the `test_example_pc_fwd.1/` directory and consists of files `output_example_pc_fwd_XXXX.dat.myarch`, and diff files named `diff_example_pc_fwd_XXXX.myarch` (where XXXX is 'airs' or 'iasi') showing the differences between the test output and the reference output. The diff files should typically have zero size, or otherwise should show only a small number of differences in the least significant digits.

6. Running RTTOV v10 for your applications

To run RTTOV v10 for a user's application, the test program **example_fwd.F90** can be used as a guide or template. 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 to the users. For users with HP compilers it may be necessary to convert the *.F90* file extensions to *.f90* for all the routines. Use the modules **rttov_types** and **rttov_const** in your program for the definition of derived types and constants (see Annexes V and W), and module **parkind1** for the definition of standard RTTOV integer, real and logical kinds. 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: previously these flags were contained in the **rttov_const** module. It is also important to allocate the various input and output arrays for **rttov_direct** to the correct dimensions (see **example_fwd** or **rttov_test** for examples). Figure 2 gives a process diagram of what routines to call when running RTTOV v10. Annex X gives an example program with comments to guide the user and this source code is provided in the tar file.

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 v10. 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 previous versions of RTTOV will notice that some of the inputs to RTTOV have been consolidated into common structures in RTTOV v10. 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 P.

```
call rttov_direct(errorstatus, chanprof, opts, profiles, coefs, calcemis,
emissivity, emissivity_out, transmission, radiancedata, traj, pccomp,
channels_rec)
```

6.1. Set RTTOV options

The user must first declare the values of **opts** of the *rttov_options* derived type. This structure contains a list of flags with which the user can configure various aspects of RTTOV. These include:

- addclouds set to true to compute cloudy radiances (default: false)
- addaerosl set to true to compute aerosol-affected radiances (default: false)
- addpc set to true to enable Principal Components calculations (default: false)
- addinterp set to true to use RTTOV profile interpolator (default: false)
- addsolar set to true to include solar radiation in near infrared channels (default: false)

The full structure is described in Annex V.

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

6.2. 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%addclouds** or **opts%addaerosl** set to true).
- **optp** : structure containing optical properties (only used if **opts%addclouds** or **opts%addaerosl** set to true).
- **coef_pccomp** : structure containing coefficients for the Principal Components calculations (only used if **opts%addpc** set to true).

The user should allocate the **coefs** structure for the desired number of instruments required to run inside the program and then call **rttov_setup** (see Annex C) for each instrument to load the appropriate coefficient files. This subroutine

calls **rttov_errorhandling** followed by **rttov_read_coefs** and **rttov_init_coefs** (Annex D and E). These latter two routines prepare the necessary members of **coefs** according the flags in **opts**. Alternatively, you may call **rttov_read_coefs** and **rttov_init_coefs** instead of **rttov_setup** as these routines provide more control over the reading of the coefficient files: this method is recommended if you supply the coefficient filenames. If you are supplying the instrument ID triplet, **rttov_setup** is the recommended routine.

If fast performance is required for reading the coefficient files, it is better to access binary 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 binary files. It can also produce ASCII or binary coefficient files for a subset of channels which can be useful for AIRS and IASI. 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 10-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 will give an error message if an incompatible binary coefficient file is being read, but this may not be fully failsafe.

The coefficient file can define the surface emissivity model RTTOV uses. The infrared ocean surface emissivity model ISEM is unchanged in RTTOV v10 for normal radiance calculations and is invoked by setting **calcemis** to true. For microwave instruments, as of RTTOV v10.2 it is possible to select the version of the FASTEM model by setting the **opts%fastem_version** variable to a value between 1 and 5. If it has a value outside this range, the version number specified in the "FASTEM" section of the coefficient file will be used. This latter is the default, and all v10 coefficient files specify FASTEM-4 by default. As for ISEM, FASTEM is invoked by setting **calcemis** to true.

The coefficient file also defines the variable trace gases allowed in the input profile. 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. The options for the different versions of the coefficient files, all of which can be read by the RTTOV v10 code, are given in Table 13. (Note that RTTOV v8 predictors may also be read, but are not recommended for use other than with SSU).

	RTTOV v7 predictors RTTOV v10 file format		RTTOV v9 predictors RTTOV v10 file format	
Gas	Profile	Coeffs	Profile	Coeffs
Mixed	Y	Y	Y	Y
H_2O	Y	Y	Y	Y
O_3	O	O	O	O
CO_2	N	N	O	O
N_2O	N	N	O	O
CO	N	N	O	O
CH_4	N	N	O	O

Y=Input mandatory

O=Input optional depending on options flag and contents of coeff file

N=No input possible

Table 13. Coefficient file options for gaseous absorption in RTTOV v10.

If the coefficient file being used contains coefficients for a particular gas (e.g. **coefs(id)%coef%nozone** > 0) then a profile may be supplied for that gas and the relevant flag in **opts** (e.g. **opts%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 all the microwave coefficients do not include ozone as an active gas.

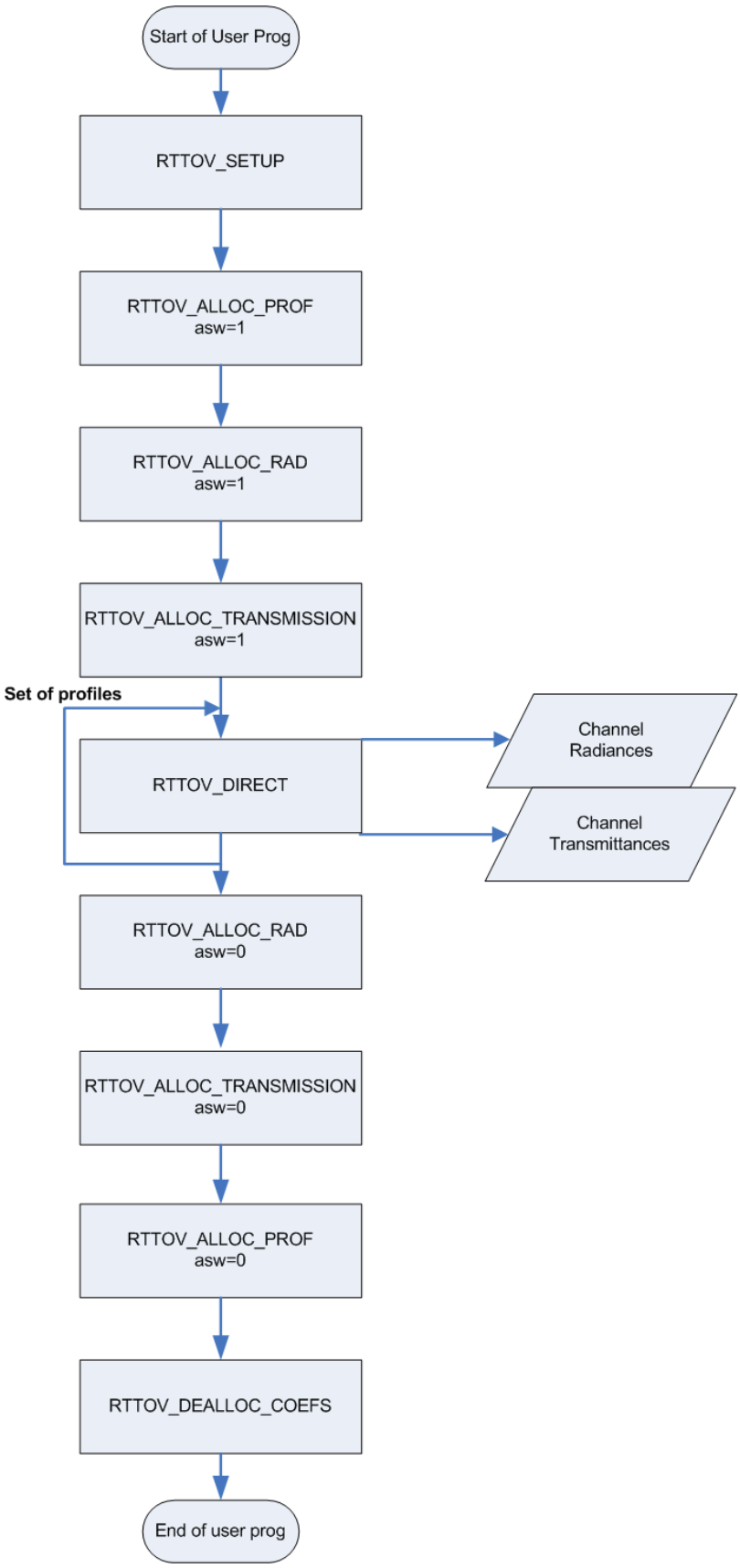

Figure 2. Process diagram of user program calling RTTOV v10 forward model.

6.3. Set up input profiles for RTTOV v10

Allocate a **profiles** array (derived type *profile_type*) with size equal to the number of profiles one wishes RTTOV to process in each call. The **rttov_alloc_prof** subroutine (see Annex F) 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. The members of the array **profiles** as listed in Table 14 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 14 also indicates which profile variables are treated as constants for the tangent linear, and which are active in the TL calculation.

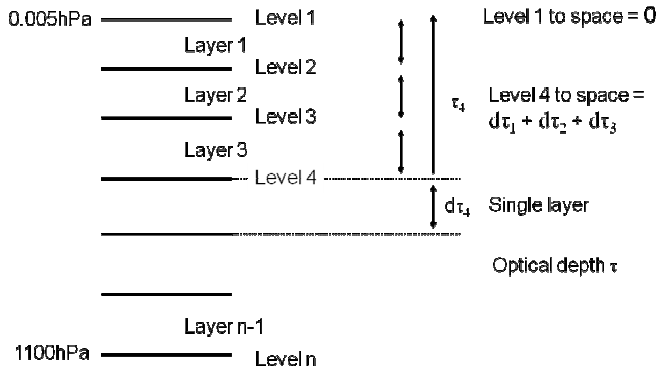

Figure 3. Internal RTTOV coefficient levels and optical depth computations.

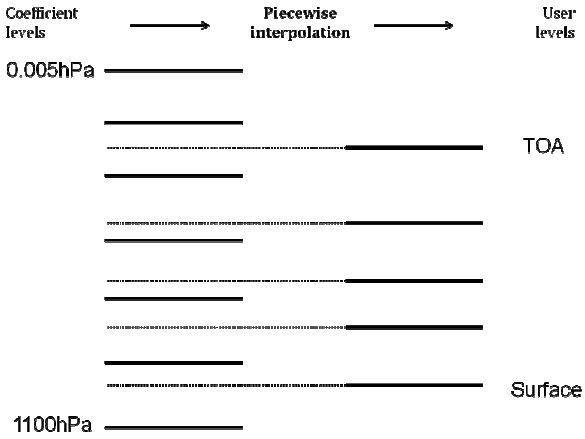

Figure 4. Interpolation to and from to user levels.

The user's input profile is normally on different pressure levels from that required internally by RTTOV (see Figures 3 and 4). As in RTTOV v9, RTTOV v10 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%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 one call to RTTOV v10 the number of levels must remain the same.

The example program in Annex X is a useful guide on how to set up the profile arrays. If one uses a coefficient file with trace gas coefficients (i.e. AIRS and IASI) and doesn't have the trace gas profile concentrations then the **opts%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%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 Annex W). This means, for example, that all input gas profiles (water vapour, ozone, etc) must have values greater than 1E-10 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.

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%apply_reg_limits** is set to false, RTTOV v10 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%verbose_checkinput_warnings** to false), but RTTOV v10 will nevertheless perform the full radiative transfer calculation, using the profile values outside the regression limits. If **opts%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 will still be 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.

There are several points which need to be considered if the internal profile interpolation is used (i.e. the input profile is on different levels to the coefficient file). Points to note are:

i. 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.
ii. The user profile lowest level should be equal or below the surface pressure
iii. 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.
iv. If the interpolation is used, **profiles_tl%p** can be non-zero (and for sigma levels should be). In this case **opts%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 also applies to _ad and _k codes.

Input profile arrays	Description	Units	Mandatory?	When mandatory	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) % p(:)	Pressure levels	hPa	Y		Y if *opts % lgradp* true
profiles(1) % t(:)	Temperatures on levels	K	Y		Y
profiles(i) % q(:)	Water vapour conc on levels	ppmv	Y		Y
profiles(i) % o3(:)	Ozone conc on levels	ppmv	N	If flag .true.	Y
profiles(i) % co2(:)	CO_2 conc on levels	ppmv	N	If flag .true.	Y
profiles(i) % n2o(:)	N_2O conc on levels	ppmv	N	If flag .true.	Y
profiles(i) % co(:)	CO conc on levels	ppmv	N	If flag .true.	Y
profiles(i) % ch4(:)	CH_4 conc on levels	ppmv	N	If flag .true.	Y
profiles(i) % aerosols(:,:) (2^{nd} array element is layers)	Aerosol components conc on layers	cm^{-3}	N	Aerosol scatt	Y
profiles(i) % cloud(:,:)	Cloud water/ice content on layers	$g.m^{-3}$	N	Cloud option	Y
profiles(i) % cfrac(:,:)	Cloud fractional cover on layers	0-1	N	Cloud option	Y
profiles(i) % icede(:)	Ice particle effective diameter	microns	N	Never; only used with cloud option	Y
profiles(i) % clw(:)	Microwave cloud liquid water	Kg/kg	N	Microwave	Y
profiles(i) % idg	IWC eff diam scheme	1-4	N	Cloud option	N
profiles(i) % ish	Ice crystal shape	1-2	N	Cloud option	N
profiles(i) % s2m	2m surface variables (see Annex V)		Y		Y
profiles(i) % skin	Skin surface variable (see Annex V)		Y		t, fastem, salinity
profiles(i) % ctp	Cloud top pressure for simple cloud	hPa	N	Simple cloud	Y

profiles(i) % cfraction	Cloud fraction for simple cloud	0-1	N	Simple cloud	Y
profiles(i) % zenangle	Satellite zenith angle (0-90)	deg	Y		N
profiles(i) % azangle	Satellite azimuth angle (0-360; east=+90)	deg	N	Using FASTEM or Solar option	N
profiles(i) % sunzenangle	Solar zenith angle (0-90)	deg	N	Solar option	N
profiles(i) % sunazangle	Solar azimuth angle (0-360; east=+90)	deg	N	Solar option	N
profiles(i) % latitude	Latitude (-90 to +90)	deg	Y	For refractivity or emis atlas	N
profiles(i) % longitude	Longitude (0-360)	deg	N	Using IR or MW emis atlas	N
profiles(i) % snow_frac	Surface snow cover fraction	0-1	N	Using 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 Earth magnetic field and wave propagation direction		N	Zeeman	N

Table 14. Profile input parameters for user profile i.

6.4. Setting up input arrays before each call to RTTOV

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 calculated 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 15 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 channel indices are determined by **opts%ipcreg** as described in section 2.8.

A logical array **calcemis(:)** and a real (kind *jprb*) array **emissivity(:)** should be allocated with the same number of elements as **chanprof(:)**. For each radiance **j** to be calculated, the user may set **calcemis(j)** to true in which case either the ISEM or FASTEM model (for IR or MW instruments respectively) will be used to calculate sea surface emissivities, while for land and sea-ice surfaces, default values are used (see Tables 5 and 13). If **calcemis(j)** is set to false then the user must supply a suitable surface emissivity value in **emissivity(j)**. The next section describes how emissivity values may be obtained from the new IR and MW land surface emissivity atlases. The user may wish to use these atlases to populate the **emissivity(:)** array for profiles over land surfaces.

If Principal Component computations are being carried out (i.e. **opts%addpc** is true), the internal emissivity model must be used so **calcemis** should be set to true. If the user has requested radiances reconstructed from the PC scores (i.e. **opts%addradrec** is true), the array **channels_rec(:)** may be supplied. This should contain the indices of the channels for which reconstructed radiances are required. If it is omitted, reconstructed radiances for all channels will be returned. 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 15. Examples of RTTOV v10 input parameters.

6.5. Use of land surface emissivity atlases

The IR and MW emissivity atlases supplied with RTTOV v10 provide 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. 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. 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.

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

- **rttov_atlas_setup** : this allocates the necessary arrays and reads the atlas data appropriate to the sensor into them. This should be called after the **coefs%coef** structure has been initialised. See Annex K.
- **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(:)** array for input to RTTOV. See Annex L.
- **rttov_deallocate_atlas** : this carries out deallocation of arrays and should be called once the atlas is no longer required. See Annex M.

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. Examples of retrieving emissivity values from the atlases can be found in the atlas test programs `rttov_iratlas_test.F90`, `rttov_mwatlas_test.F90` and `rttov_cnrmmwatlas_test.F90` which are located in the `src/emis_atlas/` directory.

6.6. 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 N. 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. Each time the rttov input dimensions or options change, i.e. number of channels, absorbing gas, number of profiles, the user should call **rttov_alloc_traj** to deallocate the previous **traj** and allocate a new one). This capability has been enabled 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. A specific trial should be done by the user in order to test if this external allocation saves time or not. For Linux workstations (gfortran/ifort/pgf90/etc compilers) there is no benefit to using the **traj** structure.

6.7. Output arrays from RTTOV v10

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, calcemis,
emissivity, emissivity_out, transmission, radiancedata, 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.

Instances **transmission** and **radiancedata** of the derived types *transmission_type* and *radiance_type* should be declared. These may be initialised using the subroutines **rttov_alloc_transmission** and **rttov_alloc_rad** respectively (see Annexes H and G). Annex V defines fully these output radiance and transmittance structures. Table 16 defines in more detail which arrays are used for output by users and their 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**). There can be confusion in the role of the surface pressure in the output radiance quantities on layers (e.g. **rad%overcast** etc). All values relate to the standard pressure levels defined by the user *except* for the layer containing

the surface pressure where the level below the surface pressure contains the radiances from the surface level defined by the surface pressure not the profile level.

The **emissivity_out(:)** real array (kind *jprb*) should be allocated with the same number of elements as **chanprof(:)**. This array contains the surface emissivities 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 I). This structure will contain the computed PC scores, and (if **opts%addradrec** is true) the reconstructed radiances. This structure is also defined in Annex V. The **pccomp** argument is not required if **opts%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
real	cloudy(nchanprof)	Cloudy top of atmosphere radiance for 100% fraction for each channel at given cloud top pressure
real	upclear(nchanprof)	Clear sky upwelling radiance without reflection term
real	dnclear(nchanprof)	Clear sky downwelling radiance
real	reflclear(nchanprof)	Reflected clear sky downwelling radiance
real	overcast(nlayer,nchanprof)	Level to space overcast radiance given black cloud for each layer (layers,channels)
real	up(nlayer,nchanprof)	Above cloud upwelling atmospheric radiance for the layers defined by each pressure level down to surface
real	down(nlayer,nchanprof)	Above cloud downwelling atmospheric radiance for the layers defined by each pressure level down to surface
real	surf(nlayer,nchanprof)	Radiance emitted by a black cloud in each layer
Radiance_Type Brightness Temperatures *degK*		
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
Transmission_Type Transmittances 0-1		
real	tau_total(nchanprof)	Transmittance from surface for each channel
real	tau_levels(nlevel, nchanprof)	Transmittance from each standard pressure level to top of atmosphere for each channel
Output Emissivity 0-1		
real	emissivity_out(nchanprof)	Emissivity vales used in RTTOV calculation (same as input emissivity values if **calcemis** is false).

Table 16. Main RTTOV v10 output arrays. The green rows are those most commonly used.

6.8. Running RTTOV v10

You need to ensure the following in your program which calls RTTOV v10:

- 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 V.
- An instance of the *rttov_coefs* type should be populated either by calling **rttov_setup** or by calling **rttov_read_coefs** and **rttov_init_coefs**. In the former case, you must make sure the coefficient file for the instrument you want to simulate is in the same directory as the executable (or, better, a symbolic link to the file is made in the directory). In the latter case you can supply the coefficient filename as an input to **rttov_read_coefs**.
- Allocate the input/output structures to RTTOV with the number of channels, computed radiances, and profiles you want to run with. This can be achieved with the allocation routines described in Annexes F-I. For efficiency you may wish to allocate a trajectory structure using **rttov_alloc_traj** (Annex N). If the emissivity atlas is required this should be initialised now (see Annex K). See above and Annex L for a detailed description of the variables required for input to RTTOV v10 and Annex X for example code.
- Initialise the profile structure with your atmospheric profile. This is shown in Table 14 and listed in Annex V. Be careful that units for water vapour, ozone etc are volume mixing ratio (ppmv) and not specific concentration (kg/kg) as for RTTOV v7.
- 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 U). If this is used, **opts%do_checkinput** can be set to false. You may also find the **rttov_print_profile** subroutine (Annex U) useful for debugging purposes: this prints out the contents of a single profile structure.
- Inialise up the **chanprof** array with the channel and profile indexes as described in section 6.4.
- You may give a surface emissivity value 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.
- Ensure the variables **profiles(k)%zenangle** and **profiles(k)%azangle** contain the satellite zenith angle at the surface and satellite azimuth angle at the surface (from north – east is +90 and west is +270) for each profile. Note the latter can be set to zero unless either FASTEM-3 or FASTEM-4 are required or the reflected solar option is used.
- 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_prof**, **rttov_alloc_rad**, and (optionally) **rttov_alloc_pccomp** routines (see Annexes F, G, I and J). If the emissivity atlas was initialised, this should also be de-allocated now (see Annex M).
- 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 allow for variation in the path length follow the guidance in section 2.1.
- If you want to include reflected solar radiation for the SWIR channels follow the guidance in section 2.1.
- If you want to run the cloud or aerosol options follow the guidance in sections 2.5/2.6.
- 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. The test program supplied **rttovscatt_test** can be used as an example and similar rules apply to calling **rttov_direct**. Remember to compile using the `mw_scat` or `all` targets.
- The **rttov_parallel** routines (**rttov_parallel_direct**, **rttov_parallel_tl**, and so on) have an almost identical interface to the core routines (**rttov_direct** etc). One key additional parameter is **nthreads** which specifies the number of parallel threads to which RTTOV assigns the computations. To use the parallel routines, RTTOV v10 must be compiled with OpenMP.

7. Reporting and known bugs for RTTOV v10

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. 10.2)
- 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 v10 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.

There are several known issues in the version RTTOV v10.2 of the code which are listed below. Corrections to these will be provided via the RTTOV v10 web page http://research.metoffice.gov.uk/research/interproj/nwpsaf/rtm/rtm_rttov10.html as they become available. They are:

i. RTTOV v10 is not compatible with early versions of some compilers and version 10 of the pgf90 compiler has some limitations with the test suite (see section 5).

ii. On the NEC architecture some subroutines cannot be compiled with optimisations. See the *nec-meteofrance* file in the *build/arch/* directory. The internal RTTOV interpolator is found to run relatively slowly on the NEC when compared to other platforms.

iii. When calculating weighting functions (i.e. d(tau)/d(ln(p))) using transmittances on user levels calculated with the RTTOV interpolator, the resulting curves are not smooth. This was also the case in RTTOV v9.

8. Frequently asked questions

This section has now been put on the RTTOV v10 web site to allow updating.

9. 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.,(in press).*

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.

Borbas, E. E. and B. C. Ruston, 2010. The RTTOV UWiremis IR land surface emissivity module. NWP SAF report. http://research.metoffice.gov.uk/research/interproj/nwpsaf/vs_reports/nwpsaf-mo-vs-042.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

Eyre J. R. 1991 A fast radiative transfer model for satellite sounding systems. ECMWF Research Dept. Tech. Memo. 176 (available from the librarian at ECMWF).

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/publications

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

Liu, Q., F. Weng, and S. English, 2010: An Improved Fast Microwave Water Emissivity Model, *IEEE Geosci. Remote Sensing, (in press).*

Masuda, K., T. Takashima and Y. Takayama 1988 Emissivity of pure and sea waters for the model sea surface in the infrared window regions. *Remote Sensing Environ.* **24** *313-329.*

Matricardi, M., F. Chevallier and S. Tjemkes 2001 An improved general fast radiative transfer model for the assimilation of radiance observations. ECMWF Research Dept. Tech. Memo. 345. http://www.ecmwf.int/publications

Matricardi, M. 2003 RTIASI-4, a new version of the ECMWF fast radiative transfer model for the infrared atmospheric sounding interferometer. ECMWF Research Dept. Tech. Memo. 425. http://www.ecmwf.int/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 56.*

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.*

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://research.metoffice.gov.uk/research/interproj/nwpsaf/rtm/papers/isem6.pdf

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

Annex A - Coefficient conversion tools

The program **rttov_conv_coef.exe** (located in the `bin/` directory of the RTTOV build) can be used to convert ASCII coefficient files to binary files and to create coefficient files for subsets of channels.

The usage is as follows:

```
$ rttov_conv_coef.exe \
        --format-in  FORMATTED|UNFORMATTED \
        --format-out FORMATTED|UNFORMATTED \
        --channels 1 2 3 4 5 ... \
        --coef-in  ... --scaer-in  ... --sccld-in  ... --pccoef-in  ... \
        --coef-out ... --scaer-out ... --sccld-out ... --pccoef-out ...
```

Most arguments are optional, though both `--coef-in` and `--format-out` must be specified at least.

Argument	Description
--format-in FORMATTED|UNFORMATTED	Format of input coefficient file(s). FORMATTED=ASCII; UNFORMATTED=binary.
--format-out FORMATTED|UNFORMATTED	Format of output coefficient files(s).
--channels 1 2 3 4 5 ...	List of channels to extract.
--coef-in/--coef-out	Input/output RTTOV coefficient file.
--scaer-in/--scaer-out	Input/output aerosol scattering coefficient file.
--sccld-in/--sccld-out	Input/output cloud scattering coefficient file.
--pccoef-in/--pccoef-out	Input/output Principal Components coefficient file.

The program **rttov789_conv_coef.exe** (also located in the `bin/` directory) can be used to convert v7-, 8- and 9-format coefficient files to the v10 format. The input file must be in ASCII format. Note that this binary is not compiled by default: the user must specify an appropriate build target such as `all` or `coef_io_789`.

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 v10-compatible RTTOV coefficient file.

Annex B – RTTOV_ERRORHANDLING interface

```
call rttov_errorhandling (err_unit, verbosity_level)
```

rttov_errorhandling may be called at any time. The purpose is to define the level of information messages which are output, and the logical unit to which they are sent. The verbosity level allows the user to get various levels of error messages or all the information. The logical error unit defines the fortran file unit number on which messages are written. The default value is the one given in the **rttov_const** module, 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. The number of error messages output is also controlled.

Type	In/Out	Variable	Description
Integer	Intent(in)	err_unit	Logical error unit
Integer	Intent(in)	verbosity_level	0 = no error messages output 1 = FATAL errors only printed. These are errors which mean that profile should be aborted (e.g. unphysical profile input) 2 = WARNING errors only printed. Errors which can allow the computation to continue but the results may be suspect (e.g. profile outside basis profile limits) 3 = INFORMATION messages which inform the user about the computation Any other value is treated as 3.

Annex C – RTTOV_SETUP interface

```
call rttov_setup (
        err,
        err_unit,
        verbosity_level,
        opts,
        coefs,
        instrument,
        channels,
        channels_rec)
```

rttov_setup is called for each instrument. It defines the logical unit and verbosity level for information messages (by calling **rttov_errorhandling** – see Annex B) and it reads the coefficients for a set of instruments and an optional list of channels. The routine 'creates' the filename of the coefficient files. If **channels** is present, only the corresponding list of channels (all >0 values) is extracted from the coefficient file.

Type	In/Out	Variable	Description
Integer	Intent(out)	err	Return code
Integer	Intent(in)	err_unit	Logical error unit
Integer	Intent(in)	verbosity_level	0 = no error messages output 1 = FATAL errors only printed. 2 = WARNING errors only printed. 3 = INFORMATION messages Any other value is treated as 3
Type(rttov_options)	Intent(in)	opts	RTTOV options structure.
Type(rttov_coefs)	Intent(out)	coefs	RTTOV coefficients structure.
Integer	Intent(in)	instrument(3)	platform id; satellite id, instrument id (see Tables 2/3).
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 % addpc** and **opts % addradrec** are true).

Annex D – 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)
```

The user may opt to read the coefficient files manually, rather than through **rttov_setup**. This can be achieved using the **rttov_read_coefs** and **rttov_init_coefs** (see Annex D,E) subroutines, and has the advantage of allowing the user to specify the path and filenames for the coefficient files. If **channels** is present, only the corresponding list of channels (all >0 values) is extracted from the coefficient file.

Only the arguments relevant to the required coefficient files are necessary. The routine will attempt to guess the format of coefficient files if this is not supplied. The user can supply the path and filename of coefficient files, or the logical unit of the file if it has already been opened. If neither of these arguments are supplied, the **instrument** argument is mandatory so that the routine can use it to construct the coefficient filename.

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 % addpc** and **opts % addradrec** are true).
Character	Intent(in), optional	form_coef	Format of RTTOV coefficient file: should be either "unformatted" (binary) or "formatted" (ASCII).
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.

Annex E – RTTOV_INIT_COEFS interface

```
call rttov_init_coefs (
        err,
        opts,
        coefs )
```

If **rttov_read_coefs** has been called to read the coefficient file, this should be followed by a call to **rttov_init_coefs** to prepare the coefficients structure for use.

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

Annex F – 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.

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 % addclouds** or **opts % addaerosl** are true, otherwise it may be omitted.
Logical	Intent(in), optional	init	Additionally initialise profiles structure

Annex G – RTTOV_ALLOC_RAD interface

```
call rttov_alloc_rad (
             err,
             nchannels,
             radiance,
             nlayers,
             asw,
             init)
```

rttov_alloc_rad is called in the user's program to allocate or de-allocate the memory for the **radiance** 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)
Logical	Intent(in), optional	init	Additionally initialise radiance structure

Annex H – 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

Annex I – 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 % addradrec** is true.

Annex J – 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.

Annex K – RTTOV_ATLAS_SETUP interface

```
call rttov_atlas_setup (
              err,
              imonth,
              coef,
              path,
              ir_atlas_version,
              mw_atlas_version)
```

This routine initialises the emissivity atlas appropriate to the sensor defined in the **coefs % coef** structure (IR or MW). Climatological emissivities 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 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 in Annex L. 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.

Type	In/Out	Variable	Description
Integer	Intent(out)	err	Return code
Integer	Intent(in)	imonth	Month (1-12) of atlas data to be loaded.
Type(rttov_coef)	Intent(in)	coef	RTTOV instrument coefficients structure. NB this is **coefs % coef**
Character	Intent(in), optional	path	Path of directory containing emissivity atlas data files. Defaults to the current directory.
Integer	Intent(in), optional	ir_atlas_version	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_version	Specify MW atlas version. Valid values are 100 (the default) or 200 (see above and rttov_get_emis routine for more details).

Annex L – RTTOV_GET_EMIS interface

The **rttov_get_emis** takes the **profiles(:)** array as input: the routine uses the **latitude** and **longitude** members of each profile. 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**. 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,
              chanprof,
              profiles,
              coef,
              emissivity,
              emis_std,
              emis_flag)
```

The IR atlas also uses the **snow_frac** and **skin % surftype** members of the **profiles(:)** structure. 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_chanprof)	Intent(in)	chanprof(:)	Chanprof structure.
Type(profile_type)	Intent(in)	profiles(:)	Profiles structure.
Type(rttov_coef)	Intent(in)	coef	RTTOV instrument coefficient structure. NB this is **coefs % coef**
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,
              chanprof,
              profiles,
              coef,
              resolution,
              emissivity,
              emis_std,
              emis_cov)
```

TELSEM is loaded by default if no version is specified when calling **rttov_atlas_setup** 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_chanprof)	Intent(in)	chanprof(:)	Chanprof structure.
Type(profile_type)	Intent(in)	profiles(:)	Profiles structure.
Type(rttov_coef)	Intent(in)	coef	RTTOV instrument coefficient structure. NB this is **coefs % coef**
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,
              chanprof,
              profiles,
              coef,
              emissivity,
              pbats_veg)
```

The CNRM atlas has emissivity datasets 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_atlas_setup** must be called with **mw_atlas_version=200**.

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_coef)	Intent(in)	coef	RTTOV instrument coefficient structure. NB this is **coefs % coef**
Real	Intent(out)	emissivity(nchanprof)	Emissivity values.
Real	Intent(out), optional	pbats_veg	Vegetation type.

Annex M – RTTOV_DEALLOCATE_ATLAS interface

```
call rttov_deallocate_atlas (coef)
```

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

Type	In/Out	Variable	Description
Type(rttov_coef)	Intent(in)	coef	RTTOV instrument coefficient structure. NB this is **coefs % coef**

Annex N – 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 with every call to RTTOV, the user may use this subroutine to create one or more trajectory structures and pass them in to the RTTOV core routines (direct, TL, AD or K). Note this is primarily intended for certain supercomputer architectures (especially the NEC SX8) for which allocation/deallocation of memory can have significant overheads. This subroutine is not required for Linux workstations.

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**.

Annex O – 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%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 D). 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%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 2.8). 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 ...  --ipcreg ...
```

where the arguments are the PC coefficient file name and the index of the PC regression set respectively.

Annex P – RTTOV_DIRECT interface

```
call rttov_direct (errorstatus, chanprof, opts, profiles, coefs, calcemis,
                   emissivity, emissivity_out, transmission, radiancedata,
                   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(:)**.

For sea surfaces, the emissivity is calculated for channels where the corresponding **calcemis(:)** is true. The model used depends on the sensor and on the coefficient file: for IR the model is ISEM and for MW FASTEM 1-5. The version of the model inside the coefficient file defines the version of the emissivity algorithm or, for FASTEM, the required version may be specified in **opts%fastem_version**. For land surfaces, a fixed default value is used for IR, and a simple emissivity model is used for MW which requires **profiles(:)%skin%fastem(:)** to be populated with appropriate values (see Table 5). For channels where **calcemis(:)** is false, the user must supply corresponding emissivity values in **emissivity(:)**. The emissivity atlases may be used for this purpose for land (IR and MW) and sea-ice (IR only) surface types.

The files `src/test/example_fwd.F90` and `src/test/example_pc_fwd.F90` provide examples of running **rttov_direct** for classical RTTOV and PC-RTTOV.

For Principal Components calculations 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%ipcreg** for each profile being simulated in the call to **rttov_direct**, as illustrated in Table 15.

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.
Logical	Intent(in)	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).
Real	Intent(in)	emissivity(nchanprof)	Input emissivities.
Real	Intent(inout)	emissivity_out(nchanprof)	Emissivities used in RTTOV calculations.
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).
Type(rttov_traj)	Intent(inout), optional	traj	Trajectory structure to hold temporary data.
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 % addradrec** is .true.

Annex Q – RTTOV_K interface

```
call rttov_k (errorstatus, chanprof, opts, profiles, profiles_k, coefs,
              calcemis, emissivity, emissivity_k, emissivity_out,
              emissivity_out_k, transmission, transmission_k, radiancedata,
              radiancedata_k, traj, traj_k, pccomp, pccomp_k, profiles_k_pc,
              profiles_k_rec, channels_rec)
```

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** array contains the Jacobian for the surface emissivity.

The **opts%switchrad** flag determines the input perturbation array within **radiancedata_k**, and so the unit of **profiles_k**. If **switchrad** is false the radiance array **radiancedata_k%total** is the considered the input; if **switchrad** is true then the brightness temperature **radiancedata_k%bt** is the input perturbation. Ensure that all other arrays in **radiancedata_k** are initialised to 0.

In RTTOV v9, the **radiancedata_k** argument was optional, but in RTTOV v10 this argument is mandatory. To obtain the same behaviour as the default in RTTOV v9 you should initialise **radiancedata_k** to 0 (this can be done in the allocation call to **rttov_alloc_rad**), and then set either **radiancedata_k%total(:)** or **radiancedata_k%bt(:)** to 1.0, depending on the setting of **opts%switchrad**.

For Principal Components calculations, **profiles_k** will contain the Jacobians for the PC predictor channel set. These are *always* in terms of radiances (not brightness temperatures), regardless of the setting of **opts%switchrad**. If **opts%addradrec** is false, you should supply the **profiles_k_pc(:)** argument. RTTOV writes the Jacobian of the PC scores to this array. The size of the array must be the number of PC scores requested multiplied by the number of profiles. In this case **opts%switchrad** has no effect on the output.

If **opts%addradrec** is true, you should supply the **profiles_k_rec(:)** argument instead. RTTOV writes the Jacobian of the reconstructed radiances to this array. The size of the array must be the number of reconstructed radiances multiplied by the number of profiles. The setting of **opts%switchrad** determines the units of the output (radiances or brightness temperatures).

Note that for Principal Components calculations it is not possible to specify the input radiance, brightness temperature or PC score perturbation: **rttov_k** sets the perturbation within the code. The **pccomp_k** structure contains the perturbations set by the code: which member of this structure is perturbed depends on **opts%addradrec** and **opts%switchrad**.

The **emissivity_k** and **emissivity_out_k** arguments should be initialised to zero before calling **rttov_k**.

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)	K-matrix on profile variables.
Type(rttov_coefs)	Intent(in)	coefs	RTTOV coefficient structure.
Logical	Intent(in)	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).
Real	Intent(in)	emissivity(nchanprof)	Input emissivities.
Real	Intent(inout)	emissivity_k(nchanprof)	K matrix on input surface emissivity.
Real	Intent(out)	emissivity_out(nchanprof)	Emissivities used in RTTOV calculations.
Real	Intent(inout)	emissivity_out_k(nchanprof)	K matrix on output surface emissivity.
Type(transmission_type)	Intent(inout)	transmission	Output transmittances (0-1).
Type(transmission_type)	Intent(inout)	transmission_k	K of transmittances.
Type(radiance_type)	Intent(inout)	radiancedata	Direct model output radiances (mw/cm^{-1}/sr/m^2 & degK).
Type(radiance_type)	Intent(inout)	radiancedata_k	K of radiances.
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	K of principal components.
Type(profile_type)	Intent(inout). optional	profiles_k_pc(npcscores* nprofiles)	K matrix on principal component scores. Only required if **opts % addradrec** is .false.
Type(profile_type)	Intent(inout). optional	profiles_k_rec(nchannelsrec* nprofiles)	K matrix on profile variables for reconstructed radiance channels. Only required if **opts % addradrec** is .true.
Integer	Intent(in), optional	channels_rec(nchannelsrec)	Channels for which to compute reconstructed radiances if **opts % addradrec** is .true.

Annex R – RTTOV_TL interface

```
call rttov_tl (errorstatus, chanprof, opts, profiles, profiles_tl, coefs,
               calcemis, emissivity, emissivity_tl, emissivity_out,
               emissivity_out_tl, transmission, transmission_tl, radiancedata,
               radiancedata_tl, traj, traj_tl, pccomp, pccomp_tl, channels_rec)
```

rttov_tl should be called for each instrument required. Each call carries out computations for the profiles in the array **profiles(:)**. **rttov_tl** calculates the tangent linear of the RTTOV direct model.

The **profiles_tl(:)** array should be initialised with suitable perturbations about the profiles specified in **profiles(:)**. As an example, the subroutine `src/test/rttov_make_profile_inc.F90` is used to generate profile perturbations for the RTTOV test suite.

For Principal Components calculations and where the FASTEM model is used, RTTOV will calculate the surface emissivity perturbation from variables in **profiles_tl** (see Table 5). In all other cases, the surface emissivity perturbation may be specified in **emissivity_tl**. The emissivity perturbation used in the TL calculation is written to **emissivity_out_tl**.

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.
Logical	Intent(in)	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).
Real	Intent(in)	emissivity(nchanprof)	Input emissivities.
Real	Intent(in)	emissivity_tl(nchanprof)	Input surface emissivity increments.
Real	Intent(out)	emissivity_out(nchanprof)	Emissivities used in RTTOV calculations.
Real	Intent(out)	emissivity_out_tl(nchanprof)	TL on output surface emissivity.
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).
Type(radiance_type)	Intent(inout)	radiancedata_tl	TL of radiances.
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 % addradrec** is .true.

Annex S – RTTOV_AD interface

```
call rttov_ad (errorstatus, chanprof, opts, profiles, profiles_ad, coefs,
               calcemis, emissivity, emissivity_ad, emissivity_out,
               emissivity_out_ad, transmission, transmission_ad, radiancedata,
               radiancedata_ad, traj, traj_ad, pccomp, pccomp_ad, channels_rec)
```

rttov_ad should be called for each instrument required. Each call carries out computations for the profiles in the array **profiles(:)**. **rttov_ad** calculates the adjoint of the RTTOV direct model, the output being written to **profiles_ad**. The **emissivity_ad** array contains the adjoint for the surface emissivity.

The **opts%switchrad** flag determines the input perturbation array within **radiancedata_ad**, and so the unit of **profiles_ad**. If **switchrad** is false the radiance array **radiancedata_ad%total** is the considered the input; if **switchrad** is true then the brightness temperature **radiancedata_ad%bt** is the input perturbation. Ensure that all other arrays in **radiancedata_ad** are initialised to 0.

For Principal Components calculations, **profiles_ad** will contain the adjoints for the PC predictor channel set. These are *always* in terms of radiances (not brightness temperatures), regardless of the setting of **opts%switchrad**. The input perturbation must be specified in the **pccomp_ad** structure: the member to which the perturbation applies depends on the setting of **opts%addradrec** and **opts%switchrad**. If reconstructed radiances are not required (**opts%addradrec** is false), then the input perturbation should be specified in **pccomp_ad%pcscores**. If reconstructed radiances are required, the input perturbation should be specified in **pccomp_ad%clear_pccomp** if **opts%switchrad** is false, or **pccomp_ad%bt_pccomp** if **opts%switchrad** is true. For PC-RTTOV, all elements of the **radiancedata_ad** structure should be initialised to 0.

The **emissivity_ad** and **emissivity_out_ad** arguments should be initialised to zero before calling **rttov_ad**.

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)	Input profile variable increments.
Type(rttov_coefs)	Intent(in)	coefs	RTTOV coefficient structure.
Logical	Intent(in)	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).
Real	Intent(in)	emissivity(nchanprof)	Input emissivities.
Real	Intent(inout)	emissivity_ad(nchanprof)	AD on input surface emissivity.
Real	Intent(out)	emissivity_out(nchanprof)	Emissivities used in RTTOV calculations.
Real	Intent(inout)	emissivity_out_ad(nchanprof)	AD on output surface emissivity.
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).
Type(radiance_type)	Intent(inout)	radiancedata_ad	AD of radiances.
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	AD of principal components.
Integer	Intent(in), optional	channels_rec(nchannelsrec)	Channels for which to compute reconstructed radiances if **opts % addradrec** is .true.

Annex T – RTTOV_SCATT interface

```
call rttov_scatt (errorstatus, nlevels, chanprof, frequencies, profiles,
                  cld_profiles, coef_rttov, coef_scatt, calcemiss,
                  emissivity_in, radiance, cfrac, lnewcld, lusercfrac)
```

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_fwd_rttovscatt.F90` for an example of using **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.
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)	profiles_cld(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).
Real	Intent(in)	emissivity_in(nchanprof)	Input 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).
Logical	Intent(in), optional	lnewcld	.true. for revised cloud/rain treatment, .false. for old treatment (default is .true.)
Logical	Intent(in), optional	lusercfrac	.true. to take average cloud fraction from that supplied (default is .false.)

Annex U – RTTOV Utility routines

1. RTTOV_USER_OPTIONS_CHECKINPUT interface

`call` **`rttov_user_options_checkinput`**`(opts, coefs, err)`

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
Type(rttov_options)	Intent(in)	opts	Options structure.
Type(rttov_coefs)	Intent(in)	coefs	RTTOV coefficients structure.
Integer	Intent(out)	err	Return code.

2. RTTOV_USER_PROFILE_CHECKINPUT interface

`call` **`rttov_user_profile_checkinput`**`(opts, prof, coefs, err)`

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. In this case, **opts % 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 regression limits are exceeded, **errorstatus_warning** is returned in **err**. 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
Type(rttov_options)	Intent(in)	opts	Options structure.
Type(profile_type)	Intent(in)	prof	Profile structure.
Type(rttov_coefs)	Intent(in)	coefs	RTTOV coefficients structure.
Integer	Intent(out)	err	Return code.

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. AER_CLIM_PROF.EXE

This executable may be used to generate profiles on the layers defined by the user's pressure levels for the 10 RTTOV aerosol types (excluding volcanic ash) 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 `aer_clim_prof.exe` from the `data/` directory. The program prompts the user for a latitude, an elevation, 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_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, East=0, positive counter-clockwise)
Real	Intent(in)	sensor_relative_aziang	Solar azimuth angle relative to sensor azimuth angle (0 to 360, positive counter-clockwise)
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.

Annex V – RTTOV v10 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.

Type	Variable	Description
Integer	ipcreg	The index of the required set of PC predictors: 1-4 for IASI (corresponding to 300, 400, 500 and 600 predictors), and 1-3 for AIRS (for 200, 300 and 400 predictors) (default = -1)
Integer	fastem_version	Select the version of the FASTEM model to use. Valid versions are 1-5; other values result in the version specified in the coeff file being used (default = 0).
Real	cldstr_threshold	Threshold for stream weights to compute for IR scattering calculations (default = -1); recommended to be set negative when calling TL, AD or K models.
Logical	addinterp	If true, input profiles may be supplied on user-defined levels, and internal interpolation is used (default = false).
Logical	addpc	If true, carry out Principal Components calculations (default = false).
Logical	addradrec	If true, and addpc is true, the PC calculations will return reconstructed radiances, as well as the PC scores (default = false)
Logical	addsolar	If true, enable calculation of solar component for near-IR channels (default = false)
Logical	addaerosl	If true, account for scattering due to aerosols (default = false)
Logical	addclouds	If true, account for scattering due to clouds (default = false)
Logical	switchrad	Determines input perturbation for AD/K routines: if true, radiancedata_ad/k % bt is used, otherwise radiancedata_ad/k % total is used (default = false)
Logical	spacetop	If true, treat user's model top as space boundary (default = true)
Logical	lgradp	Allow TL/AD of user pressure levels if addinterp is true (default = false)
Logical	use_q2m	If true, activate use of surface humidity (default = false)
Logical	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 warnings (default = false)
Logical	verbose_checkinput_warnings	If false and apply_reg_limits is false, turns off warnings when input profiles exceed regression limits (default = true)
Logical	ozone_data	If true, user is supplying ozone profiles (default = false)
Logical	co2_data	If true, user is supplying co2 profiles (default = false)
Logical	n2o_data	If true, user is supplying n2o profiles (default = false)
Logical	co_data	If true, user is supplying co profiles (default = false)
Logical	ch4_data	If true, user is supplying ch4 profiles (default = false)
Logical	clw_data	If true, user is supplying cloud liquid water profiles (for MW) (default = false)
Logical	addrefrac	If true, RTTOV calculations account for atmospheric refraction (default = false)

Logical	do_checkinput	If true, checks whether input profiles are within both absolute and regression limits. If false, no check is performed (default = true)

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
Real	t	Radiative skin temperature (K)
Real	salinity	Practical salinity unit %o – *FASTEM-4/5 only.*
Real	fastem(1:5)	Land/sea-ice surface parameters for FASTEM.
Surface 2m – type *s2m_type*		
Real	t	Temperature (K)
Real	q	Water vapour (ppmv) – *only used if opts%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)
Atmospheric profile – type *profile_type*		
Character(len=128)	id	User may give text ID to each profile.
Integer	date(3)	Year, month, day – *not currently used.*
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).
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*
Real	aerosols(naertyp,nlayers) *(naertyp is the number of aerosol types, currently 11)*	Aerosols (cm^{-3}) – *IR only.* Note: the number of aerosol types is defined in the IR scattering coefficient file.
Real	cloud(ncldtyp,nlayers) *(ncldtyp is the number of water cloud types – currently 5 – plus 1 for cirrus clouds)*	Cloud water/ice (g/m^{-3}) – *IR only.* Note: the number of water cloud types is defined in the IR scattering coefficient file.
Real	cfrac(ncldtyp,nlayers)	Cloud fractional cover (0-1) – *IR only. Specify at most one non-zero value per layer.*
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.*

Integer	idg	Scheme for IWC to eff diameter,Dg 1=Ou and Liuou; 2=Wyser et al.; 3=Boudala et al; 4=McFarquhar et al.
Integer	ish	Choose the shape of ice crystals, 1=Hexagonal ice crystals; 2=Ice aggregates
Real	zenangle	Local satellite zenith angle (0-90 deg)
Real	azangle	Local satellite azimuth angle (0-360 deg; east=90)
Real	sunzenangle	Local solar zenith angle (0-90 deg)
Real	sunazangle	Local solar azimuth angle (0-360 deg; east=90)
Real	elevation	Elevation (km)
Real	latitude	Latitude (deg) -90 to +90
Real	longitude	Longitude (deg) 0-360. *Used only by IR emissivity atlas.*
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).
Real	cfraction	Cloud fraction (0 - 1) 1 for 100% cloud cover.

Chanprof structure

The *rttov_chanprof* structure replaces the "channels" and "lprofiles" arrays of earlier RTTOV versions. An array should 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 15).

Type	Variable	Description
Integer	chan	Channel index.
Integer	prof	Profile index.

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
Real	total(nchanprof)	Clear+cloudy top of atmosphere radiance for given cloud top pressure and fraction for each channel
Real	cloudy(nchanprof)	Cloudy top of atmosphere radiance for 100% fraction for each channel at given cloud top pressure
Real	upclear(nchanprof)	Clear sky upwelling radiance without reflection term
Real	dnclear(nchanprof)	Clear sky downwelling radiance
Real	reflclear(nchanprof)	Reflected clear sky downwelling radiance
Real	overcast(nlayer,nchanprof)	Level to space overcast radiance given black cloud on each level (layers,channels)
Real	up(nlayer,nchanprof)	Above cloud upwelling atmospheric radiance for each pressure level down to surface
Real	down(nlayer,nchanprof)	Above cloud downwelling atmospheric radiance for each pressure level down to surface

Real	surf(nlayer,nchanprof)	Radiance emitted by a black cloud at each pressure level except for the surface
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

Transmission structure

The *transmission_type* structure now has just two members. The transmittances are unitless and lie in the interval [0, 1].

Type	Variable	Description
Real	tau_total(nchanprof)	Transmittance from surface for each channel
Real	tau_levels(nlevel, nchanprof)	Transmittance from each standard pressure level to top of atmosphere for each channel

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 % 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	clear_pccomp(nchannels_rec)	Clear-sky radiances constructed using principal components.
Real	bt_pccomp(nchannels_rec)	BTs equivalent to reconstructed radiances.

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.
Real	cfrac	***Optional:*** *if 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 (kg/(m^2)/s)
Real	sp(:)	nlevels of solid precipitation flux (kg/(m^2)/s)

Annex W – Contents of rttov_const.F90

```
!
Module rttov_const
  ! Description:
  ! Definition of all parameters (constants) for 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 2002, EUMETSAT, All Rights Reserved.
  !
  ! History:
  ! Version   Date     Comment
  ! -------   ----     -------
  !  1.0   01/12/2002  New F90 code with structures (P Brunel A Smith)
  !  1.1   29/01/2003  New platforms and instruments (P Brunel)
  !                    Hard limits for input profiles
  !  1.2   19/02/2003  Some changes to limits and comments (R Saunders)
  !  1.3   06/05/2003  Change version number to 7.3.1
  !                    and add references for physical constants (P Brunel)
  !  1.4      08/2003  Added variables for MW scattering (F Chevallier)
  !  1.5   18/09/2003  Added coefficients for cloud absorption properties (P
Francis)
  !  1.6   15/10/2003  Added new sections in parameter files for scatt   (F
Chevallier)
  !  1.7   23/11/2003  Added new definitions of polarisations 2.1 (S English)
  !  1.8   25/08/2005  Made inst_name a parameter (R Saunders)
  !  1.9   11/01/2006  Added logical flag for surface humidity use (R Saunders)
  !  1.10  12/01/2006  Marco Matricardi (ECMWF):
  !           --       Added variables for CO2,CO,N2O and CH4 molecules.
  !           --       Added parameters for the computation of the refractive
index
  !           --       of air.
  !  1.11  06/02/2006  Added logical flag for linear in tau approx (R Saunders)
  !  1.12  06/04/2006  Added Meghatropiques (R. Saunders)
  !  1.13  14/03/2007  Added units conversion constants
  !  1.14  16/05/2007  Added polarimetric sensor type (R Saunders)
  !  1.15  25/09/2007  Added maximum number of warnings for checkinput (P
Brunel)
  !  1.16  11/10/2007  Remove zhusta* and zice* constants ( P.Marguinaud )
  !  1.17  07/12/2007  Remove maximum number of warnings for checkinput (P
Brunel)
  !  1.18  12/12/2007  Added hard limits for trace gases (R Saunders)
  !  1.19  13/12/2007  Renamed linear_tau (R Saunders)
  !  1.20  01/11/2007  Added parameters for section length and AD/K code (A.
Geer)
  !  1.21  16/01/2008  Facility to apply regression limits  (N. Bormann)
  !  1.22  04/03/2008  Made min hard limit > zero (R Saunders)
  !  1.23  14/04/2008  Added SSM/T2 (R Saunders)
  !  1.24  02/06/2008  Changed mixing ratio for CO (R Saunders)
  !  1.25  12/08/2008  Added SSMISZ for SSMIS chan19-22 - Zeeman (P. Rayer)
```

```
  !  1.26  29/01/2009  Add Kalpana and FY-3 (R Saunders)
  !  1.27  26/05/2009  Add more platforms and sensors (R Saunders)
  !  1.28  02/12/2009  Add principal component capability (Marco matricardi)
  !  1.29  15/01/2010  Add rttov9 intervals constants (P Marguinaud)
  !  1.30  05/07/2010  Add maximum solar zenith angle constant (J Hocking)
  !  1.31  01/02/2011  Updates to platform and sensor lists (J Hocking)
  !
  ! 2010/03 Code cleaning Pascal Brunel, Philippe Marguinaud
  !

  Use parkind1, Only : jpim     ,jprb
  Implicit None

  !1.0 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

  !1.1 general
  !-----------
  ! Version number of the current code

  Integer(Kind=jpim), Parameter :: version = 10
  Integer(Kind=jpim), Parameter :: release = 2
  Integer(Kind=jpim), Parameter :: minor_version = 0

  Integer(Kind=jpim), Parameter :: version_compatible_min = 10 ! minimum version
number
  Integer(Kind=jpim), Parameter :: version_compatible_max = 10 ! maximum version
number
          ! compatible for coefficients.
          ! coef files with "id_comp_lvl" outside range will be rejected

  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

  !1.2 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)
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

! Gravity from NIST 9.80665 ms-1 (exact)
Real(Kind=jprb), Parameter :: gravity = 9.80665_JPRB

!
! 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

! The Cosmic Microwave Background Spectrum from the Full COBE FIRAS Data Set
! Fixsen D.J. et all
! 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

! 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

! zero temperature(K)
Real(Kind=jprb), Parameter :: t0 =273.15

!1.3 satellite and instrument information
!----------------------------------------

!platform id codes
Integer(Kind=jpim), Parameter :: nplatforms = 29
```

```
  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_sentinel = 19, &
       & platform_id_meghatr  = 20, &
       & platform_id_kalpana  = 21, &
       & platform_id_insat_3d = 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 names
  Character (len=8), Parameter :: platform_name(nplatforms) = &
       & (/ 'noaa    ', 'dmsp    ', 'meteosat', 'goes    ', 'gms     ', &
          & 'fy2     ', 'trmm    ', 'ers     ', 'eos     ', 'metop   ', &
          & 'envisat ', 'msg     ', 'fy1     ', 'adeos   ', 'mtsat   ', &
          & 'coriolis', 'jpss    ', 'gifts   ', 'sentinel', 'meghatr ', &
          & 'kalpana ', 'insat_3d', 'fy3     ', 'coms    ', 'meteor-m', &
          & 'gosat   ', 'calipso ', 'dummy   ', 'gcom-w  '/)

  !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_amsr   = 17, inst_id_mtsatim= 18,
inst_id_atms   = 19, &
       & inst_id_mviri  = 20, inst_id_seviri = 21, inst_id_goesim = 22,
inst_id_goessd = 23, &
       & inst_id_gmsim  = 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_kavhrr = 37, inst_id_iimager= 38,
inst_id_isoundr= 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

  Integer(Kind=jpim), Parameter :: ninst = 54
  ! List of instruments  !!!! HIRS is number 0
  Character (len=8), Dimension(0:ninst-1),parameter :: inst_name =       &
        & (/ 'hirs    ', 'msu     ', 'ssu     ', 'amsua   ', 'amsub   ',  &
           & 'avhrr   ', 'ssmi    ', 'vtpr1   ', 'vtpr2   ', 'tmi     ',  &
           & 'ssmis   ', 'airs    ', 'hsb     ', 'modis   ', 'atsr    ',  &
           & 'mhs     ', 'iasi    ', 'amsr    ', 'imager  ', 'atms    ',  &
           & 'mviri   ', 'seviri  ', 'imager  ', 'sounder ', 'imager  ',  &
           & 'vissr   ', 'mvisr   ', 'cris    ', 'cmis    ', 'viirs   ',  &
           & 'windsat ', 'gifts   ', 'ssmt1   ', 'ssmt2   ', 'saphir  ',  &
           & 'madras  ', 'ssmisz  ', 'kavhrr  ', 'iimager ', 'isoundr ',  &
           & 'mwts    ', 'mwhs    ', 'iras    ', 'mwri    ', 'abi     ',  &
           & 'mi      ', 'msumr   ', 'tansofts', 'iir     ', 'mwr     ',  &
           & 'dummyir ', 'dummymw ', 'dummyhi ', 'dummypo '/)


  !1.4 Coefficient file Section names
  !----------------------------------
  Integer(Kind=jpim), Parameter :: nsections = 38
  Integer(Kind=jpim), Parameter :: lensection = 23
  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       '/)

  !sensors 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

!sensors names
Character (len=2), Parameter :: sensor_name(nsensors) = &
      & (/ 'ir', 'mw', 'hi', 'po' /)

! these codes are for the instrument from 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 /)

!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 names
Character (len=12), Parameter :: gas_name(ngases_max) = &
      & (/ 'Mixed_gases ', &
         & 'Water_vapour', &
         & 'Ozone       ', &
         & 'WV_Continuum', &
         & 'CO2         ', &
         & 'N2O         ', &
         & 'CO          ', &
         & 'CH4         ' /)

!gas units
Integer(Kind=jpim), Parameter :: ngases_unit = 2
Integer(Kind=jpim), Parameter :: &
      & gas_unit_specconc  = 1, &
      & gas_unit_ppmv      = 2
Character (len=12), Parameter :: gas_unit_name(ngases_unit) = &
      & (/ 'spec. concen', &
         & 'ppmv        '  /)


!1.5 error reporting
!-------------------
!error status values
Integer(Kind=jpim), Parameter :: nerrorstatus = 3
Integer(Kind=jpim), Parameter :: errorstatus_success = 0
Integer(Kind=jpim), Parameter :: errorstatus_warning = 1
```

```
  Integer(Kind=jpim), Parameter :: errorstatus_fatal   = 2
  Integer(Kind=jpim), Parameter :: errorstatus_info    = 3
  Character(len=*), Parameter :: errorstatus_text(0:nerrorstatus) = &
       & (/ 'success', &
       & 'warning', &
       & 'fatal  ', &
       & 'info   '  /)


  !1.6 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

  !1.7 water types
  !---------------
  Integer(Kind=jpim), Parameter :: nwatertype = 1
  Integer(Kind=jpim), Parameter :: watertype_fresh_water = 0
  Integer(Kind=jpim), Parameter :: watertype_ocean_water = 1

  !1.8 cloud emissivity
  !--------------------
  Integer(Kind=jpim), Parameter :: overlap_scheme = 2    ! overlap scheme
  ! 1 => Geleyn and Hollingsworth (1979)
  ! 2 => Raisanen (1998)

  !
  !1.9 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
  ! 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)


  !1.10  Maximum Optical Depth
  !--------------------------
  ! 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

  !1.11  Maximum solar zenith angle for which to apply solar calculation
  !---------------------------------------------------------------------
  Real(Kind=jprb), Parameter  :: max_sol_zen = 84._JPRB


  !2 RTTOV7 aux parameters
  !----------------------
  Integer(Kind=jpim), Parameter :: fastem_sp = 5  ! max. number of fastem
surface parameters
  Real(Kind=jprb), Parameter    :: mwcldtp = 322.0_JPRB  ! Upper pressure level
(HPa) for lwp calcs
  Real(Kind=jprb), Parameter    :: pressure_top = 0.004985_JPRB ! Pressure of
top level for
                                                  ! Line/Line calculations (hPa)
  Real(Kind=jprb) , Dimension(8), Parameter :: dcoeff =        &! 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/)

  !2.1 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), Dimension(7), Parameter :: npolar_compute = &
   & (/ 2, 2, 2, 1, 1, 2, 4/)
  Integer(Kind=jpim), Dimension(7), Parameter :: npolar_return = &
   & (/ 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/) )

!3 RTTOVSCATT aux parameters
!---------------------------
! 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

!4 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
```

```
!5 RTTOV8_M_SCATT
!-----------------------------------------------------
Integer(Kind=jpim), Parameter :: naer_max = 11
Integer(Kind=jpim), Parameter :: naer_cl  = 10
Integer(Kind=jpim), Parameter :: nhumaer(naer_max)=                    &
     & (/1,8,1,8,8,1,1,1,1,8,1/)

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

Character (len=4), Parameter :: aer_name(naer_max) = &
      & (/ 'inso', &
         & 'waso', &
         & 'soot', &
         & 'ssam', &
         & 'sscm', &
         & 'minm', &
         & 'miam', &
         & 'micm', &
         & 'mitr', &
         & 'suso', &
         & 'vola' /)

Integer(Kind=jpim), Parameter :: nwcl_max = 5
Integer(Kind=jpim), Parameter :: nhumwcl(nwcl_max)=                    &
     & (/1,1,1,1,1/)

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

Integer(Kind=jpim), Parameter:: jpazn=11

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
```

```
  Real(Kind=jprb), Parameter :: min_tau = 1.0e-8_JPRB
  Real(Kind=jprb), Parameter :: min_od  = 1.0e-5_JPRB

!
! These are the RTTOV9 wavenumbers that make intervals
!
  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
End Module rttov_const
```

Annex X – 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 SUITE FORWARD MODEL ONLY
  !          RTTOV VERSION 10
  ! To run this program you must have the following files
  ! either resident in the same directory or set up as a
  ! symbolic link:
  !   prof.dat                       --  input profile
  !   rtcoef_platform_id_sensor.dat --  coefficient file to match
  !   the sensor you request in the input dialogue
  ! 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.
  !
  !
  ! If the user wants to use this example to create his own
  ! program he will have to modify the code between
  ! comment lines of that kind:
  !     !================================
  !     !======Read =====start===========
  !          code to be modified
  !     !======Read ===== end ===========
  !     !================================
  !
  ! Current Code Owner: SAF NWP
  !
  ! History:
  ! Version   Date        Comment
  ! -------   ----        -------
  !  1.0    27/04/2004   orginal (based on tstrad) P. Brunel
  !  1.1    09/08/2004   modified to allow for variable no. channels/per profile
  !                       R. Saunders
  !  1.2    13/04/2007   Modified for RTTOV-90
  !  1.3    31/07/2007   Modified for RTTOV-91 R Saunders
  !  1.4    11/10/2007   Parallel version P.Marguinaud
  !  2.0    25/06/2010   Modified for RTTOV-10 J Hocking
  !  2.1    23/11/2011   Updates for v10.2 J Hocking
  !
  ! Code Description:
  !   Language:          Fortran 90.
  !   Software Standards: "European Standards for Writing and
  !     Documenting Exchangeable Fortran 90 Code".
  !
  Use rttov_const, Only :  &
```

```
       & errorstatus_success,&
       & errorstatus_warning,&
       & errorstatus_fatal  ,&
       & q_mixratio_to_ppmv, &
       & platform_name,      &
       & inst_name

  Use rttov_types, Only :   &
       & rttov_options,     &
       & rttov_coefs,       &
       & profile_Type,      &
       & transmission_Type, &
       & radiance_Type,     &
       & rttov_chanprof

  Use parkind1, Only : jpim, jprb, jplm
  !
  Implicit None
  !
#ifdef _RTTOV_EXAMPLE_FWD_PARALLEL
#include "rttov_parallel_direct.interface"
#define rttov_direct rttov_parallel_direct
#else
#include "rttov_direct.interface"
#endif
#include "rttov_setup.interface"
#include "rttov_copy_prof.interface"
#include "rttov_dealloc_coefs.interface"
#include "rttov_alloc_rad.interface"
#include "rttov_alloc_transmission.interface"
#include "rttov_alloc_prof.interface"
#include "rttov_errorreport.interface"
#include "rttov_user_options_checkinput.interface"
#include "rttov_print_opts.interface"
#include "rttov_print_profile.interface"


  !-------------------------
  !
  Integer(Kind=jpim) :: iup=20      ! unit for profile file
  Integer(Kind=jpim) :: ioout=21    ! unit for output

  ! One profile per call and one sensor only for this simple example
  Integer (Kind=jpim) :: nprof = 1           ! Number of profiles per call
  Integer(Kind=jpim)  :: nrttovid = 1        ! Number of sensor coeff files to
read

  ! RTTOV_errorhandling interface
  !====================
  Integer(Kind=jpim) :: Err_Unit        ! Logical error unit (<0 for default)
  Integer(Kind=jpim) :: verbosity_level ! (<0 for default)

  ! RTTOV_setup interface
  !====================
  Integer(Kind=jpim)              :: setup_errorstatus  ! setup return code
  Integer(Kind=jpim), Allocatable :: instrument(:,:)    ! platform id, sat id
and sensor id
  Type(rttov_options)             :: opts               ! Options structure
  Type(rttov_coefs), Allocatable  :: coefs(:)           ! Coefficients structure
```

```
  ! RTTOV interface
  !====================
  Integer(Kind=jpim)                :: rttov_errorstatus  ! rttov error return
code
  Integer(Kind=jpim)                :: nchannels, nchanprof
  Type(rttov_chanprof), Allocatable :: chanprof(:)
  Type(profile_Type), Allocatable   :: profiles(:)
  Logical(Kind=jplm), Allocatable   :: calcemis(:)
  Real(Kind=jprb), Allocatable      :: emissivity_in (:)
  Real(Kind=jprb), Allocatable      :: emissivity_out (:)
  Type(transmission_Type)           :: transmission ! transmittances and layer
optical depths
  Type(radiance_Type)               :: radiance

  Integer(Kind=jpim) :: alloc_status(20)
  Integer(Kind=jpim) :: errorstatus
  Real(Kind=jprb), Allocatable :: emissivity(:)
  Character (len=80) :: errMessage
  Character (len=11)  :: NameOfRoutine = 'example_fwd'

  ! variables for input
  !====================
  Integer(Kind=jpim), Parameter :: mxchn = 9000 ! max number of channels
  Integer(Kind=jpim) :: input_chan(mxchn)
  Real(Kind=jprb)    :: input_ems(mxchn)
  Real(Kind=jprb)    :: zenith
  Real(Kind=jprb)    :: azimut
  Real(Kind=jprb)    :: lat
  Real(Kind=jprb)    :: zerht
  Real(Kind=jprb)    :: sunzang
  Real(Kind=jprb)    :: sunazang
  Integer(Kind=jpim) :: nlevels
  Integer(Kind=jpim) :: ivch, ich
  Integer(Kind=jpim) :: asw          ! allocate or deallocate switch
  Real(Kind=jprb)    :: ems_val
  Integer(Kind=jpim), Allocatable :: nchan(:) ! number of channels per profile
  Integer(Kind=jpim) :: isurf
  Integer(Kind=jpim) :: nwater
  logical(Kind=jplm) :: addrefrac
  logical(Kind=jplm) :: addsolar
  logical(Kind=jplm) :: addaerosl
  logical(Kind=jplm) :: addclouds
  logical(Kind=jplm) :: all_channels
  Logical(Kind=jplm) :: addinterp  ! switch for the interpolator
  ! printing arrays
  Real(Kind=jprb), Allocatable :: pr_radcld(:)
  Real(Kind=jprb), Allocatable :: pr_trans(:)
  Real(Kind=jprb), Allocatable :: pr_emis(:)
  Real(Kind=jprb), Allocatable :: pr_trans_lev(:,:)
  ! 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 ------------------------------------------------------
```

```
  errorstatus     = 0_jpim
  alloc_status(:) = 0_jpim
  allocate (instrument(3,nrttovid),stat= alloc_status(1))

  !=====================================================
  !========== Interactive inputs == start ==============
  Write(0,*) 'enter platform number'
  Read(*,*) instrument(1,nrttovid)
  Write(0,*) 'enter satellite number '
  Read(*,*) instrument(2,nrttovid)
  Write(0,*) 'enter instrument number'
  Read(*,*) instrument(3,nrttovid)
  Write(0,*) 'enter surface type (0=land, 1=sea, 2=ice/snow)'
  Read(*,*) isurf
  Write(0,*) 'enter water type (0=fresh water, 1=ocean water)'
  Read(*,*) nwater
  Write(0,*) 'enter number of profile levels'
  Read(*,*) nlevels
  Write(0,*) 'enter zenith angle in degrees'
  Read(*,*) zenith
  !
  ! Prescribe other inputs
  azimut = 0._jprb   ! Satellite azimuth angle
  sunzang = 0._jprb  ! solar zenith angle
  sunazang = 0._jprb ! solar azimuth angle
  lat = 0._jprb      ! profile latitude
  zerht = 0._jprb    ! elevation of surface
  !
  ! Set flags
  addrefrac=.True.     ! include refraction in path calc
  addsolar=.False.     ! Do not include reflected solar
  addaerosl=.False.    ! Don't include aerosol effects
  addclouds=.False.    ! Don't include cloud effects
  all_channels=.True.  ! Read all channels into memory from coef file
  addinterp = .True.   ! Allow interpolation of input profile
  !
  Allocate (nchan(nprof))
  nchan(:)  = 0_jpim
  nchannels = 0_jpim
  Read(*,*,iostat=ios) ich, ivch, ems_val ! channel number, validity, emissivity
  Do While (ios == 0 )
     If( ivch /= 0 ) Then
        nchannels = nchannels + 1_jpim
        input_chan(nchannels) = ich
        input_ems(nchannels)  = ems_val
     Endif
     Read(*,*,iostat=ios) ich, ivch, ems_val
  End Do

  nchan(:) = nchannels
  nchanprof=SUM(nchan(:))  ! here it is also nchannels * nprof

  !Pack channels and emmissivity arrays
  Allocate(chanprof(nchanprof))   ! Note these array sizes nchan can vary per
profile
  Allocate(emissivity(nchanprof)) ! but for this example assume 1 profile/call
with same channels
  Allocate(emissivity_out(nchanprof))
```

```
! Build the list of profile indices
nch = 0_jpim
Do j = 1 , nprof
  DO  jch = 1,nchan(j)
    nch = nch + 1_jpim
    chanprof(nch)%prof = j
    chanprof(nch)%chan = input_chan(jch)
    emissivity(nch)    = input_ems(jch)
  End Do
End Do

! Initialise options structure
opts % addrefrac  = addrefrac
opts % addinterp  = addinterp
opts % addsolar   = addsolar
opts % addclouds  = addclouds
opts % addaerosl  = addaerosl
opts % ozone_data = .True.       ! we have an ozone profile
opts % co2_data   = .False.      ! we do not have profiles
opts % n2o_data   = .False.      ! for any other constituents
opts % ch4_data   = .False.      !
opts % co_data    = .False.      !
opts % clw_data   = .False.      !

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

!Initialise error management with default value for
!the error unit number and all error message output
Err_unit = -1
verbosity_level = 3_jpim
Call rttov_errorhandling(Err_unit, verbosity_level)

allocate (coefs(nrttovid),stat= alloc_status(1))
If( any(alloc_status /= 0) ) then
   errorstatus = errorstatus_fatal
   Write( errMessage, '( "mem allocation error for coefs")' )
   Call rttov_errorreport (errorstatus, errMessage, NameOfRoutine)
   Stop
End If

!Read and initialise coefficients
Call rttov_setup (&
    & setup_errorstatus,  &! out
    & Err_unit,           &! in
    & verbosity_level,    &! in
    & opts,               &! in
    & coefs(nrttovid),    &! out
    & instrument)          ! in
if( setup_errorstatus /= errorstatus_success ) then
   write ( *,* ) 'rttov_setup fatal error'
   stop
endif

! security if input number of channels is higher than number
! stored in coeffs
If( nchannels > coefs(nrttovid) % coef % fmv_chn ) Then
    nchannels = coefs(nrttovid) % coef % fmv_chn
```

```
      nchan(nprof) = coefs(nrttovid) % coef % fmv_chn
  Endif

  !Ensure the options and coefficients are consistent
  Call rttov_user_options_checkinput(opts, coefs(nrttovid), errorstatus)
  If( errorstatus /= errorstatus_success ) Then
     write ( *,* ) 'error in rttov options'
     Stop
  Endif

  !Open output file which prints results

Open(IOOUT,file='output_example_fwd.dat',status='unknown',form='formatted',iosta
t=ios)
  If( ios /= 0 ) Then
     Write(*,*) 'error opening the output file ios= ',ios
     Stop
  Endif

  !================================================
  !========== Read profile == start ==============
  Open(iup, file='prof.dat',status='old',iostat=ios)
  If( ios /= 0 ) Then
     Write(*,*) 'error opening profile file ios= ',ios
     Stop
  Endif

  ! Do allocation of input profile arrays with the number of levels.
  asw = 1 ! allocate
  allocate( profiles(nprof),stat= alloc_status(1))
  profiles(1) % nlevels = nlevels
  call rttov_alloc_prof(         &
      & errorstatus,             &
      & nprof,                   &
      & profiles,                &
      & nlevels,                 &
      & opts,                    &
      & asw,                     &
      & coefs = coefs(nrttovid), &
      & init = .true._jplm       )
  If( errorstatus /= errorstatus_success ) Then
     errorstatus = errorstatus_fatal
     Write( errMessage, '( "mem allocation error for profile arrays")' )
     Call rttov_errorreport (errorstatus, errMessage, NameOfRoutine)
     Stop
  Endif

  ! Read pressure (hPa), temp (K), WV (ppmv), O3 (ppmv)
  ! take care of doing the unit conversions to
  ! hPa, K and ppmv
  Read(iup,*) profiles(1) % p(:)
  Read(iup,*) profiles(1) % t(:)
  Read(iup,*) profiles(1) % q(:)
  Read(iup,*) profiles(1) % o3(:)

  ! 2 meter air variables
  Read(iup,*) profiles(1) % s2m % t ,&
        & profiles(1) % s2m % q ,&
        & profiles(1) % s2m % p ,&
```

```
      & profiles(1) % s2m % u ,&
      & profiles(1) % s2m % v

! Skin variables
Read(iup,*) profiles(1) % skin % t ,&
      & profiles(1) % skin % fastem

! Cloud variables
Read(iup,*) profiles(1) % ctp,&
      & profiles(1) % cfraction
Close(iup)

! Other variables from interactive inputs
profiles(1) % zenangle    = zenith
profiles(1) % azangle     = azimut
profiles(1) % latitude    = lat
profiles(1) % elevation   = zerht
profiles(1) % sunzenangle = sunzang
profiles(1) % sunazangle  = sunazang
! surface type
profiles(1) % skin % surftype  = isurf
profiles(1) % skin % watertype = nwater
profiles(1) % idg         = 0._jprb
profiles(1) % ish         = 0._jprb
if( opts % addclouds ) then
  profiles(1) % cloud(:,:) = 0._jprb
  profiles(1) % cfrac(:,:) = 0._jprb
  profiles(1) % icede(:) = 0._jprb
endif
!========== Read profile == end ==============
!=============================================
! copy profile 1 to 2..nprof
If ( nprof > 1_jpim ) Then
  Do iprof = 2, nprof
    Call rttov_copy_prof(      &
          & profiles(iprof:iprof),  &
          & profiles(1_jpim:1_jpim), &
          & larray=.true._jplm,    &
          & lscalar=.true._jplm)
    ! here can introduce some variability
    profiles(iprof) % skin % t = profiles(1) % skin % t + (iprof-1)
  End Do
End If

! allocate radiance results arrays with number of channels
asw = 1 ! allocate
call rttov_alloc_rad( &
    & errorstatus,    &
    & nchanprof,      &
    & radiance,       &
    & nlevels-1_jpim,      &
    & asw)
If( errorstatus /= errorstatus_success) Then
   errorstatus = errorstatus_fatal
   Write( errMessage, '( "mem allocation error for radiance arrays")' )
   Call rttov_errorreport (errorstatus, errMessage, NameOfRoutine)
   Stop
Endif
```

```
Allocate( calcemis ( nchanprof ) ,stat= alloc_status(1))
Allocate( emissivity_in ( nchanprof ) ,stat= alloc_status(2))
If( Any(alloc_status /= 0) ) Then
   errorstatus = errorstatus_fatal
   Write( errMessage, '( "mem allocation error in emissivity arrays")' )
   Call rttov_errorreport (errorstatus, errMessage, NameOfRoutine)
   Stop
End If

! allocate transmittance structure
call rttov_alloc_transmission( &
    & errorstatus,             &
    & transmission,            &
    & nlevels-1_jpim,          &
    & nchanprof,               &
    & asw,                     &
    & init = .true._jplm)
If( errorstatus /= errorstatus_success) Then
   errorstatus = errorstatus_fatal
   Write( errMessage, '( "mem allocation error for transmission arrays")' )
   Call rttov_errorreport (errorstatus, errMessage, NameOfRoutine)
   Stop
Endif

! save input values of emissivities for all calculations
! calculate emissivity where the input emissivity value is less than 0.01
emissivity_in(:) = emissivity(:)
calcemis(:) = emissivity(:) < 0.01_JPRB

! Call RTTOV forward model
call rttov_direct(          &
      & rttov_errorstatus, &! out   error flag
      & chanprof,          &! in    channel and profile index structure
      & opts,              &! in    options structure
      & profiles,          &! in    profile array
      & coefs(nrttovid),   &! in    coefficients strucutre
      & calcemis,          &! in    flag for intermal emissivity calc
      & emissivity_in,     &! in    input emissivities per channel
      & emissivity_out,    &! out   emissivities used by RTTOV per channel
      & transmission,      &! inout computed transmittances
      & radiance)           ! inout computed radiances

If ( rttov_errorstatus /= errorstatus_success ) Then
   Write ( 0, * ) 'rttov_direct error'
   Stop
End If

! transfer data to printing arrays
Allocate(pr_radcld(nchannels), stat= alloc_status(1))
Allocate(pr_trans(nchannels), stat= alloc_status(2))
Allocate(pr_emis(nchannels), stat= alloc_status(3))
Allocate(pr_trans_lev(nlevels,nchannels), stat= alloc_status(4))
If( Any(alloc_status /= 0) ) Then
   errorstatus = errorstatus_fatal
   Write( errMessage, '( "mem allocation error for printing arrays")' )
   Call rttov_errorreport (errorstatus, errMessage, NameOfRoutine)
   Stop
End If
```

```
  Do iprof = 1, nprof
    pr_radcld(:) = 0.0_JPRB
    pr_trans(:) = 0.0_JPRB
    pr_emis(:) = 0.0_JPRB
    pr_trans_lev(:,:) = 0.0_JPRB
    !
    joff = (iprof-1_jpim) * nchannels
    Do j = 1 , nchannels
      pr_radcld(j) = radiance % cloudy(j+joff)
      pr_trans(j)  = Transmission % tau_total(j+joff)
      pr_emis(j)   = emissivity_out(j+joff )
      Do ilev = 1 ,  nlevels
          pr_trans_lev(ilev,j) = Transmission % tau_levels(ilev,J+joff)
      Enddo
    Enddo
    !
    !     OUTPUT RESULTS
    !
    NPRINT = 1+ Int((nchannels-1)/10)
    Write(IOOUT,*)' -----------------'
    Write(IOOUT,*)' Instrument ', inst_name(instrument(3,nrttovid))
    Write(IOOUT,*)' -----------------'
    Write(IOOUT,*)' '
    Write(IOOUT,*)' Profile ',iprof
    Write(IOOUT,*)' '

    Call rttov_print_opts(opts, lu=IOOUT)
    Call rttov_print_profile(profiles(iprof), lu=IOOUT)
    WRITE(IOOUT,777) platform_name(instrument(1,nrttovid)),
instrument(2,nrttovid)

    WRITE(IOOUT,*)'CHANNELS PROCESSED:'
    WRITE(IOOUT,111) (chanprof(j) % chan, j = 1+joff,nchannels+joff)
    WRITE (IOOUT,*)' '
    Write(IOOUT,222) (radiance % bt(j), j = 1+joff,nchannels+joff)
    Write(IOOUT,*)' '
    Write(IOOUT,*)'CALCULATED RADIANCES: SAT =', instrument(2,nrttovid)
    Write(IOOUT,222) (radiance % total(j), j = 1+joff,nchannels+joff)
    Write(IOOUT,*)' '
    Write(IOOUT,*)'CALCULATED OVERCAST RADIANCES: SAT =', instrument(2,nrttovid)
    Write(IOOUT,222) (pr_radcld(j), j = 1,nchannels)
    Write (IOOUT,*)' '
    Write(IOOUT,*)'CALCULATED SURFACE TO SPACE TRANSMITTANCE: S'&
              & ,'AT =',instrument(2,nrttovid)
    Write(IOOUT,4444) (pr_trans(j), j = 1,nchannels)
    Write (IOOUT,*)' '
    Write(IOOUT,*)'CALCULATED SURFACE EMISSIVITIES '&
              & ,'SAT =',instrument(2,nrttovid)
    Write(IOOUT,444) (pr_emis(j), j = 1,nchannels)
    !
    !
    If(nchan(nprof) <= 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+joff+(NP-1)*10,Min(Int(10+joff+(NP-
1)*10,jpim),nchannels))
          Do  ILEV = 1 , NLEVELS
```

```
            Write(IOOUT,4445)ILEV,(pr_trans_lev(ilev,J),&
                    & J = 1+(NP-1)*10,Min(Int(10+(NP-1)*10,jpim),nchannels))
          End Do
          Write(IOOUT,1115) (chanprof(j+ joff) % chan,&
                  & J = 1+joff+(NP-1)*10,Min(Int(10+joff+(NP-
1)*10,jpim),nchannels))
        End Do
      Endif
    End Do
    !
    ! Deallocate arrays
    deallocate( chanprof,       stat=alloc_status(1))
    deallocate( emissivity,     stat=alloc_status(2))
    deallocate( emissivity_in,  stat=alloc_status(3))
    deallocate( emissivity_out, stat=alloc_status(4))
    deallocate( calcemis,       stat=alloc_status(5))

    ! dealloc printing arrays
    deallocate( pr_radcld,      stat=alloc_status(6))
    deallocate( pr_trans,       stat=alloc_status(7))
    deallocate( pr_emis,        stat=alloc_status(8))
    deallocate( pr_trans_lev,   stat=alloc_status(9))
    If( any(alloc_status /= 0) ) then
      errorstatus = errorstatus_fatal
      Write( errMessage, '( "mem deallocation error")' )
      Call rttov_errorreport (errorstatus, errMessage, NameOfRoutine)
      Stop
    End If

    asw = 0 ! deallocate radiance arrays
    call rttov_alloc_rad (errorstatus,nchannels,radiance,nlevels-1_jpim,asw)
    If(errorstatus /= errorstatus_success) Then
      Write( errMessage, '( "radiance deallocation error")' )
      Call rttov_errorreport (errorstatus, errMessage, NameOfRoutine)
    Endif

    asw = 0 ! deallocate transmission arrays
    call rttov_alloc_transmission (errorstatus,transmission,nlevels-
1_jpim,nchannels,asw)
    If(errorstatus /= errorstatus_success) Then
      Write( errMessage, '( "radiance deallocation error")' )
      Call rttov_errorreport (errorstatus, errMessage, NameOfRoutine)
    Endif

    asw = 0 ! deallocate profile arrays
    call rttov_alloc_prof (errorstatus,nprof,profiles,nlevels,opts,asw)
    deallocate(profiles,stat=alloc_status(1))
    If(errorstatus /= errorstatus_success .or. alloc_status(1) /= 0) Then
      Write( errMessage, '( "profile deallocation error")' )
      Call rttov_errorreport (errorstatus, errMessage, NameOfRoutine)
    Endif

    Call rttov_dealloc_coefs (errorstatus, coefs(nrttovid))
    If(errorstatus /= errorstatus_success) Then
      Write( errMessage, '( "coefs deallocation error")' )
      Call rttov_errorreport (errorstatus, errMessage, NameOfRoutine)
    Endif
    Deallocate(coefs,stat=alloc_status(1))
    If(errorstatus /= errorstatus_success .or. alloc_status(1) /= 0) Then
```

```
    Write( errMessage, '( "coefficients array deallocation error")' )
    Call rttov_errorreport (errorstatus, errMessage, NameOfRoutine)
  Endif

  !Close output file
  Close(IOOUT,iostat=ios)
  If( ios /= 0 ) Then
    Write(*,*) 'error closing the output file ios= ',ios
    Stop
  Endif

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(/,'CALCULATED BRIGHTNESS TEMPERATURES: SAT = ',A8,I3 )

End Program example_fwd
```

End of User Guide