NWP SAF

Satellite Application Facility for Numerical Weather Prediction

Document NWPSAF-MO-VS-022

Version 1.4

29 September 2006

Incorporation of RTTOV-8 in the JCSDA CRTM

P.F.W. van Delst
Joint Center for Satellite Data Assimilation

and

R.W. Saunders
Met Office, UK

EUMETSAT Met Office ECMWF KNMI METEO FRANCE Toujours un temps d'avance

NWP SAF Visiting Scientist Report

Incorporation of RTTOV-8 in the JCSDA CRTM

P.F.W. van Delst[1] and R.W. Saunders[2]

[1]Joint Center for Satellite Data Assimilation
NOAA/NWS/NCEP/EMC
Camp Springs, MD 20746
USA

[2]Met Office,
Fitzroy Rd
Exeter EX1 3PB
U.K.

This documentation 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 Météo France.

Change record			
Version	Date	Author / changed by	Remarks
1.0	13/09/2006	P. van Delst	Initial draft
1.1	22/09/2006	R. Saunders	Comments made
1.2	22/09/2006	P. van Delst	Comment addressed. Recommendations for further work updated. Other minor updates.
1.3	26/09/2006	P. van Delst	Added RTTOV AtmAbsorption flowchart.
1.4	29/09/2006	P. van Delst	Updated Jacobian plots.

Incorporation of RTTOV-8 in the JCSDA CRTM

1. Introduction

Simulation of atmospheric radiative transfer involves a number of physical processes. The Community Radiative Transfer Model (CRTM) is based upon a framework that is intended to allow independent development of algorithms to model these different processes. The components of atmospheric radiative transfer considered by the CRTM can be loosely divided into four main categories,

1. Absorption of radiation by the gaseous constituents of the atmosphere,
2. Absorption and scattering of radiation by clouds and aerosols,
3. Surface emission of radiation and surface interaction with downwelling atmospheric radiation, and
4. Solution of the radiative transfer equation.

The CRTM framework was designed to allow for a relatively natural division of the software implementation of the above categories into modular entities so that as new or updated algorithms are developed, they can be easily integrated. This work focuses on modification of the atmospheric absorption component of the CRTM (hereafter referred to as AtmAbsorption), currently an implementation of OPTRAN, to incorporate the RTTOV equivalent into the CRTM framework. The other CRTM components – scattering, surface optics, and the radiative transfer itself – remain unchanged. The modifications are applied to all the CRTM forward(FWD), tangent-linear(TL), adjoint(AD), and K-matrix(K) models.

2. Software Modifications

The first goal was to continue the work begun by Roger Saunders in his visit to the JCSDA and get the forward model component working. His report is available as a NWPSAF visiting scientist report. However, to facilitate the later modifications for the TL, AD, and K models, it was decided to modify the CRTM interface itself to allow for sensor-based, rather than channel-based, processing.

2.1 CRTM code changes

Channel- to Sensor-based processing

For historical reasons, to compute satellite sensor radiances or Jacobians the current operational CRTM does not distinguish between channels from different sensors. Prior to use in the CRTM, the spectral (SpcCoeff) and optical depth (TauCoeff) coefficient data for each required sensor are pre-processed to concatenate all the individual sensor datafiles into one datafile per coefficient type (see Figure 1a). These datafiles are then used to initialise the CRTM by loading the concatenated SpcCoeff and TauCoeff data into the scalar shared data structures `SC` and `TC` (see Figure 1b) which are used internally in the CRTM to store these coefficient data. The various CRTM models then simply loop over the number of channels without distinguishing to which sensor any particular channel belongs. This is hereafter referred to as channel-based processing.

To aid in the incorporation of other AtmAbsorption algorithms in the CRTM (not just RTTOV's algorithm) the initialisation phase has been modified to use sensor-based processing. This eliminates the need to concatenate the separate sensor SpcCoeff and TauCoeff files prior to initialisation – the various datafiles can be read in as is (see Figure 2). Several significant changes have been made to the CRTM framework to implement the sensor-based processing,

- Spectral (SpcCoeff) and optical depth (TauCoeff) shared data coefficient structures `SC` and `TC` were changed from scalar structures to rank-1 (of size `nSensors`) structure arrays.
- The CRTM initialisation function was modified to accept a list of sensors and from that list determine the required SpcCoeff and TauCoeff datafiles to read and load into their respective shared data structure arrays. The output `ChannelInfo` structure – which contains indexing information for specific sensors/channels into the various shared data structure arrays – was changed from a scalar argument to rank-1 (of size `nSensors`).

- The main CRTM model functions (forward, tangent-linear, adjoint, and K-matrix) were modified
    - The interfaces were altered to accept the rank-1 `ChannelInfo` output from the initialisation function.
    - The internal channel loop was altered to loop over sensors and within that to loop over the channels for that sensor only.
    - The call interface to every internal routine interface was altered to accept both a sensor index (an index into a particular element of the shared data coefficient structure arrays `SC` and `TC`) and a channel index (an index into the component channel data within that structure element, e.g. `SC(`*sensor index*`)%Frequency(`*channel index*`)`).
- All the internal routine interfaces (for AtmAbsorption, CloudScatter, AerosolScatter, SfcOptics, and RTSolution computations) and code itself were modified to accept both a sensor index and channel index.

These generic changes to the trunk CRTM code made the subsequent integration of RTTOV into the CRTM framework much easier to implement. The RTTOV specific CRTM code changes are discussed in the next section.

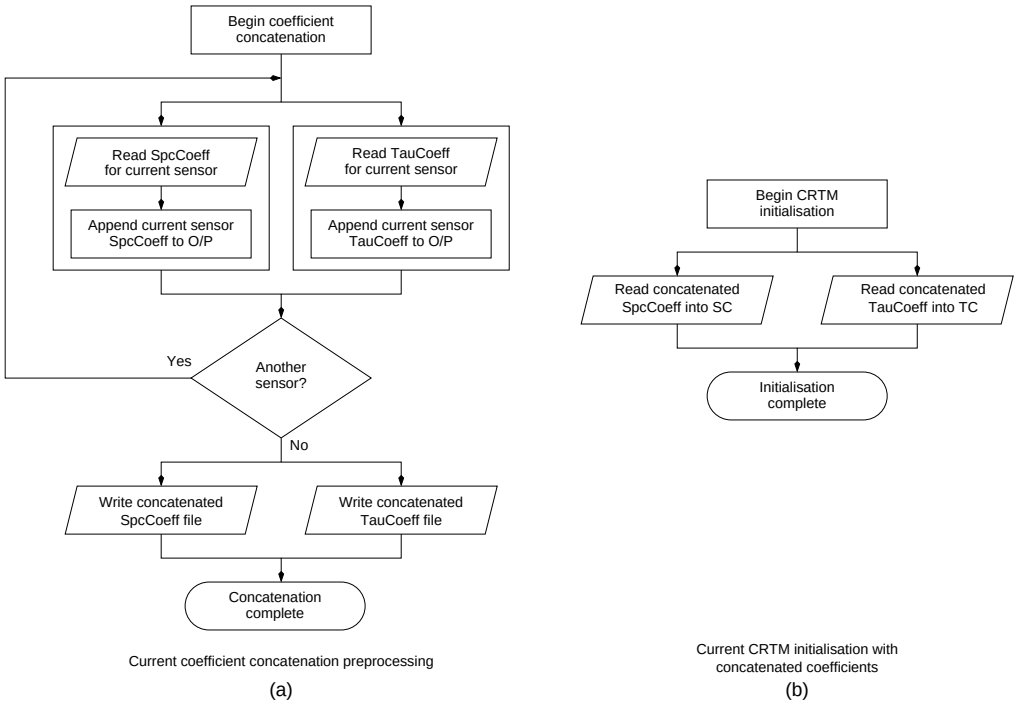


Figure 1. Current CRTM initialisation methodology for channel-based processing. **(a)** Separate sensor SpcCoeff and TauCoeff datafiles are concatenated into single SpcCoeff and TauCoeff datafiles. **(b)** The concatenated datafiles are used to initialise the CRTM.

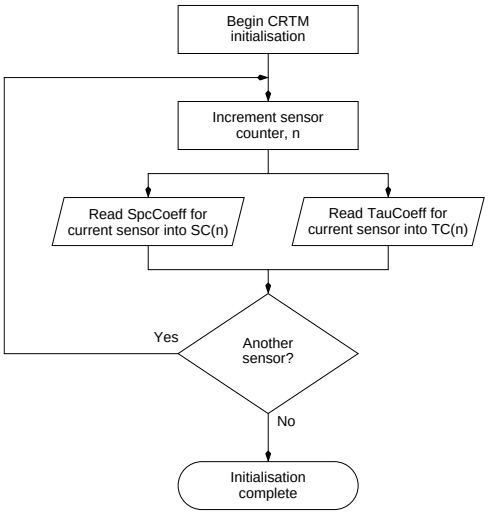


Figure 2. New CRTM initialisation methodology for sensor-based processing.

RTTOV specific CRTM source code changes

Before detailing the specific changes to the CRTM to accommodate the RTTOV AtmAbsorption algorithm, a fundamental difference between the RTTOV and CRTM design must be discussed – how the two codes process sensor channels.

The CRTM was designed to process a single channel at a time. The main reason behind this design decision was to minimise the memory footprint of the CRTM when cloud and aerosol scattering was included in the computation. For highly scattering atmospheres, the memory required to hold all the intermediate forward variables (e.g. phase functions) for use in K-matrix computations would become prohibitive for the many channels case. Thus, each frequency dependent component of the CRTM is designed for processing a single channel per call. RTTOV, however, was designed to process many channels at once. Thus, to accommodate the use of the RTTOV AtmAbsorption algorithm, the requisite calls were moved out of the channel loop. This is discussed further below.

TauCoeff loading.

The CRTM module modified to accommodate the RTTOV coefficient file reading was `CRTM_TauCoeff.f90`. This module contains the shared data TauCoeff structure holding the AtmAbsorption algorithm coefficients and was modified to call the RTTOV-specific TauCoeff load routines.

CRTM to RTTOV data mapping

A module, `RTTOV_Utility.f90`, was created to contain procedures required to map between CRTM and RTTOV specific quantities. Currently the CRTM↔RTTOV Utility procedures include

- `CRTM_to_RTTOV_SensorID`. A function to convert the CRTM sensor ID string to the RTTOV sensor ID triplet.
- `CRTM_to_RTTOV_Profile`. Functions (forward, tangent-linear, and adjoint) to convert between the CRTM input structures (Atmosphere, Surface, and GeometryInfo) and the RTTOV profile structure.

AtmAbsorption

The `CRTM_AtmAbsorption.f90` module was rewritten to accommodate the RTTOV algorithm, basically replacing the compactOPTRAN code with the calls to the various required RTTOV routines. While this rewrite was not a trivial exercise, the CRTM was designed with the idea that each component module would simply contain wrappers to provided code/algorithms. The main impediment to a "simple" integration of the RTTOV AtmAbsorption algorithm in the CRTM is the multi- versus single-channel processing issue discussed above.

The flowcharts in Figure 3 indicate the changes made to the CRTM forward model framework to accommodate the RTTOV AtmAbsorption code. The current CRTM forward model (Figure 3a) computes the AtmAbsorption predictors and then enters the sensor/channel loops computing the AtmAbsorption optical depth a channel at a time. For RTTOV integration, the AtmAbsorption predictor and all-channel optical depth computation were combined within the CRTM_AtmAbsorption.f90 module procedure and the call placed outside the sensor/channel loops.

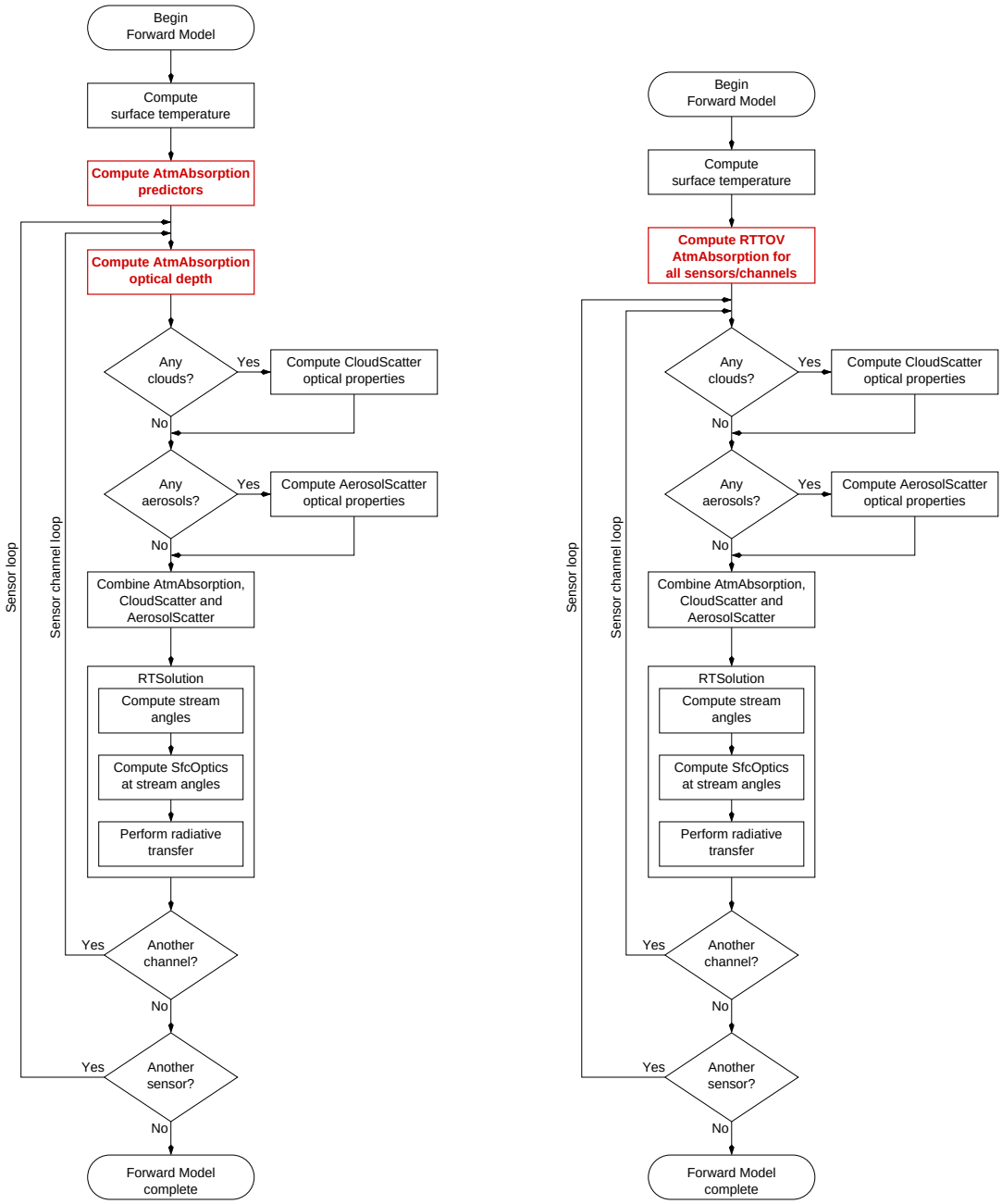


Figure 3. **(a)** Flowchart of the current CRTM Forward Model highlighting the AtmAbsorption components. **(b)** Flowchart of the CRTM Forward Model with the RTTOV AtmAbsorption. In this case all sensor/channel AtmAbsorption computations are performed outside the main sensor/channel loop.

The same procedure was followed for the tangent-linear and adjoint models also. For the K-matrix model, some additional channel looping was required in the RTTOV integration. The CRTM computes the sensor FOV surface skin temperature as an average of the temperatures for each surface

type (land, water, snow, and ice) weighted by the percentage coverage of each type in the FOV specified by the user. In the CRTM K-matrix model, the adjoint form of this averaging lies within the sensor/channel loop. However with the RTTOV AtmAbsorption because all channels are processed at once the sensor/channel loop needs to be "restarted" for the K-matrix surface temperature computation. This is shown schematically in Figure 4. A flowchart of the RTTOV library calls made in the forward and K-matrix RTTOV-based AtmAbsorption code is shown in figure 5.

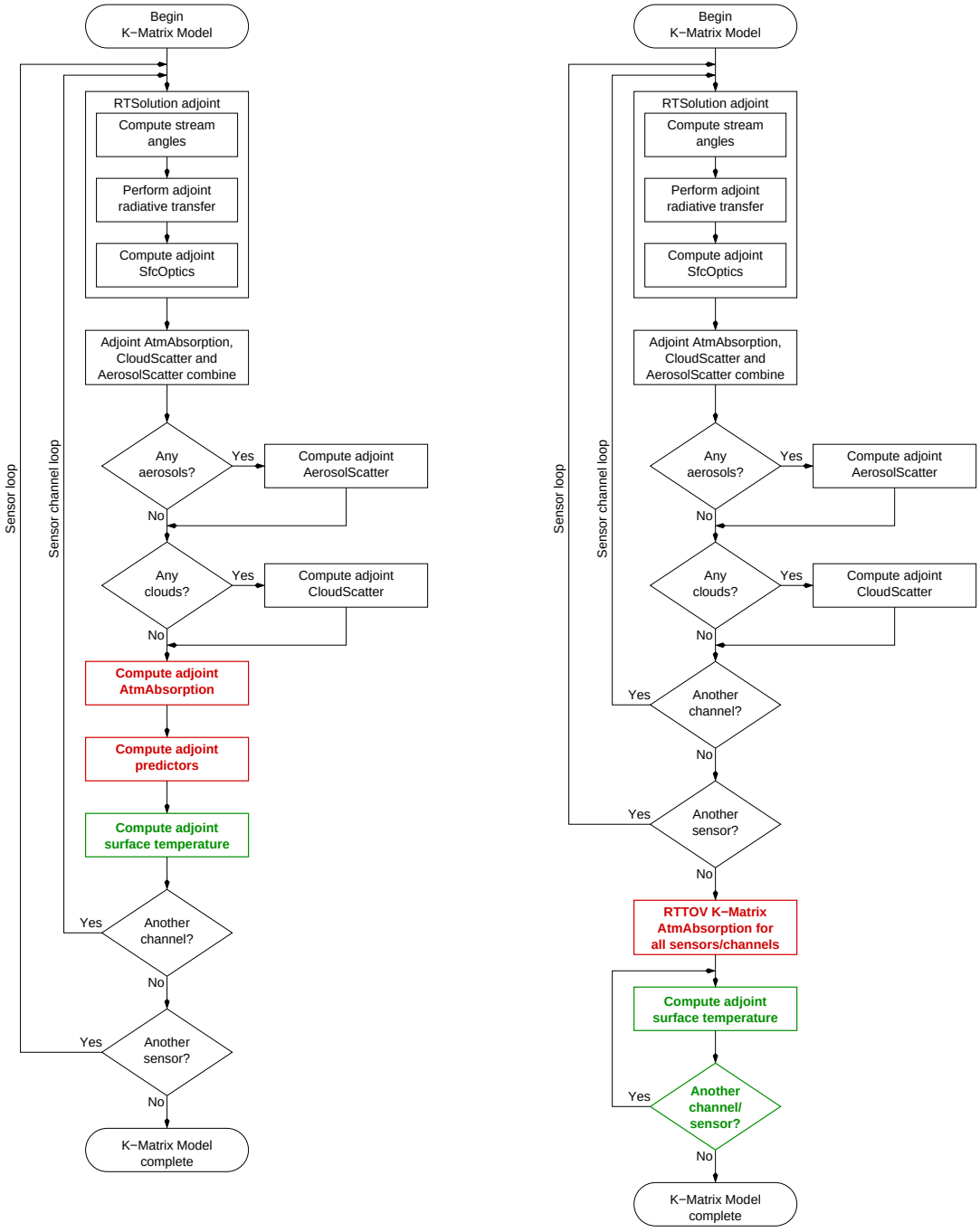


Figure 4. **(a)** Flowchart of the current CRTM K-Matrix Model indicating the AtmAbsorption and surface temperature calculation components. **(b)** Flowchart of the CRTM K-Matrix Model with the RTTOV AtmAbsorption. In this case all sensor/channel AtmAbsorption computations are performed outside the main sensor/channel loop and an additional sensor/channel loop is required for the adjoint surface temperature calculation.

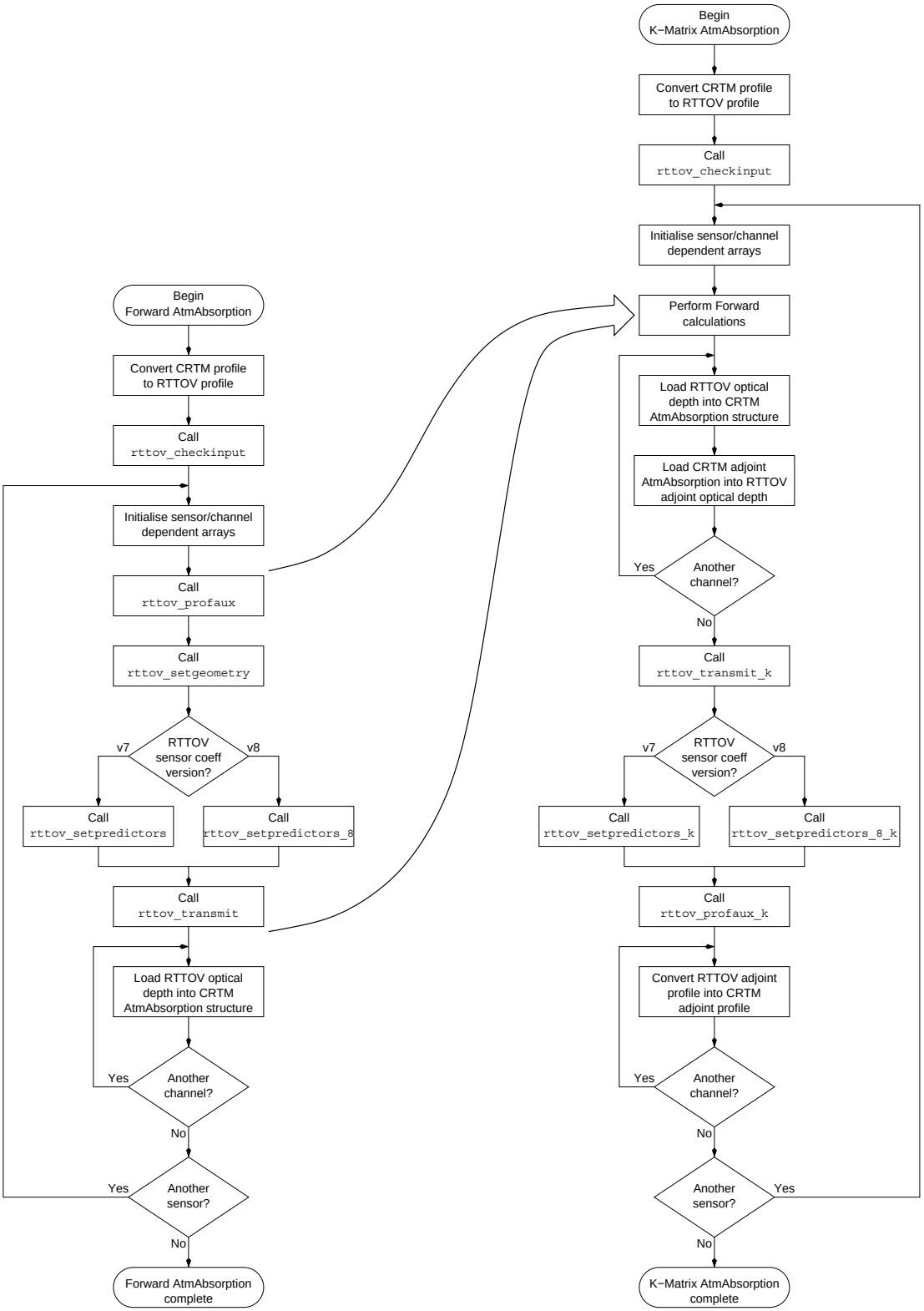


Figure 5. Flowchart of the CRTM AtmAbsorption code using with RTTOV library procedure calls. **(a)** Forward model. **(b)** K-Matrix model.

2.2 RTTOV code changes

The CRTM framework and AtmAbsorption components were designed for inputs of layer quantities (temperature, water vapour amount, ozone amount) that are supplied within the GDAS. RTTOV, however, requires the user to input level-based quantities. In the RTTOV predictor computation routines, these data are averaged across levels to produce layer quantities which are then used in the RTTOV fast transmittance computation. To use CRTM inputs in the RTTOV calculations, this level-averaging had to be removed. The effected RTTOV routines are,

- Forward model
  - `rttov_setpredictors.F90`
  - `rttov_setpredictors_8.F90`
- K-Matrix model
  - `rttov_setpredictors_k.F90`
  - `rttov_setpredictors_8_k.F90`

(The tangent-linear and adjoint routines have not yet been changed.)

The forward model changes simply consist of removing the atmospheric state variable averaging code and replacing it with a straight copy of the atmospheric profile inputs into local variables (see Figure 6), and the K-matrix code was generated from those changes (see Figure 7).

Averaging code…

```
t(1) = prof%t(1)
t(2:prof%nlevels) = (prof%t(1:prof%nlevels-1) + prof%t(2:prof%nlevels))/2._jprb

w(1) = prof%q(1)
w(2:prof%nlevels) = (prof%q(1:prof%nlevels-1) + prof%q(2:prof%nlevels))/2._jprb

If (prof%ozone_Data .And. coef%nozone>0) Then
  o(1) = prof%o3(1)
  o(2:prof%nlevels) = (prof%o3(1:prof%nlevels-1) + prof%o3(2:prof%nlevels))/2._jprb
End If
```

Replaced with direct copy code…

```
t(:) = prof%t(:)
w(:) = prof%q(:)
If (prof%ozone_Data .And. coef%nozone>0) Then
  o(:) = prof%o3(:)
End If
```

Figure 6. Replacement of level averaging code in `rttov_setpredictors.F90`. Similar changes for `rttov_setpredictors_8.F90`

```
Averaging K-Matrix code
If(prof%ozone_Data .And. coef%nozone>0)Then
  prof_k%o3(1:prof_k%nlevels-1) = prof_k%o3(1:prof_k%nlevels-1) +&
                                  0.5_JPRB*o_k(2:prof_k%nlevels,i)
  prof_k%o3(2:prof_k%nlevels)   = prof_k%o3(2:prof_k%nlevels) +&
          & 0.5_JPRB *o_k(2 : prof_k % nlevels, i)
  prof_k%o3(1) = prof_k%o3(1) + o_k(1,i)
End If

prof_k%q(1:prof_k%nlevels-1) = prof_k%q(1:prof_k%nlevels-1) +&
                               0.5_JPRB*w_k(2:prof_k%nlevels,i)
prof_k%q(2:prof_k%nlevels)   = prof_k%q(2:prof_k%nlevels) +&
                               0.5_JPRB*w_k(2:prof_k%nlevels,i)
prof_k%q(1) = prof_k%q(1)+w_k(1,i)

prof_k%t(1:prof_k%nlevels-1) = prof_k%t(1:prof_k%nlevels-1) +&
                               0.5_JPRB*t_k(2:prof_k%nlevels,i)
prof_k%t(2:prof_k%nlevels)   = prof_k%t(2:prof_k%nlevels) +&
                               0.5_JPRB*t_k(2:prof_k%nlevels,i)
prof_k%t(1) = prof_k%t(1) + t_k(1,i)
Replaced with direct copy K-Matrix code
If(prof%ozone_Data .And. coef%nozone>0)Then
  prof_k%o3(1:prof_k%nlevels) = o_k(1:prof_k%nlevels,i)
End If
prof_k%q(1:prof_k%nlevels) = w_k(1:prof_k%nlevels,i)
prof_k%t(1:prof_k%nlevels) = t_k(1:prof_k%nlevels,i)
```

Figure 7. Replacement of K-matrix level averaging code in `rttov_setpredictors_k.F90`. Similar changes for `rttov_setpredictors_8_k.F90`

3. Results

3. 1 Forward model

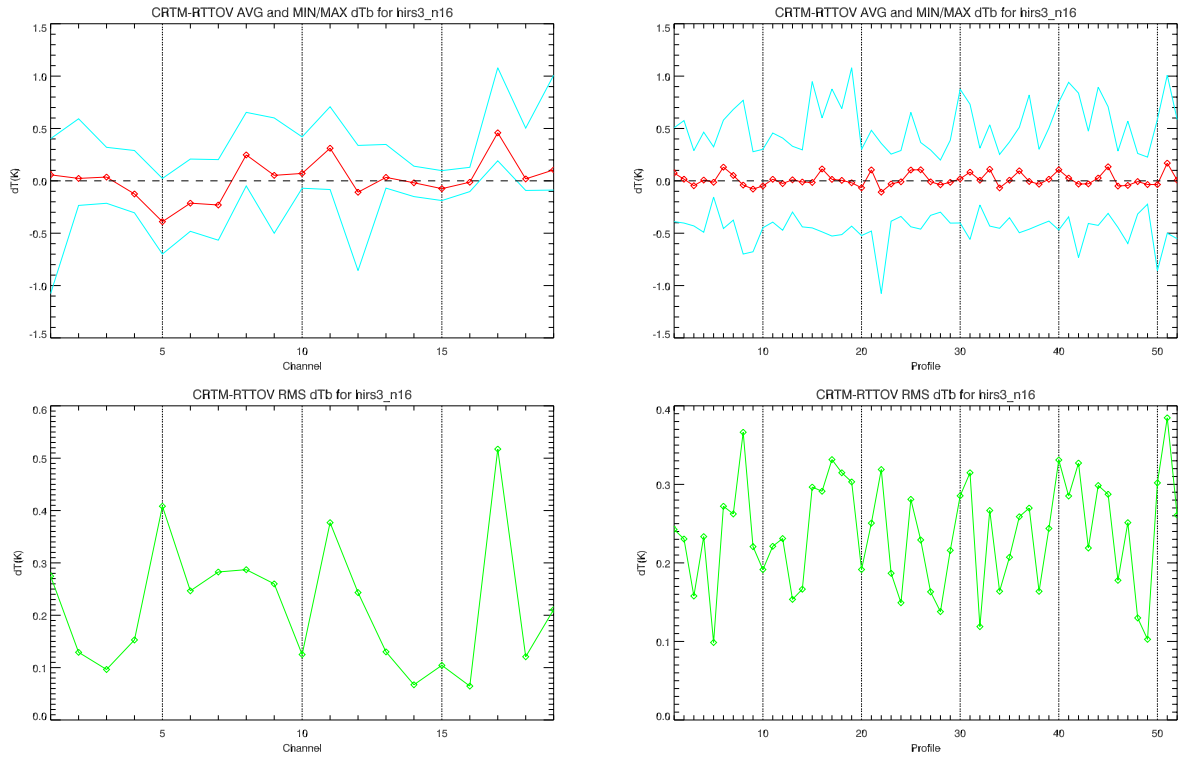
Brightness temperatures for the NOAA-16 HIRS/3 and selected channels of Aqua AIRS (see Table 1) were computed using both the CompactOPTRAN and RTTOV AtmAbsorption algorithms in the CRTM. The diverse 52-profile dataset sampled from ECMWF 60-level model fields[1] and interpolated to the AIRS 100-layers was used as input to the calculations. The RTTOV coefficients used were also at the same layering to avoid wrapping in any interpolation errors into the results. Both clear sky and cloudy calculations were performed but only the clear sky results are shown here so that any AtmAbsorption differences are not masked by cloud absorption/scattering.

AIRS channel	Frequency (cm^{-1})	AIRS channel	Frequency (cm^{-1})	AIRS channel	Frequency (cm^{-1})	AIRS channel	Frequency (cm^{-1})
71	666.7	787	917.2	1449	1330.8	1917	2229.3
77	668.2	1021	1009.2	1627	1427.1	1958	2268.7
305	737.1	1090	1040.1	1766	1544.3	1995	2305.5
453	793.1	1142	1074.3	1794	1563.5	2107	2385.9
672	871.2	1437	1323.8	1812	1576.1	2197	2500.3

Table 1. AIRS channels used for CRTM CompactOPTRAN/RTTOV integration comparisons.

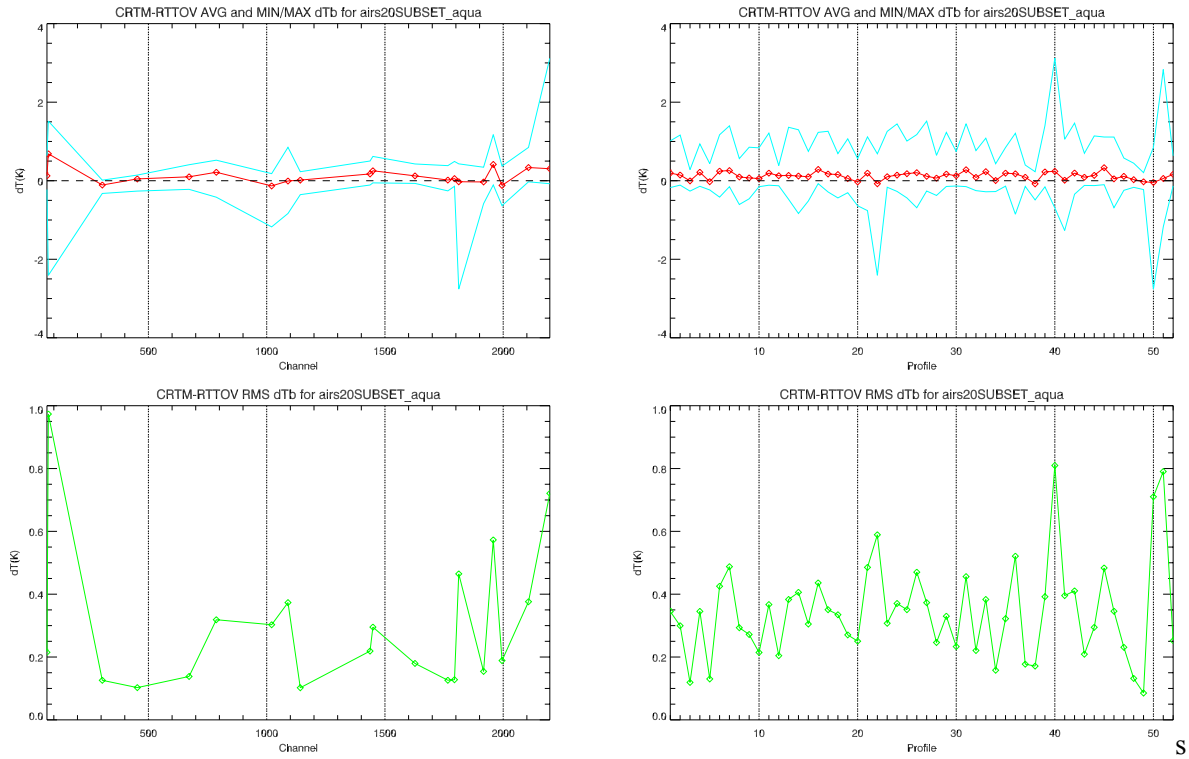
The average and RMS differences between the calculated CRTM brightness temperatures using the CompactOPTRAN and RTTOV AtmAbsorption algorithms for NOAA-16 HIRS/3 and Aqua AIRS are shown in figures 8 and 9 respectively both as a function of sensor channel and ECMWF profile. For HIRS, the average difference fluctuates about zero and the RMS differences are at the 0.2K level. For AIRS, the average difference (as well as the min/max differences) as a function of profile indicates a positive bias of about 0.1K, with the RMS differences being around the 0.3-0.4K. This result for AIRS is not entirely unexpected as the water vapour continuum component of CompactOPTRAN is not optimal in the longwave IR window regions with an approximately 0.2-0.25K RMS error between absorption lines for the dependent profile set.

[1] See Chevallier,F., Dec.2001, "Sampled databases of 60-level atmospheric profiles from the ECMWF analyses", EUMETSAT/ECMWF SAF programme, Research Report No. 4.



By HIRS Channel **By ECMWF Profile**

Figure 8. Average (top) and RMS (bottom) brightness temperature differences for the NOAA16 HIRS/3 instrument between using CompactOPTRAN and RTTOV as the AtmAbsorption algorithm in the CRTM. The left panel shows the differences averaged over all profiles as a function of channel, and the right panel shows the differences averaged over all channels as a function of the ECMWF profile.



By AIRS Channel **By ECMWF Profile**

Figure 9. Average (top) and RMS (bottom) brightness temperature differences for selected channels of the Aqua AIRS instrument between using CompactOPTRAN and RTTOV as the AtmAbsorption algorithm in the CRTM. The left panel shows the differences averaged over all profiles as a function of channel, and the right panel shows the differences averaged over all channels as a function of the ECMWF profile.

3.2 K-Matrix model

Similarly to the forward model test, the CRTM K-Matrix model was run with the CompactOPTRAN and RTTOV AtmAbsorption algorithms for both the NOAA-16 HIRS/3 and Aqua AIRS. A comparison of temperature, water vapour and ozone Jacobians for selected sensor channels and ECMWF profiles are shown in Figure 10. A quantitative comparison of the Jacobians is beyond the scope of this work, but these initial results are very encouraging. In general, the comparison of the temperature Jacobians is very good. Water vapour and ozone Jacobian comparisons are quite good for strongly absorbing channels, and poorer for weakly absorbing channels. For the stronger absorption channels, the shapes of the Jacobians tend to be well matched even if the magnitudes may not be. One interesting thing that was noticed in comparing the Jacobian profiles is that there seems to be a strong correlation between the shapes of the water vapour and ozone Jacobians. The CompactOPTRAN-based CRTM exhibits this feature more prominently, but the RTTOV-based CRTM results also indicate a similar behaviour. Also, while the CompactOPTRAN algorithm will always produce smoother Jacobians than RTTOV, it should be explicitly noted that the Jacobian comparisons shown here are relative – that is, we have not compared them to line-by-line model Jacobians so no determination should be made as to the correctness of the Jacobians based on how well behaved they may be in the vertical.

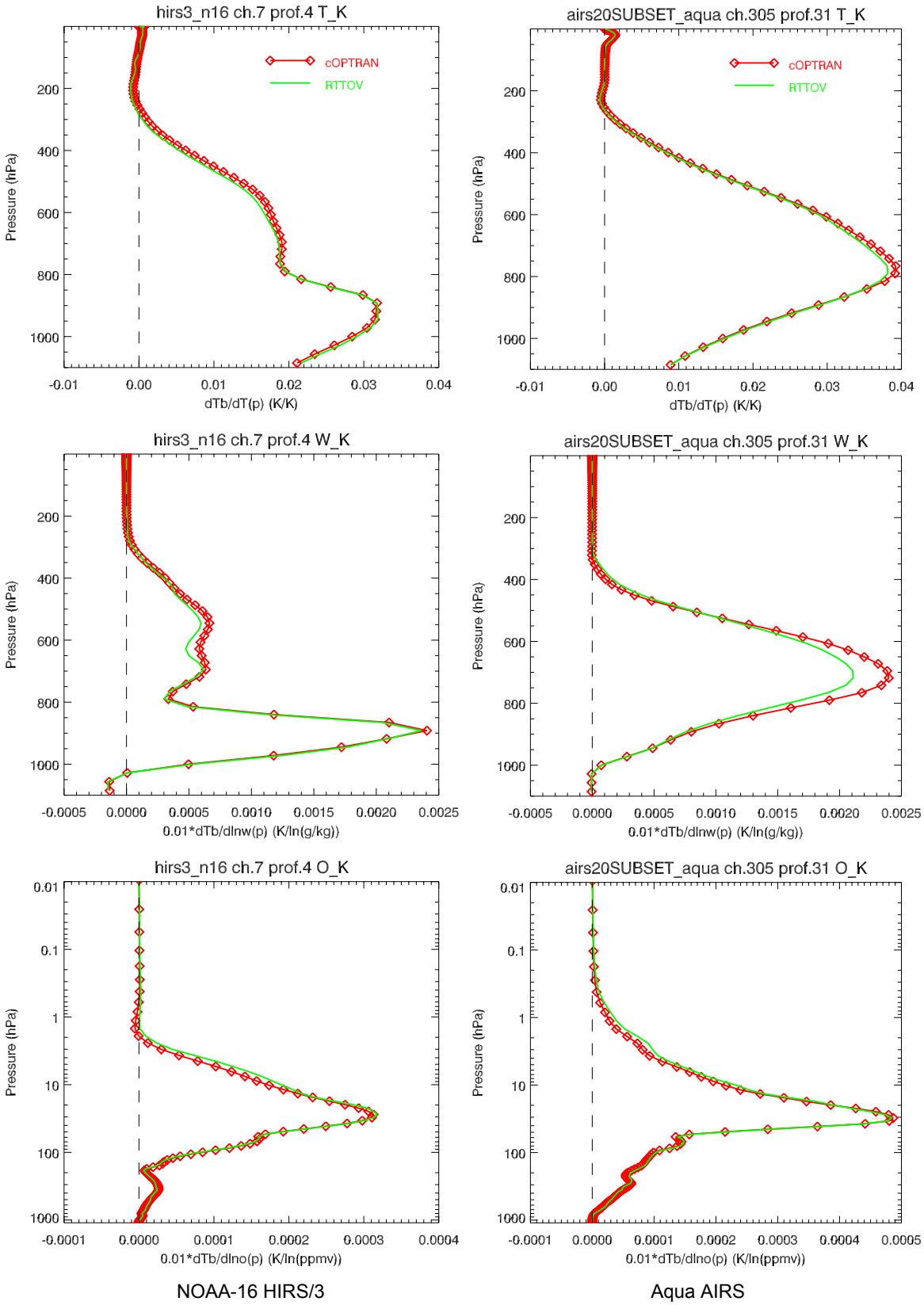


Figure 10. Temperature (top), water vapour (middle) and ozone (bottom) Jacobian profile comparisons between using CompactOPTRAN and RTTOV as the AtmAbsorption algorithm in the CRTM. The left panel of plots is for channel 7 of NOAA-16 HIRS3 for ECMWF profile #4 and the right panel of plots is for channel 305 of Aqua AIRS for ECMWF profile #31.

4. Summary of RTTOV Integration and Recommendations

The work described here was undertaken during an NWP SAF visiting scientist mission from the JCSDA to the MetOffice by Paul van Delst – a reciprocal visit to continue the work begun by Roger Saunders when he visited the JCSDA in April/May 2006. The original proposal was to integrate the tangent-linear, adjoint and K-matrix forms of RTTOV into the equivalent CRTM models and this was achieved – with, in the author's opinion, quite remarkable results for a first attempt.

One of the current goals of the CRTM is to allow simultaneous integration of different AtmAbsorption algorithms with the current emphasis on CompactOPTRAN and SARTA (the official AIRS RT model). To achieve this for the RTTOV AtmAbsorption algorithm, the previously discussed issue of how the CRTM and RTTOV process channels needs to be addressed. Due to the RTTOV requirement to process multiple channels per RTTOV call, the implementation of the RTTOV AtmAbsorption algorithm described in this report does not mesh easily with the design of the CRTM. It will also not allow for the more generic simultaneous multiple-algorithm approach of AtmAbsorption calculations within the CRTM. While RTTOV *can* process single channels at a time, it was not designed for that type of usage and the efficiency of this sort of implementation is questionable. A test of this sort of integration is recommended to determine the feasibility of the approach.

The integration of RTTOV in the CRTM provides the first opportunity to compare two current AtmAbsorption algorithms within an assimilation environment. This comparison would involve both the comparison of forward model results, i.e. comparing differences between calculation and observations for the two AtmAbsorption algorithms, but, more importantly, allow for the quantitative comparison of the impact of the different Jacobians in an assimilation system. As the K-matrix results of Figure 9 show, even though the agreement between the two approaches is quite good, there are still significant differences (in both magnitude and shape) between the Jacobians. Because the CRTM is already integrated into the GSI, this work would need to be carried out at the JCSDA and require additional resources.

5. Comments on RTTOV-87 code

The remaining recommendations relate to the RTTOV code implementation.

- Removal of array references in source code simply to document that a particular variable is an array, that is using the notation "`x(:)`" to indicate that the variable "`x`" is an array. While for most cases the use of "`x(:)`" as a euphemism for "`x`" typically has no deleterious side effects, they are *not* the same thing. The former is an array slice and the latter is not.
- Implementation of allocation and destruction procedures for the RTTOV derived data types. Rather than have a user explicitly allocate and deallocate the many components of various RTTOV structures (e.g. profile structure, auxiliary profile structure, transmittance structure, etc), helper procedures should be provided to perform this function. This will make user's code , and internal RTTOV code, much cleaner and easier to read. In addition – and more importantly – if and when structure components are added or deleted the code developer will only have to modify the helper procedures to ensure all structure components are correctly allocated and deallocation (in addition to any changes made where that structure component is actually used).
- Standardisation of argument order. It was noticed that some RTTOV routines have similar arguments (e.g. the `channel` and `polarisation` arguments) in opposite order.
- Consider the use of modules rather than separate interface bodies to define explicit interfaces for procedures. I realise this is a major change and does clash with current operational requirements (at ECMWF), but the current scenario restricts RTTOV development to only those groups who have access to the tools to automatically generate the required interface bodies. RTTOV is not that large a package that compilation cascade is an issue. In the context of use within a larger body of code this might not be true, but in that case, isn't RTTOV compiled separately into a library and linked in?
- This relates to the previous point. Definition of structures and all their helper functions (e.g. allocate, destroy, assign, test, etc) in their own modules. Currently the RTTOV data types are

all defined in a single module. Unit testing of code is much less complicated when the various components are encapsulated and loosely coupled. It also makes code maintenance more straightforward as developers modifying a particular derived data type definition (and any associated methods) will be less likely to impact other derived type definitions (and their associated methods).