Joint Center for Satellite Data Assimilation

Office Note CRTM-4

This is an unreviewed manuscript, primarily intended for informal exchange of information among JCSDA researchers

CRTM: Cloud fraction in the CRTM

Paul van Delst[a]
NCEP/EMC/IMSG

Emily Liu[b]
NCEP/EMC/SRG

Li Bi[c]
NCEP/EMC/IMSG

March 25, 2020; rev

---

[a]paul.vandelst@noaa.gov
[b]emily.liu@noaa.gov
[c]li.bi@noaa.gov

Change History

Date	Author	Change
2016-02-19	P. van Delst	Initial draft release.
2016-03-02	P. van Delst	Corrected for bug fix in averaging method plots.

DRAFT

Contents

DRAFT

List of Figures

DRAFT

1 Code Description

1.1 Overlap/cloud cover definitions

Given a cloud fraction (f_k) profile for an input atmosphere of $k = 1, \ldots, K$ layers, a cloud cover (CC) profile can be generated from the cloud fraction profile differently depending on the user-selected methodology. This cloud cover profile is then used to average the 100% clear and 100% cloudy channel radiances by selecting the last layer value, CC_K, as the total cloud cover (TCC),

$$R_{\nu,allsky} = (1 - TCC).R_{\nu,clear} + TCC.R_{\nu,cloudy} \tag{1.1}$$

Three overlap schemes are available for selection: maximum, random, and maximum/random. The formlations used in this implementation were taken from **?** as shown below.

For the maximum overlap assumption, we have

$$CC_{k,max} = \max_{i=1,k} f_i \qquad \text{for each } k = 1, \ldots, K \tag{1.2}$$

The random overlap assumption gives,

$$CC_{k,ran} = 1 - \prod_{i=1}^{k} (1 - f_i) \qquad \text{for each } k = 1, \ldots, K \tag{1.3}$$

And, finally, for the maximum-random overlap assumption, we use

$$CC_{k,maxran} = 1 - \prod_{i=1}^{k} \frac{1 - \max(f_i, f_{i-1})}{(1 - f_{i-1})} \qquad \text{for each } k = 1, \ldots, K \tag{1.4}$$

In addition to these typical overlap assumptions, the methodolgogy for computing total cloud cover based on a weighted average of the cloud water amounts in each layer [**?**] is also included, where

$$CC_{k,ave} = \frac{\sum_{i=1}^{k} \left(\sum_{n=1}^{N} q_{n,i} \right) f_i}{\sum_{i=1}^{k} \left(\sum_{n=1}^{N} q_{n,i} \right)} \qquad \text{for each } k = 1, \ldots, K \text{ where } q_{n,k} \neq 0 \tag{1.5}$$

where N represents the number of cloud *types* in a layer and $q_{n,k}$ the cloud amount for cloud type n at layer k.

1.2 Specifying the cloud fraction profile

The cloud fraction for an atmospheric profile is supplied via the new `Cloud_Fraction` component in the CRTM `Atmosphere` object definition.

Each element of the `Cloud_Fraction` component should be set to the cloud fraction, f_k, of that layer, for example

```
TYPE(CRTM_Atmosphere_type) :: atm
...
  allocate the atmosphere object...
...
atm%Cloud_Fraction        = 0.0   ! All layers are cloud free
atm%Cloud_Fraction(10:14) = 0.30  ! Layers 10-14 contain clouds, 30% fraction
```

DRAFT

1.3 Specifying the overlap/cloud cover methodology

The default cloud cover computation methodology is the averaging method, equation 1.5. A different cloud cover algorithm can be specified via the `Options` input to the CRTM.

The `Options` object definition module now includes a `CRTM_Options_SetValue()` procedure that can be used to select the cloud cover algorithm. For example, if a user wishes to use the maximum-random overlap assumption rather than the default averaging method, the option is specified like so, e.g.

```
TYPE(CRTM_Options_type) :: opt(:)
....
! Set maximum-random overlap for cloud cover
CALL CRTM_Options_SetValue( opt, Set_MaxRan_Overlap = .TRUE. )
```

See section B.1 for the complete procedure interface.

1.4 Fractional cloud coverage

Within the main CRTM procedures (forward, tangent-linear, adjoint, or K-matrix), the cloud cover computations are performed only if the supplied cloud fraction profile indicates fractional cloudiness. This determination is made by the `CRTM_Atmosphere_Coverage()` procedure given an input atmosphere object. The bulk of that procedure is replicated below.

```
USE CRTM_Parameters, WATER_CONTENT_THRESHOLD
...
! Local parameters
REAL(fp), PARAMETER :: MIN_COVERAGE_THRESHOLD = 1.0e-06_fp
REAL(fp), PARAMETER :: MAX_COVERAGE_THRESHOLD = ONE - MIN_COVERAGE_THRESHOLD
...
! Default clear
coverage_flag = CLEAR
IF ( atm%n_Clouds == 0 ) RETURN

! Check each cloud separately
Cloud_Loop: DO n = 1, atm%n_Clouds

  ! Determine if there are ANY cloudy layers
  cloudy_layer_mask = atm%Cloud(n)%Water_Content > WATER_CONTENT_THRESHOLD
  nc = COUNT(cloudy_layer_mask)
  IF ( nc == 0 ) CYCLE Cloud_Loop

  ! Get the indices of those cloudy layers
  idx(1:nc) = PACK([(k, k=1,atm%Cloud(n)%n_Layers)], cloudy_layer_mask)

  ! Check for ANY fractional coverage
  DO k = 1, nc
    IF ( (atm%Cloud_Fraction(idx(k)) > MIN_COVERAGE_THRESHOLD) .AND. &
         (atm%Cloud_Fraction(idx(k)) < MAX_COVERAGE_THRESHOLD) ) THEN
      coverage_flag = FRACTIONAL
      RETURN
    END IF
  END DO

  ! Check for ALL totally clear or totally cloudy
  IF ( ALL(atm%Cloud_Fraction(idx(1:nc)) < MIN_COVERAGE_THRESHOLD) .OR. &
       ALL(atm%Cloud_Fraction(idx(1:nc)) > MAX_COVERAGE_THRESHOLD) ) coverage_flag = OVERCAST
```

DRAFT

```
END DO Cloud_Loop
```

Note that there are two threshold checks. First, any particular cloud's water content in a layer must exceed the water content threshold (currently 10^{-6}kg/m^2) to be considered. This threshold is also used in the CRTM procedures to retrieve the cloud optical properties.

The second threshold is for the layer cloud fraction itself to specify a tolerance for what constitutes clear or cloudy coverage. In this case an arbitrary value of 10^{-6} was used.

Special mention should be made of the last check for totally clear or totally cloudy, in particular the first portion of the test, `ALL(atm%Cloud_Fraction(idx(1:nc)) < MIN_COVERAGE_THRESHOLD)`, that still leads to an overcast designation. This check for effective zero cloud coverage at that point is necessary to maintain similar behaviour for existing code using the CRTM where clouds are specified since 100% cloudiness is the default for previous versions of the CRTM.

1.5 The `CloudCover` object definition

The `CloudCover` object and method definitions are shown in detail in appendix A. A slightly truncated version is shown below to highlight differences with other CRTM-related object definitions, with the object components highlighted in blue, and the object methods in green.

```
TYPE :: CRTM_CloudCover_type
  INTEGER  :: n_Layers = 0                             ! K dimension.
  INTEGER  :: Overlap           = DEFAULT_OVERLAP_ID   ! Overlap type identifier
  REAL(fp) :: Total_Cloud_Cover = ZERO                 ! Cloud cover used in RT
  REAL(fp), ALLOCATABLE :: Cloud_Fraction(:)           ! K. The physical cloud fraction
  REAL(fp), ALLOCATABLE :: Cloud_Cover(:)              ! K. The overlap cloud cover
  TYPE(CRTM_Cloud_type), ALLOCATABLE :: Cloud(:)       ! Contains cloud information
CONTAINS
  PRIVATE
  PROCEDURE, PUBLIC, PASS(self)    :: Overlap_Id
  PROCEDURE, PUBLIC, PASS(self)    :: Overlap_Name
  PROCEDURE, PUBLIC, PASS(self)    :: Compute_CloudCover
  PROCEDURE, PUBLIC, PASS(self_TL) :: Compute_CloudCover_TL
  PROCEDURE, PUBLIC, PASS(self_AD) :: Compute_CloudCover_AD
  PROCEDURE, PUBLIC, PASS(self)    :: Is_Usable
  PROCEDURE, PUBLIC, PASS(self)    :: Destroy
  PROCEDURE, PUBLIC, PASS(self)    :: Create
  PROCEDURE, PUBLIC, PASS(self)    :: Inspect
  PROCEDURE, PUBLIC, PASS(self)    :: Set_To_Zero
  GENERIC, PUBLIC :: OPERATOR(==) => Equal
  GENERIC, PUBLIC :: OPERATOR(/=) => NotEqual
  GENERIC, PUBLIC :: OPERATOR(.Compare.) => Compare
END TYPE CRTM_CloudCover_type
```

The definition makes use of some of the object oriented features of Fortran2003/2008 by specifying procedures as part of the object definition (commonly referred to as type-bound procedures).

This simplifies the usage of the `CRTM_CloudCover_Define` module since now only the type needs to be referenced, e.g.

```
USE CRTM_CloudCover_Define, ONLY: CRTM_CloudCover_type
```

while still providing access to all of the procedures defined within.

It also changes the calling interface from a more "CRTM-like" call of

DRAFT

```
TYPE(CRTM_CloudCover_type) :: cc_obj
...
CALL CRTM_CloudCover_Inspect( cc_obj )
```

to

```
TYPE(CRTM_CloudCover_type) :: cc_obj
...
CALL cc_obj%Inspect()
```

This is mentioned here because the next major version release of the CRTM (v3) is planned to include the community Surface Emissivity Model (CSEM) and the CRTM objects visible in teh main procedure interfaces will be modified in a similar form to allow type-extension of CSEM objects.

DRAFT

2 Code Testing

2.1 Simple test

To verify the implementation of the various overlap assumptions, the simple example shown in figure 1 of **?**, here shown in figure 2.1, was used. The result of the computation is shown in figure 2.2.

Figure 2.1: Simple example from figure 1 of Hogan and Illingworth [2000].

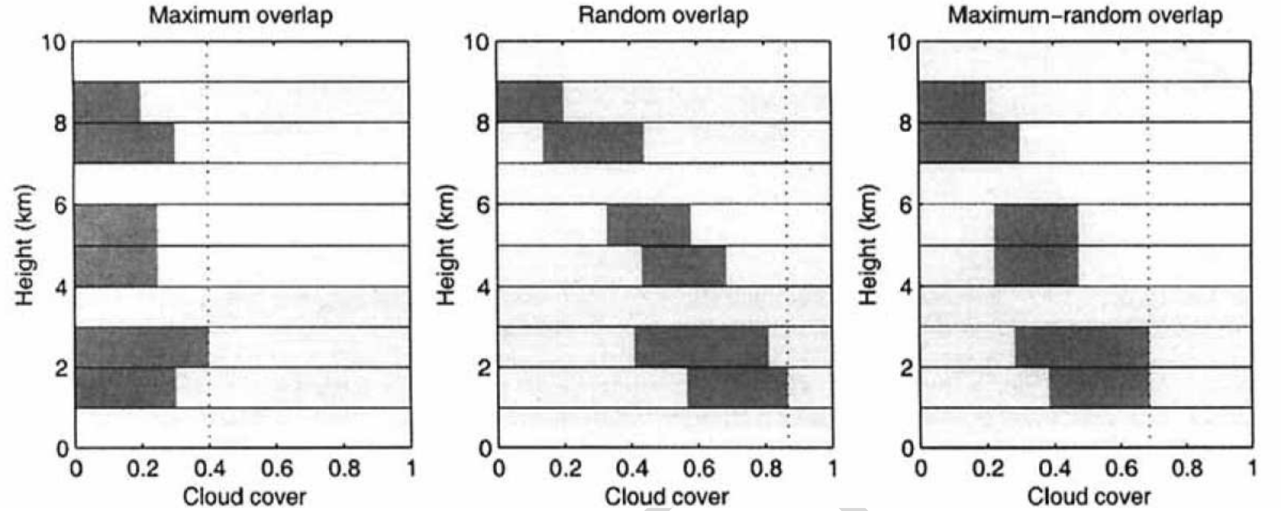
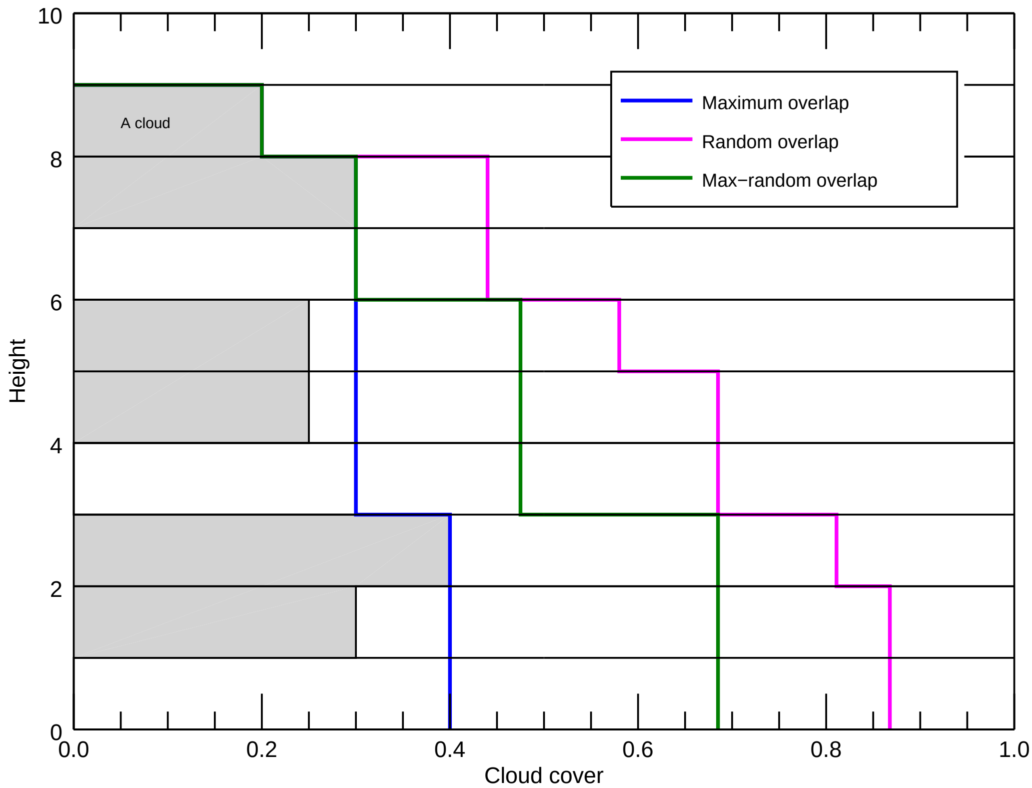


Figure 2.2: Result of the cloud overlap assumptions for comparison with the figure 1 schematic in Hogan and Illingworth [2000].

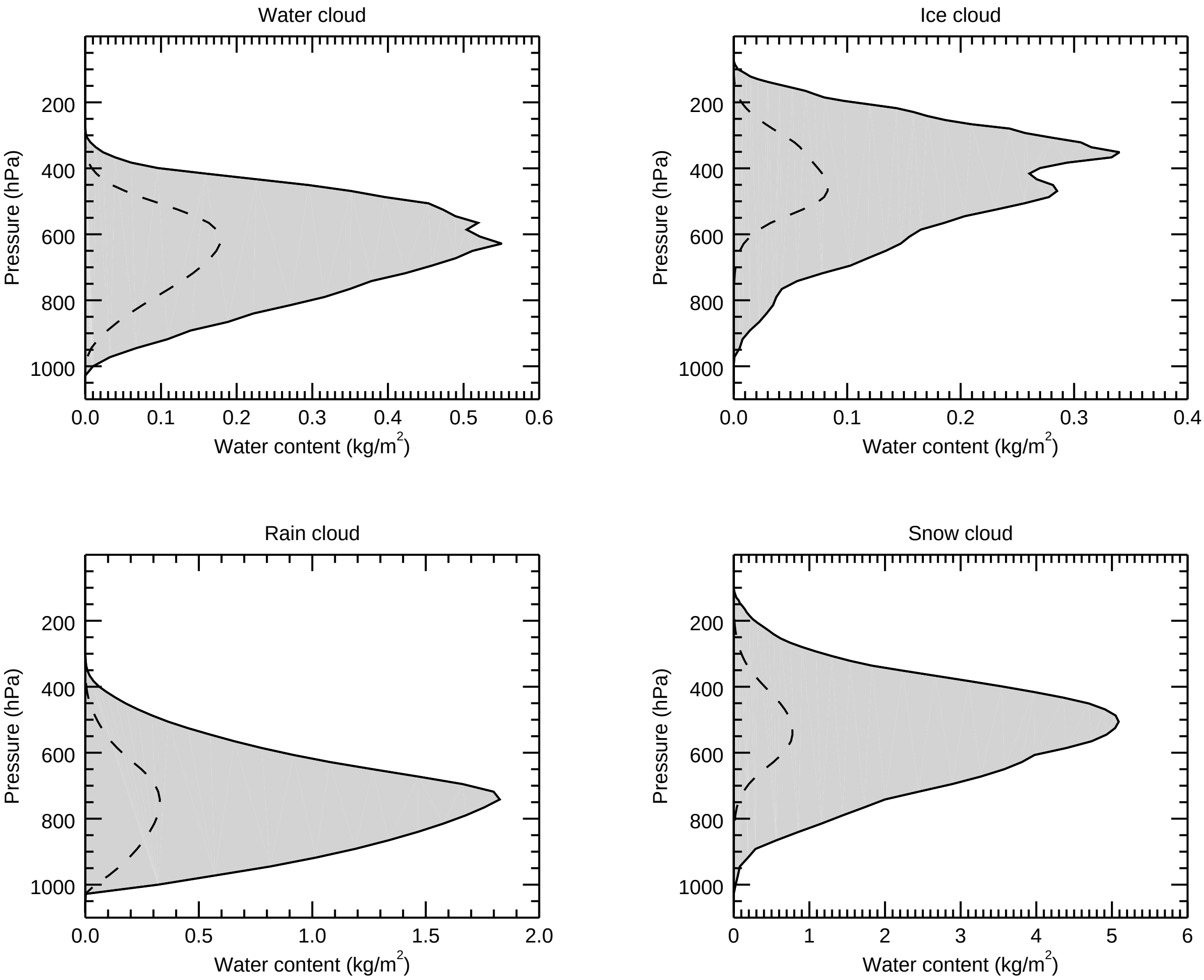


The final, near-surface cloud cover value is equivalent to the total cloud cover (TCC). As expected, the maximum overlap assumption results in a minimum cloud cover, the random overlap assumption maximises the cloud cover, and the maximum-random assumption falls between those two extremes.

2.2 Testing using ECMWF5K profile set

To more fully exercise test the cloud fraction capability, the ECMWF 5K profile set [Chevallier et al., 2006] was used. The dataset individual cloud water content spans are shown in figure 2.3.

Figure 2.3: The cloud water content span of the ECMWF 5K profile set. The dashed line is the average. Minimum value is zero.



The ECMWF5K dataset does not contain cloud fraction or a variable cloud particle effective radius for the various cloud types.

A cloud fraction profile was generated for each cloud type by scaling a random number, $A \in [0, 1)$, by the fractional cloud water content, q. So, for the n^{th} cloud in the k^{th} atmospheric layer,

$$f_{n,k} = \begin{cases} \frac{A}{c} \frac{q_{n,k}}{\max(\mathbf{q}_n)} & \text{if } q_{n,k} > 10^{-6} \\ 0 & \text{otherwise} \end{cases} \tag{2.1}$$

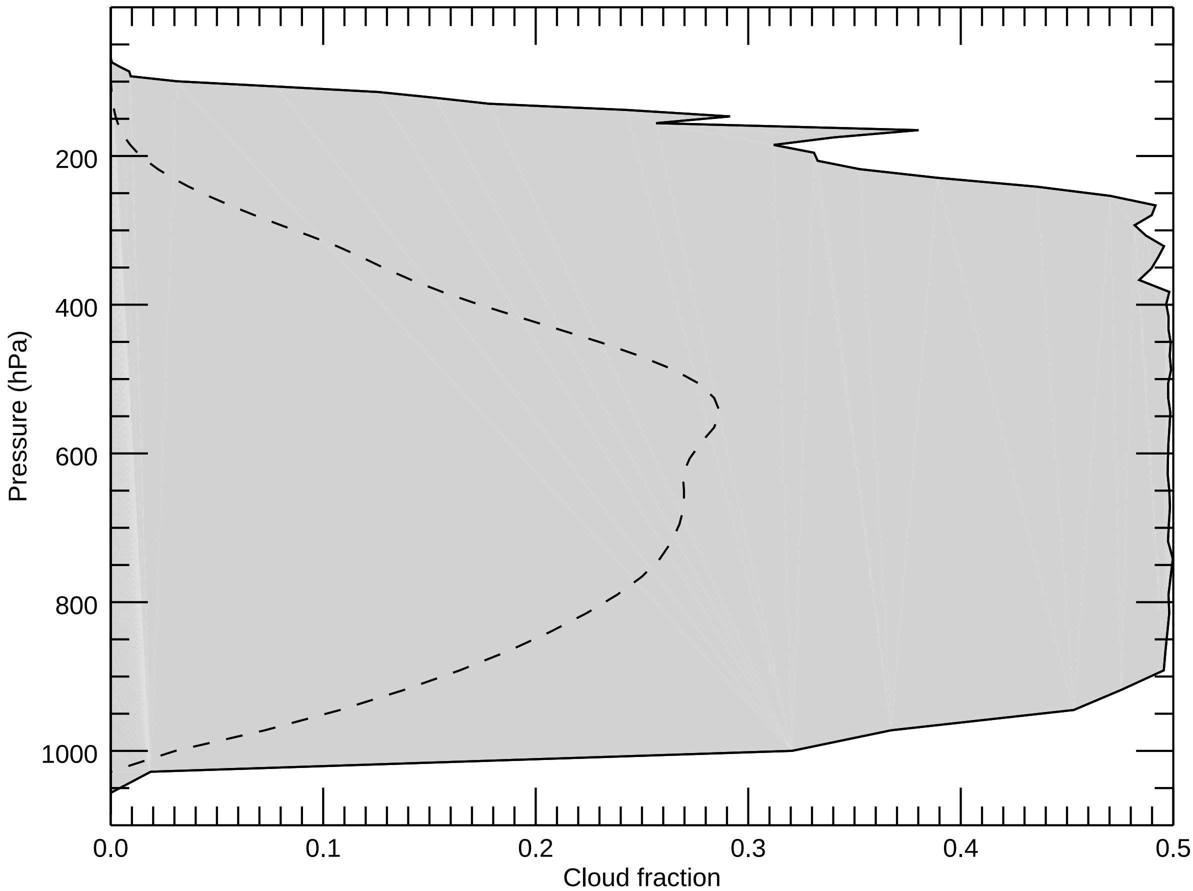
where c is an additional variable used to control the maximum cloud fraction in any profile. Values of 2 and 5 were used in testing.

The total cloud fraction in the k^{th} layer, f_k, was then simply set to the maximum value of the individual cloud fractions in that layer,

$$f_k = \max_{n=1,N} f_{n,k} \tag{2.2}$$

Figure 2.4 shows the resulting span of generating cloud fraction profiles for the set.

Figure 2.4: The cloud fraction span generated for the ECMWF 5K profile set. The dashed line is the average. Minimum value is zero.



Similarly for the cloud effective radius. A fixed "reference" value, R_{eff}, was assigned for each cloud type, n. To generate a variable effective radius for each layer, this reference value was scaled by the individual cloud water contents, q_n, at each layer, k,

$$r_{n,k} = \begin{cases} R_{eff,n}\dfrac{q_{n,k}}{\max(\mathbf{q_n})} & \text{if } q_{n,k} > 10^{-6} \\ 0 & \text{otherwise} \end{cases} \tag{2.3}$$

This gave the spread of effective radii for the dataset cloud types shown in figure 2.5.

The fractional cloudy atmosphere data was then fed into the CRTM, once for each overlap/cloud cover methodology implemented. A selection of some generated cloud cover profiles for maximum cloud fractions of 0.5 and 0.2 (for $c = 2$ and $c = 5$ respectively per equation 2.1) is shown in figure 2.6.

Figure 2.5: The effective radius span generated for the ECMWF 5K profile set. The dashed line is the average. Minimum value is zero.

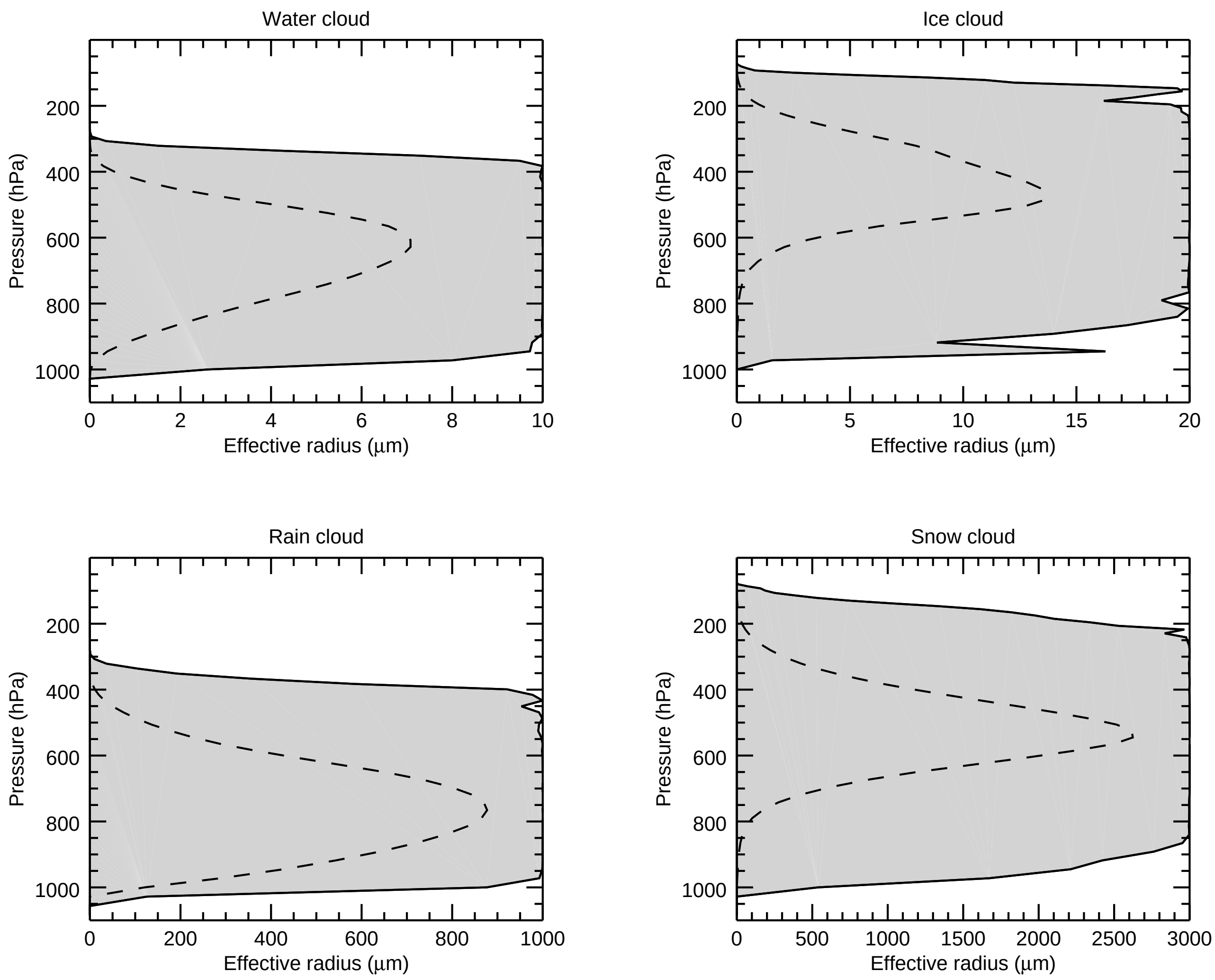
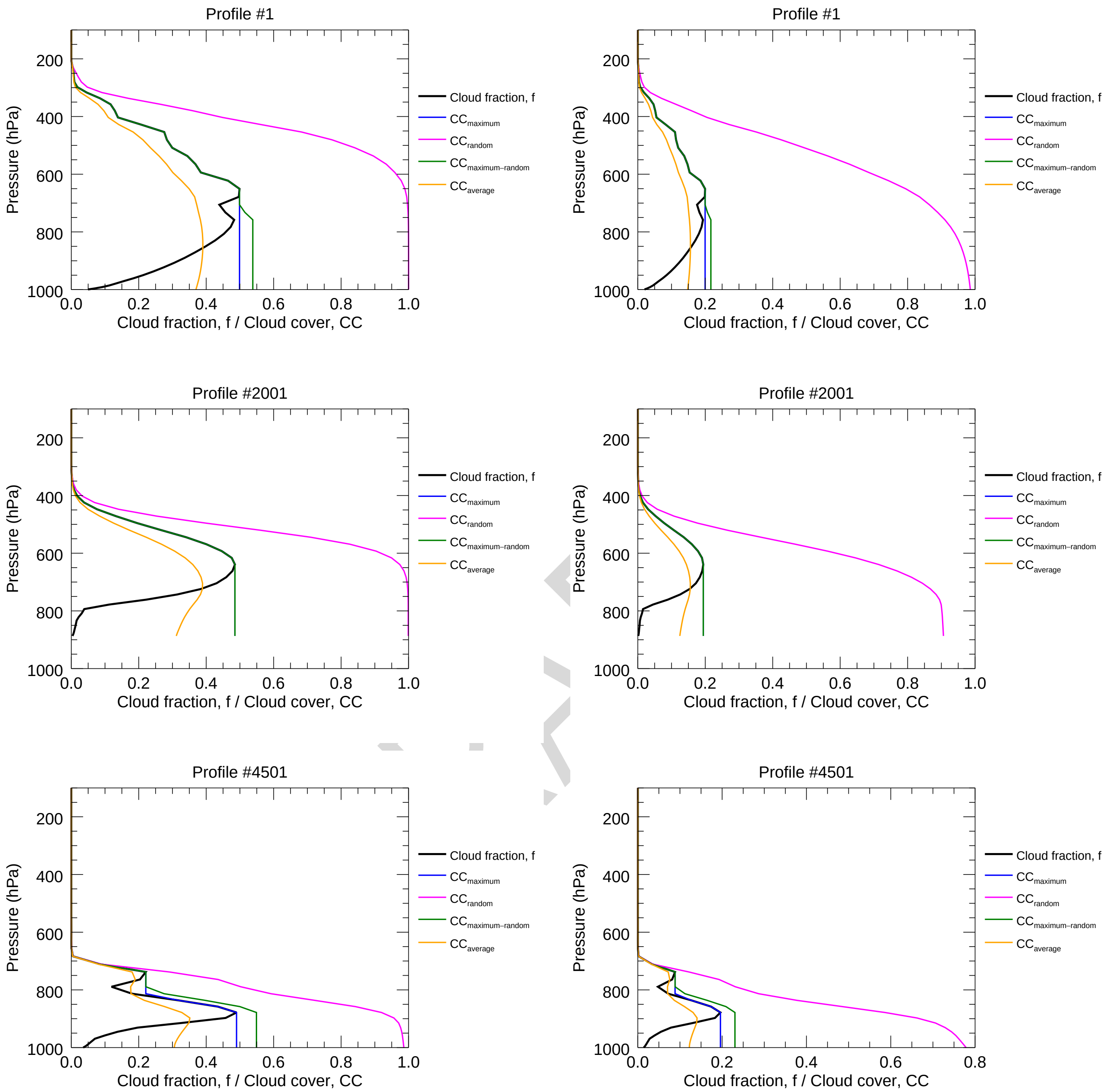
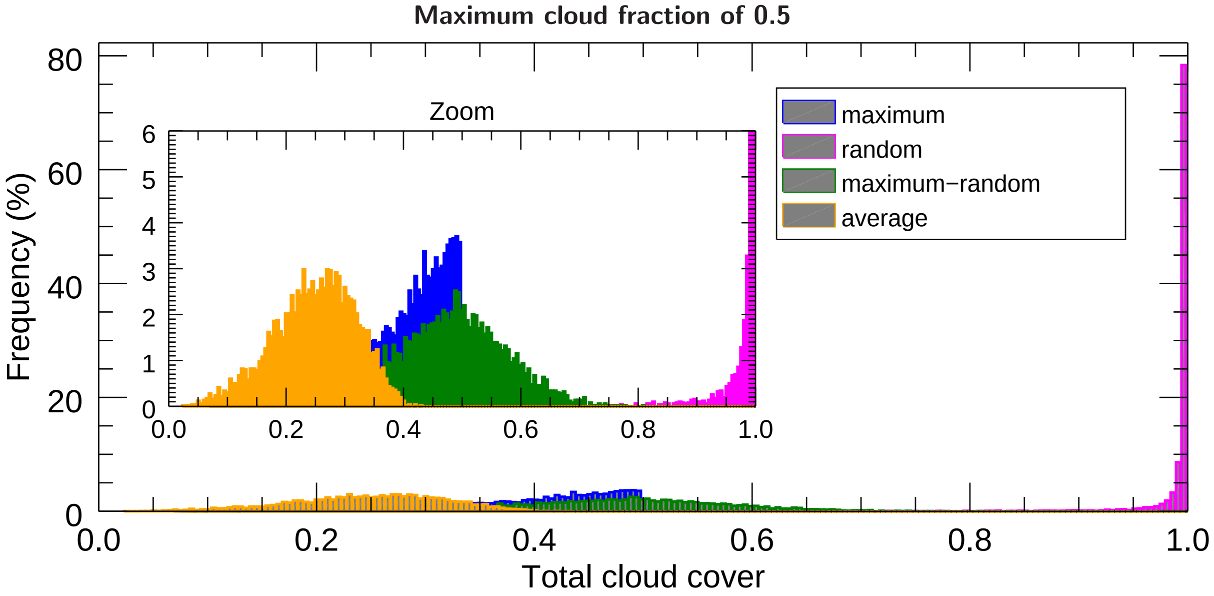


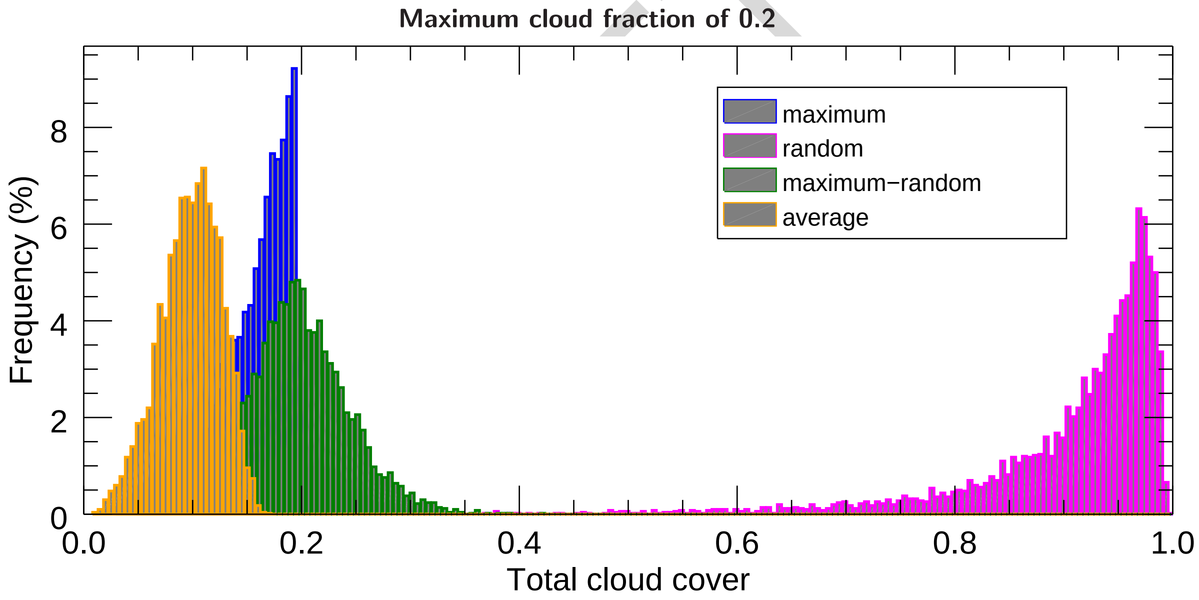
Figure 2.6: Selection of computed cloud cover profiles from the ECMWF5K profile set for two different maximum cloud fractions. **(Left)** Maximum cloud fraction of 0.5. **(Right)** Maximum cloud fraction of 0.2.



The frequency distributions of the computed cloud cover for the different methodologies are shown in figure 2.7 for the maximum cloud fractions of 0.5 (cloudier) and 0.2 (less cloudy). The random overlap assumption consistently yields a much higher total cloud cover, in most cases producing the equivalent of an overcast profile.

Figure 2.7: Frequency distribution of total cloud cover for the different overlap/cloud cover assumptions using the ECMWF5K set. **(Top)** Maximum cloud fraction of 0.5. **(Bottom)** Maximum cloud fraction of 0.2.

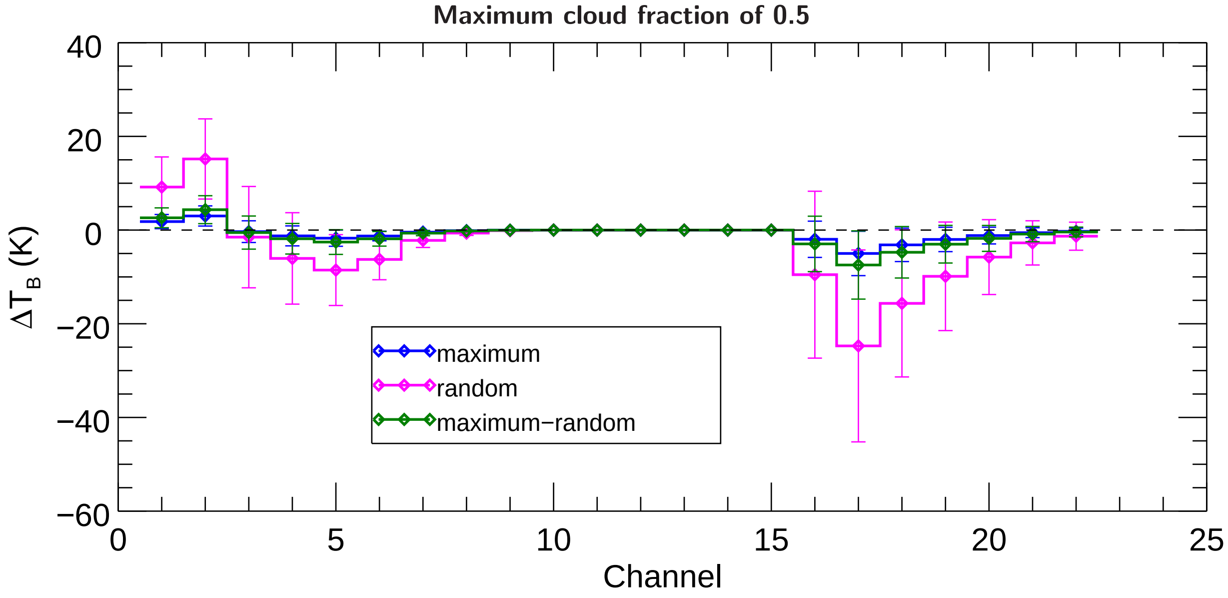




To get an idea of the relative impacts of the various cloud cover methodologies, the ECMWF5K profile set was used to compute brightness temperatures for the NPP ATMS and CrIS(399) sensors for each method, and the results for the overlap assumption methods (maximum, random, maximum-random) differenced from the averaging approach of **?**. Two runs were made, using maximum cloud fractions of 0.5 and 0.2.

The brightness temperature residuals for the ATMS and CrIS are shown in figures 2.8 and 2.9 respectively. The most evident result is that the random overlap assumption produces brightness temperatures quite different from the other methods, as well as being relatively insensitive to the maximum cloud fraction in a profile (due to the cloud cover quickly "saturating"). The other methods behave as expected with the residuals decreasing roughly similar with the maximum cloud fraction.

Figure 2.8: NPP ATMS average brightness temperature differences for cloud cover methods compared to the averaging approach using the ECMWF5K profile set. The error bars represent $\pm$ one standard deviation. **(Top)** Maximum cloud fraction of 0.5. **(Bottom)** Maximum cloud fraction of 0.2.



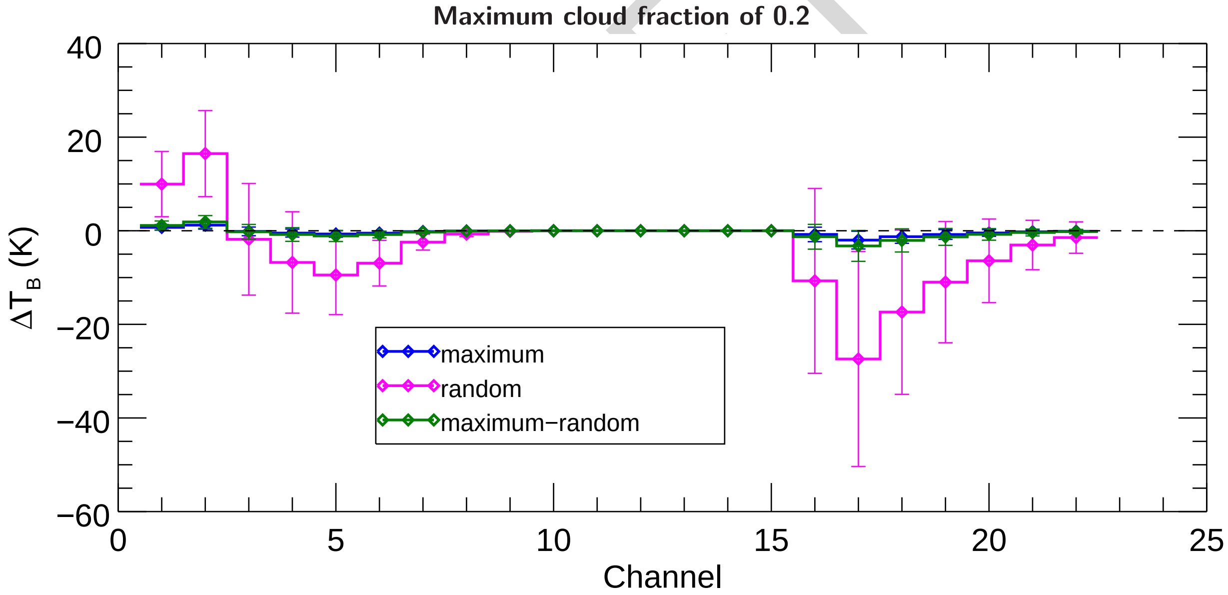
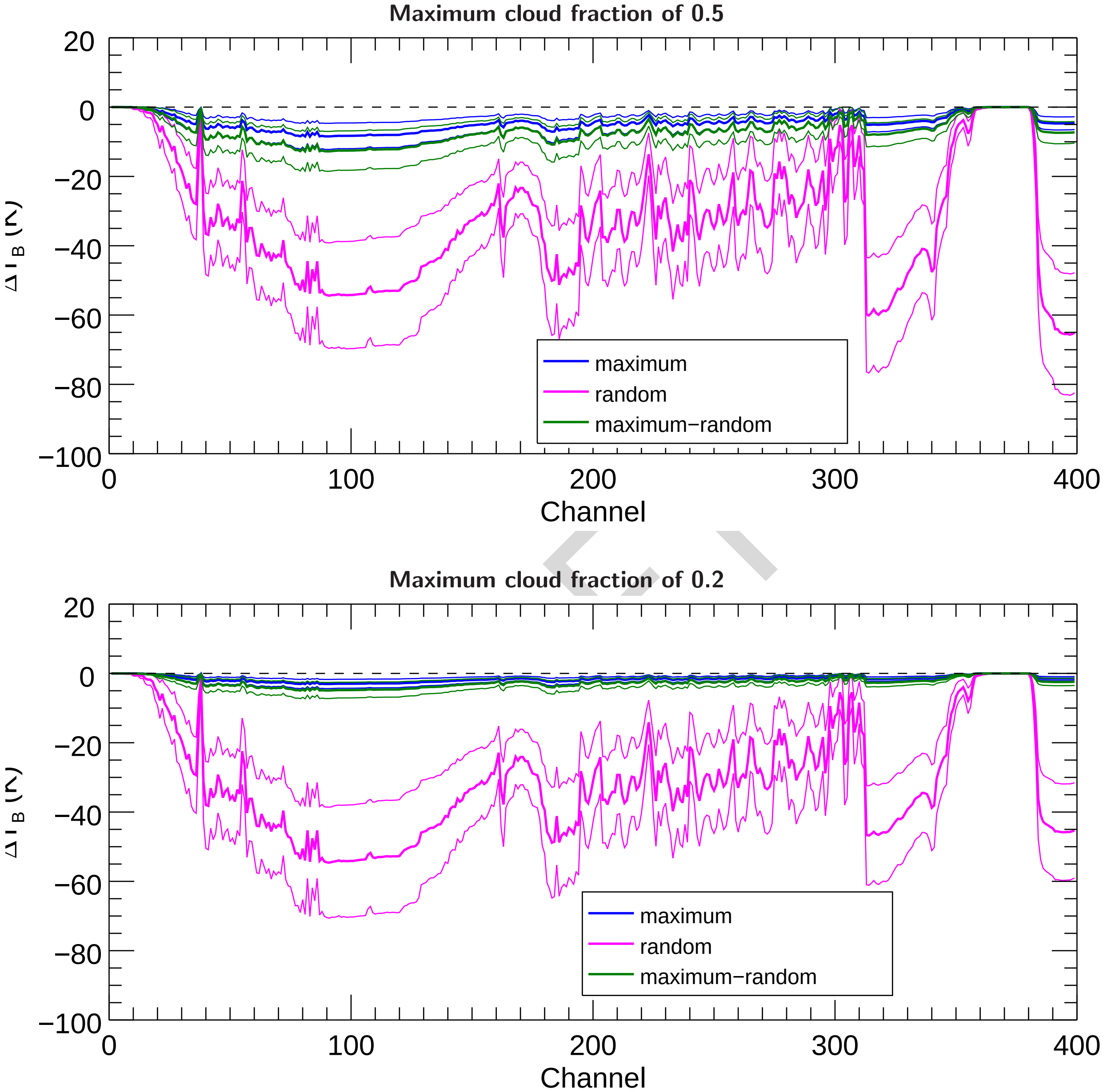


Figure 2.9: NPP CrIS average brightness temperature differences (399 channel subset) for cloud cover methods compared to the averaging approach using the ECMWF5K profile set. The thick line is the mean, the thin lines represent ± one standard deviation. **(Top)** Maximum cloud fraction of 0.5. **(Bottom)** Maximum cloud fraction of 0.2.



2.3 Finite-difference and K-matrix Jacobian comparisons

This section shows some comparisons of Jacobians computed using finite differences, via the CRTM forward model, with those produced by the CRTM K-matrix model.

The finite difference T_B Jacobian for an atmospheric state variable, x, was constructed by perturbing the state variable in a single layer to produce perturbed radiances,

$$J_{FD,k} = \frac{T_B(x_k + \Delta x) - T_B(x_k - \Delta x)}{2\Delta x} \qquad \text{for every} \quad k = 1, \ldots, K \qquad (2.4)$$

for each channel and where x is temperature, gaseous absorber concentration, or cloud water content. The perturbations, Δx, applied at each layer were 0.5K for temperature and 2.5% of the layer value for the other state variables.

Both the finite difference and K-matrix Jacobians were computed for the NPP ATMS instrument using the ECMWF5K profile set for each of the cloud cover computation methods. Brightness temperature Jacobian comparison plots for ATMS channel 2, profile 1 using the maximum overlap, random overlap, maximum-random overlap, and the cloud water amount weighted average method [?] to compute cloud cover are shown in figures 2.10 to 2.13. In all cases the unperturbed forward model brightness temperatures are shown in the bottom panel for all channels (note that the Jacobian x-axis and T_B y-axis ranges for these plots differs across the figures).

For comparison, the clear sky result for the same channel/profile is shown in figure 2.14.

The first thing to note about the Jacobian comparisons is that the finite difference and K-matrix versions agree very well.

In all cases the temperature Jacobians agree well. Comparison across cloud cover methodology are also good, with the magnitude differences due to the different amount of cloudiness in each case.

Similarly for the water vapour Jacobians. However, when comparing across cloud cover methodologies, the random-overlap case (figure 2.11) yields a differently shaped water vapour Jacobian with a maximum magnitude about an order of magnitude less.

As for the cloud water content Jacobians, there is good agreement between the finite difference (FD) and K-matrix (KM) Jacobians, both within the run and for the different cloud cover methodologies. The shape of the Jacobians for the averaging approach (figure 2.13) are different from the others, but recall that the averaging approach is the only cloud cover method that uses the cloud water content (see equation 1.5).

DRAFT

Figure 2.10: Comparison of NPP ATMS channel 2 finite-difference and K-matrix T_B Jacobians for ECMWF5K profile 1 using the *maximum overlap* method to compute fractional cloud cover. **(Top)** Temperature, water vapour, and cloud water content Jacobians. The colours for the latter represent **water**, **ice**, **rain**, and **snow** clouds. **(Bottom)** Channel brightness temperatures. Channel 2 is highlighted by the vertical green line.

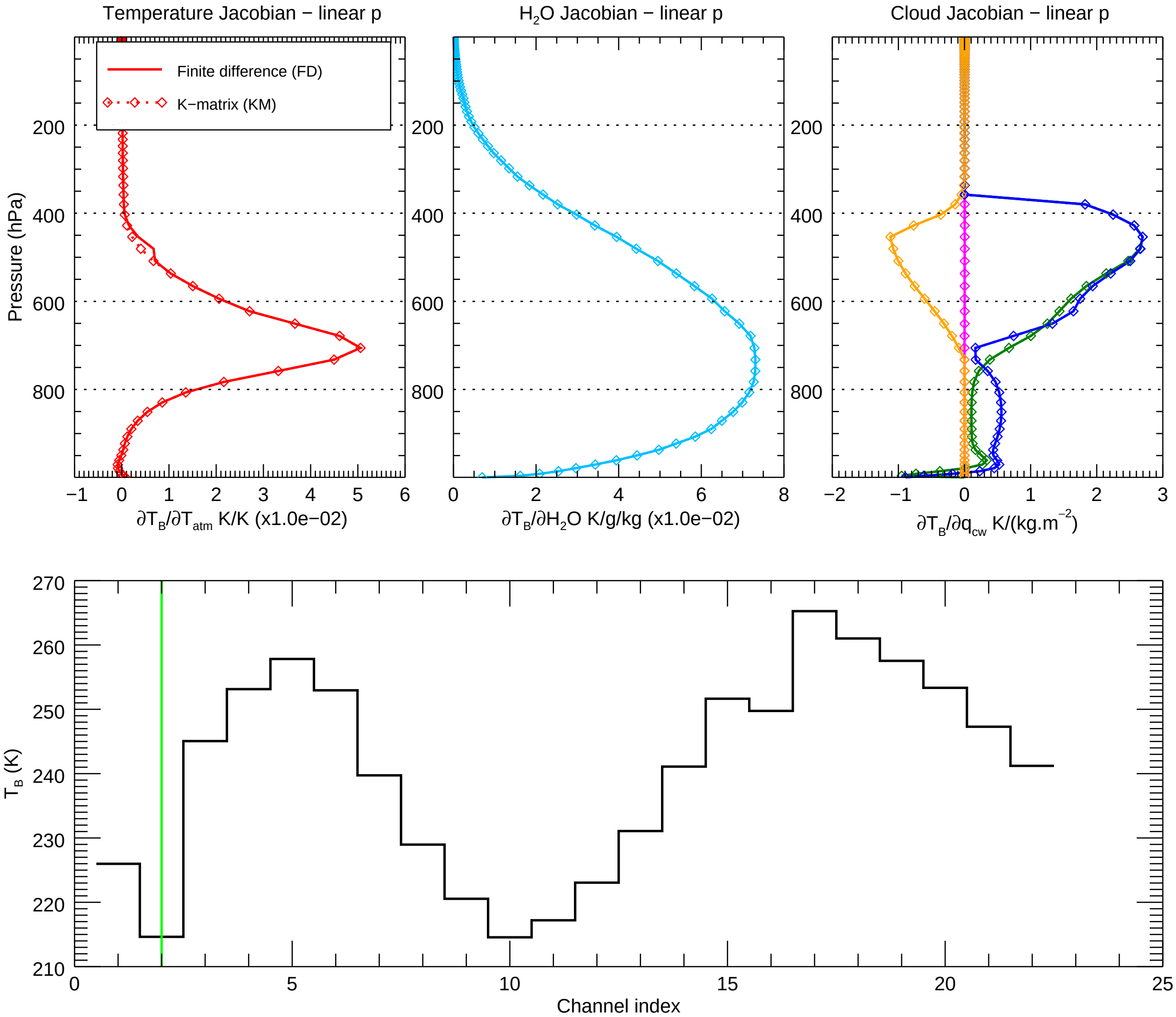
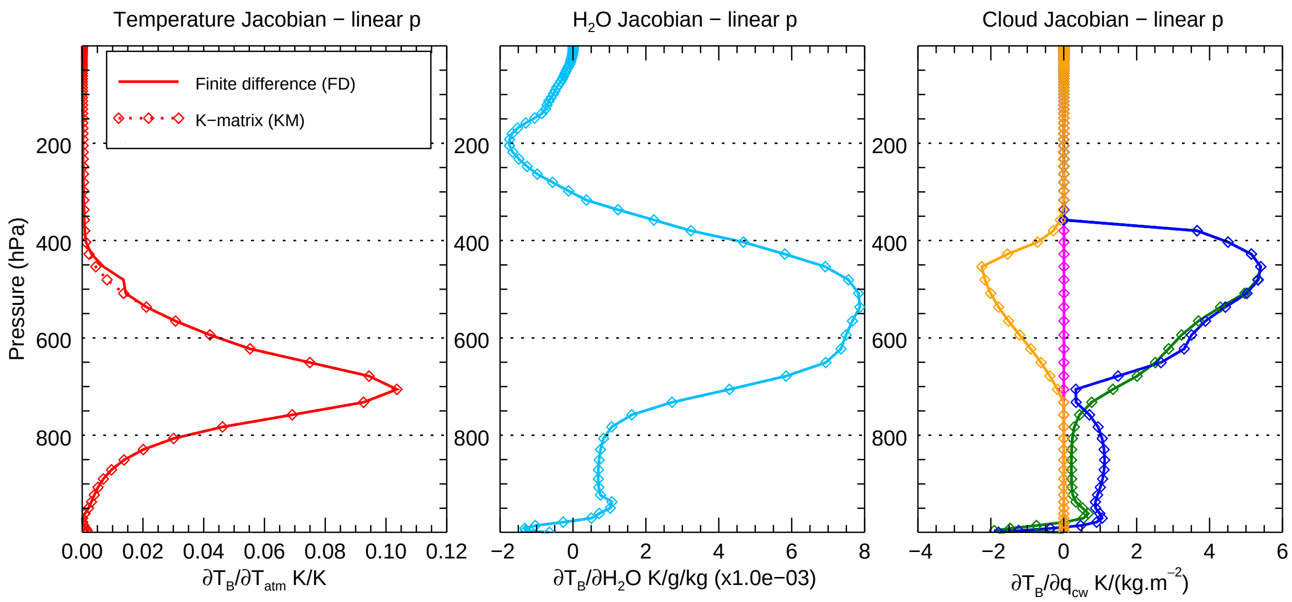


Figure 2.11: Comparison of NPP ATMS channel 2 finite-difference and K-matrix T_B Jacobians for ECMWF5K profile 1 using the *random overlap* method to compute fractional cloud cover. **(Top)** Temperature, water vapour, and cloud water content Jacobians. The colours for the latter represent **water**, **ice**, **rain**, and **snow** clouds. **(Bottom)** Channel brightness temperatures. Channel 2 is highlighted by the vertical green line.



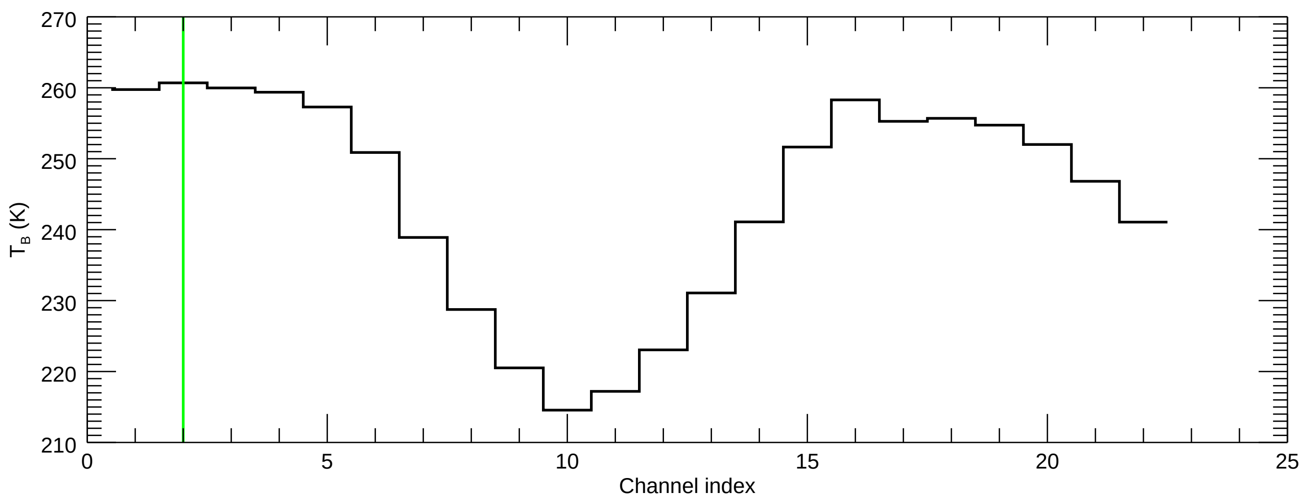


Figure 2.12: Comparison of NPP ATMS channel 2 finite-difference and K-matrix T_B Jacobians for ECMWF5K profile 1 using the *maximum-random overlap* method to compute fractional cloud cover. **(Top)** Temperature, water vapour, and cloud water content Jacobians. The colours for the latter represent **water**, **ice**, **rain**, and **snow** clouds. **(Bottom)** Channel brightness temperatures. Channel 2 is highlighted by the vertical green line.

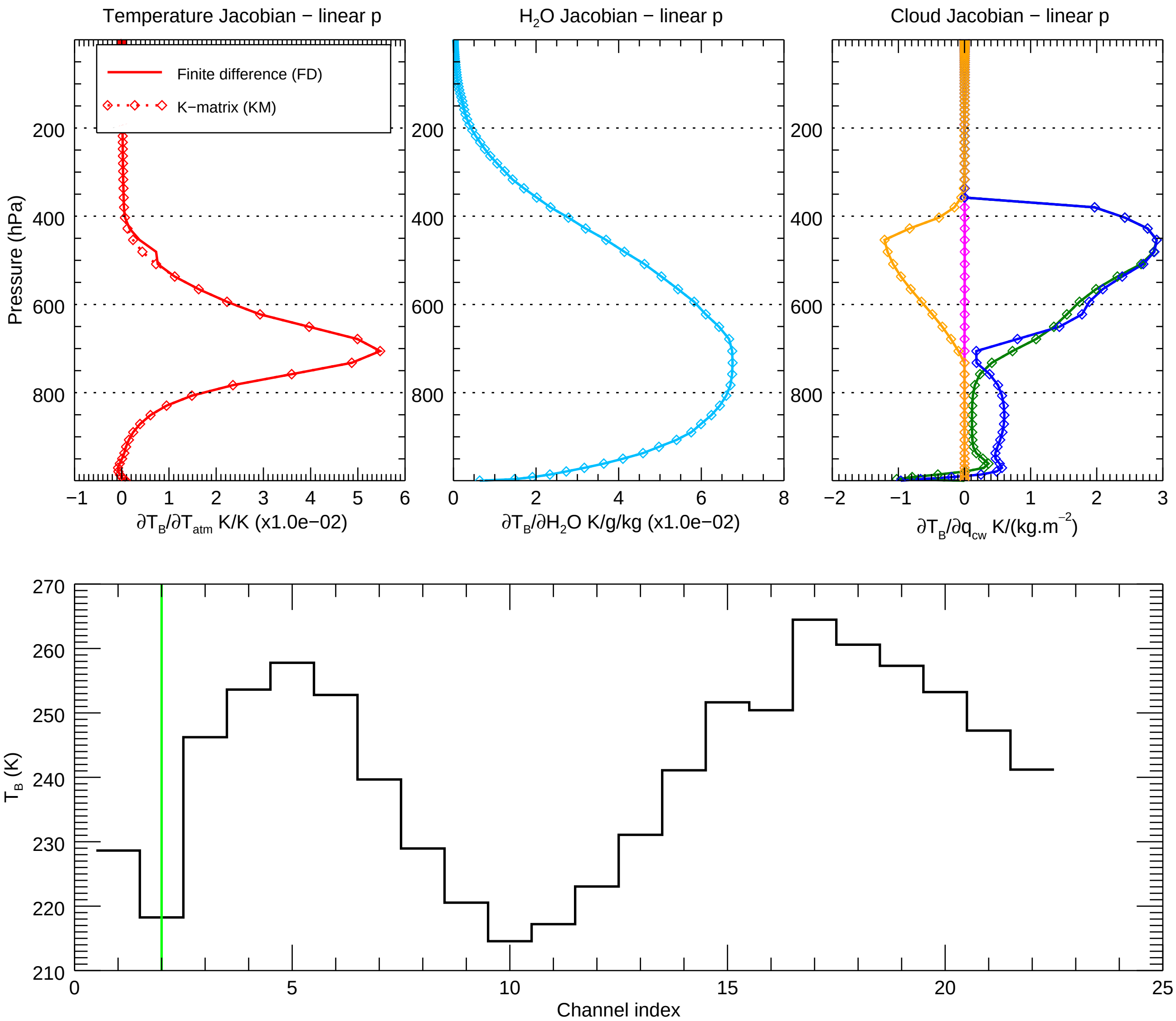


Figure 2.13: Comparison of NPP ATMS channel 2 finite-difference and K-matrix T_B Jacobians for ECMWF5K profile 1 using the *cloud water amount weighted average* method to compute fractional cloud cover. **(Top)** Temperature, water vapour, and cloud water content Jacobians. The colours for the latter represent **water**, **ice**, **rain**, and **snow** clouds. **(Bottom)** Channel brightness temperatures. Channel 2 is highlighted by the vertical green line.

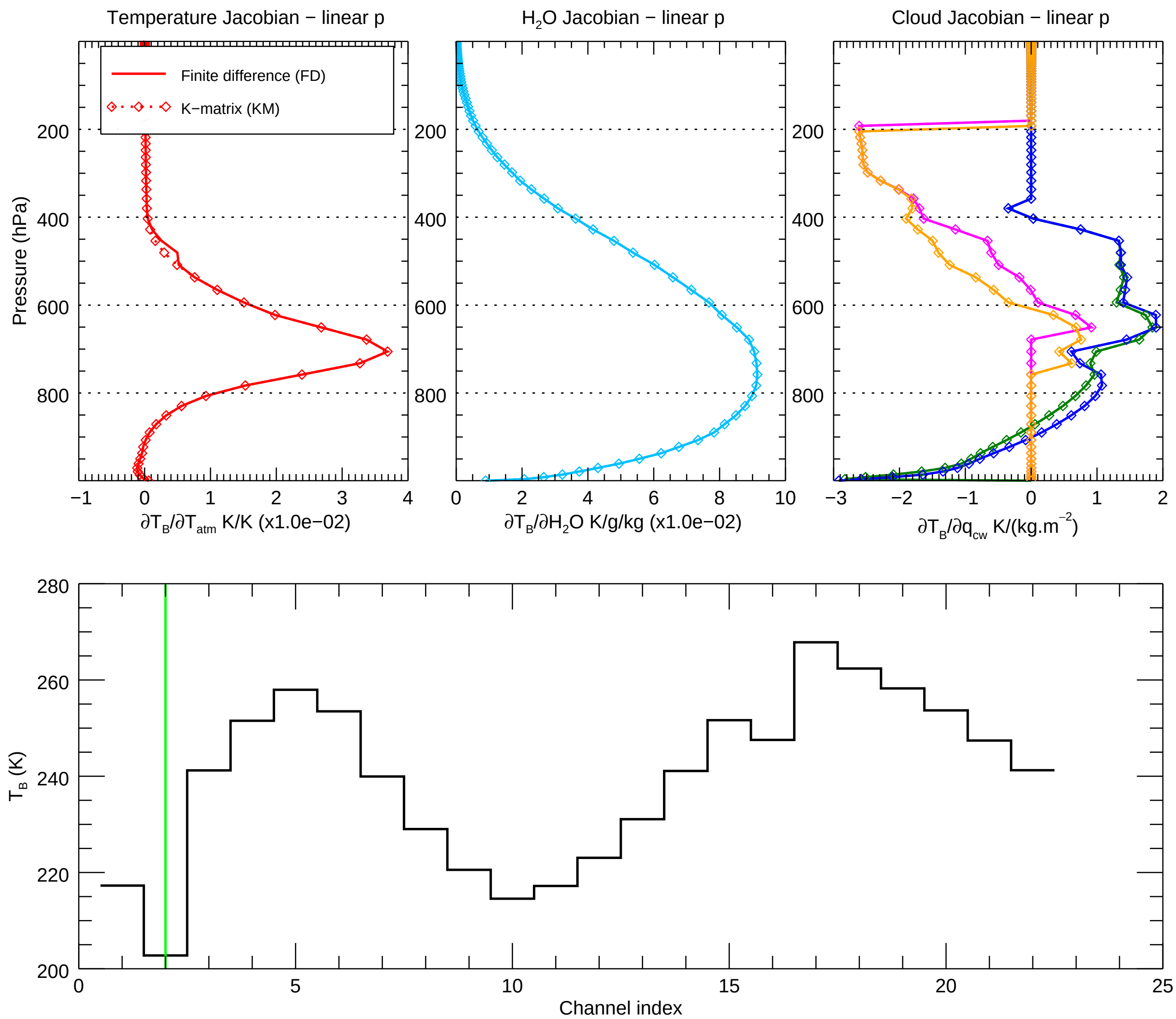
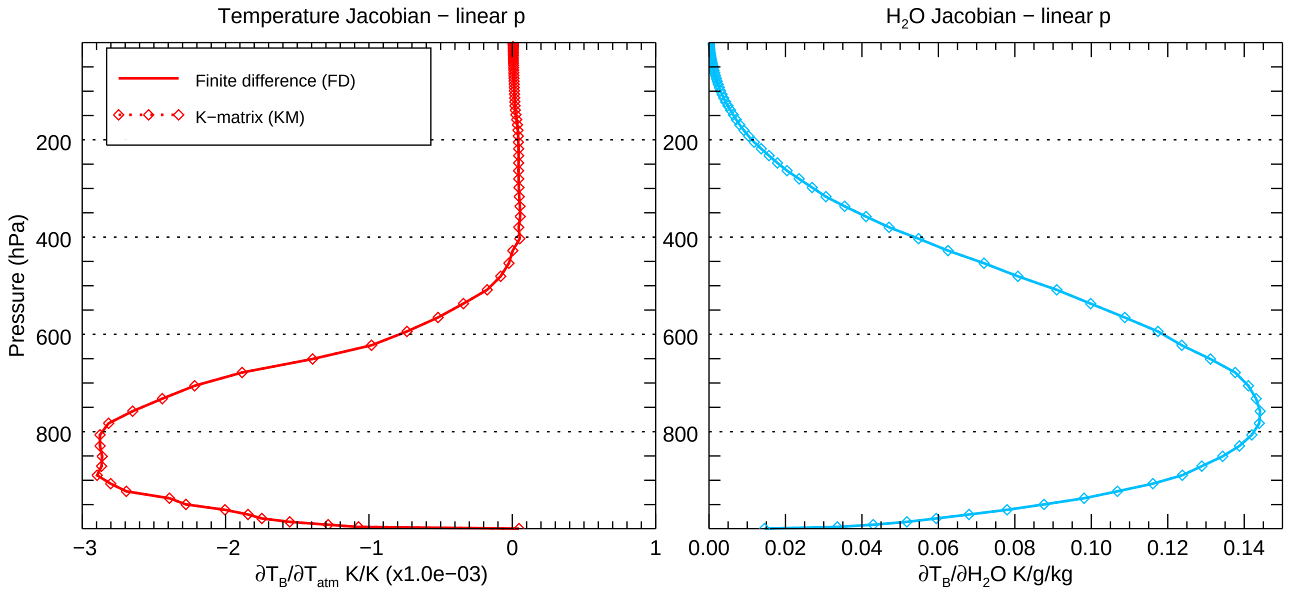


Figure 2.14: Comparison of NPP ATMS channel 2 finite-difference and K-matrix T_B Jacobians for ECMWF5K profile 1 with no clouds. **(Top)** Temperature and water vapour Jacobians. **(Bottom)** Clear sky channel brightness temperatures. Channel 2 is highlighted by the vertical green line.



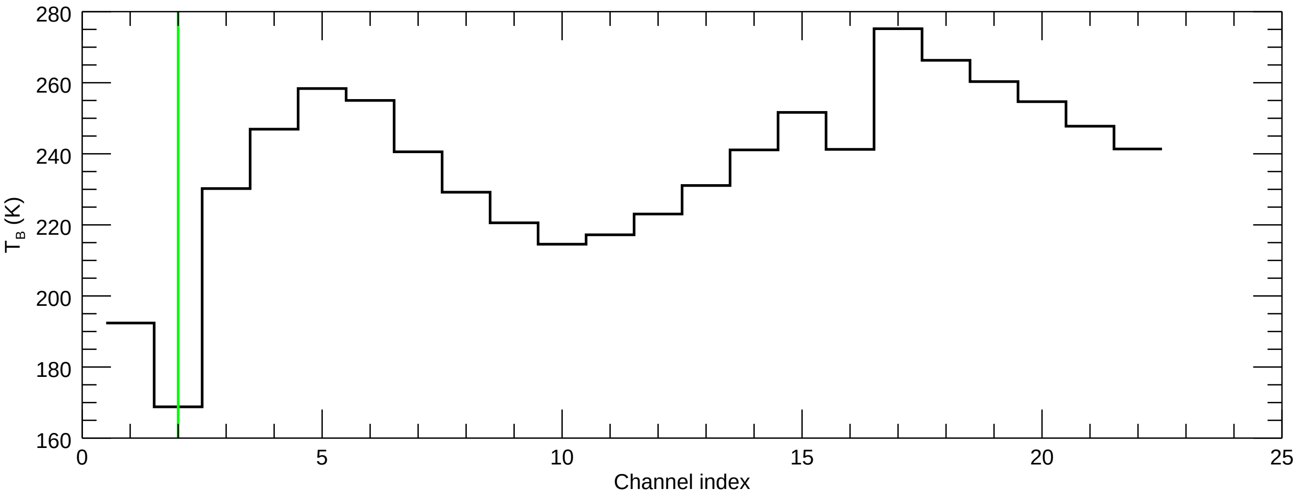
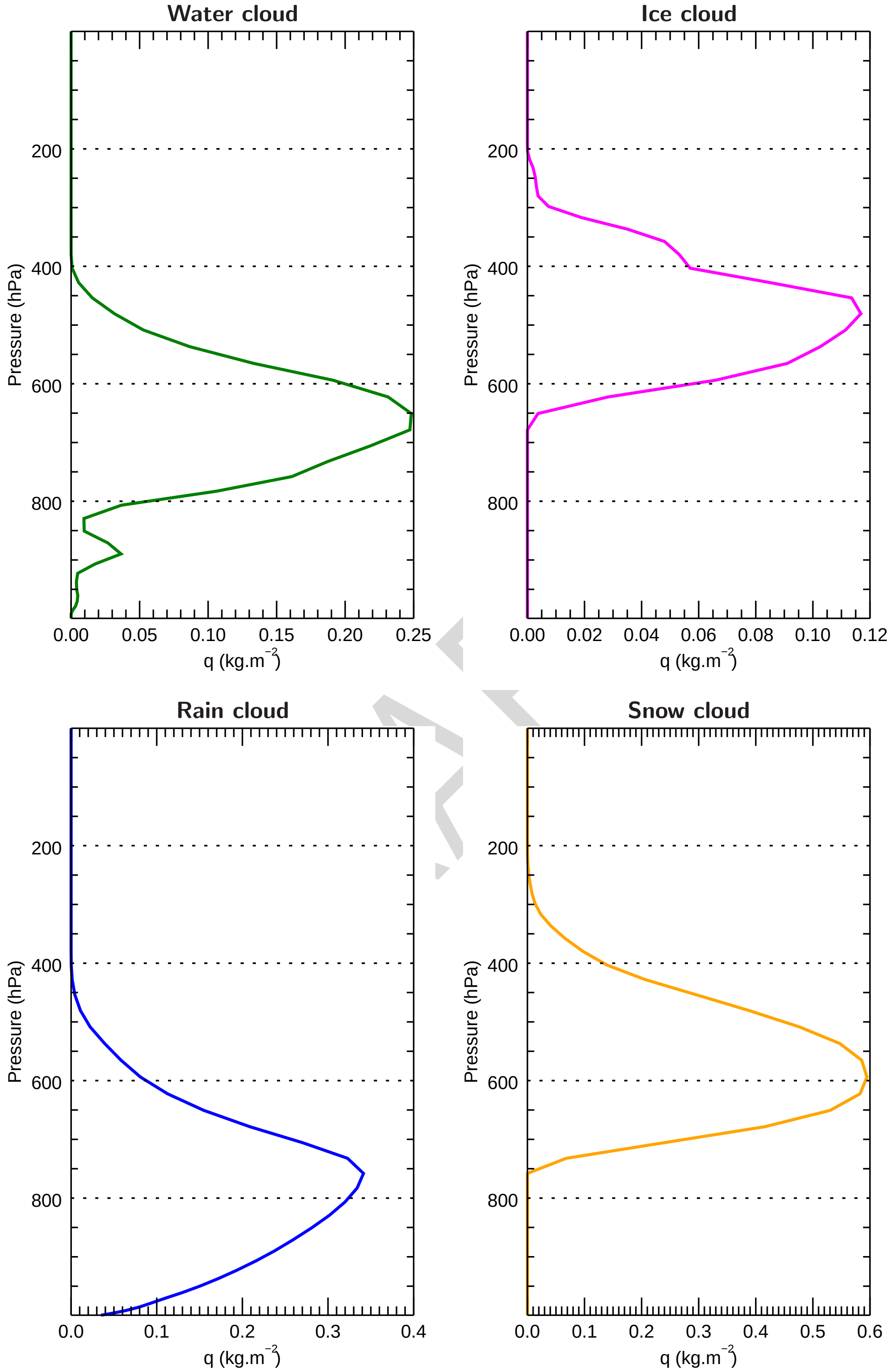


Figure 2.15: The cloud water contents for ECMWF5K profile 1.



2.4 Testing in the GSI

TBD

DRAFT

References

F. Chevallier, S. Di Michele, and A.P. McNally. Diverse profile datasets from the ECMWF 91-level short-range forecasts. NWP SAF Report NWPSAF-EC-TR-010.B, EUMETSAT/ECMWF, 2006. URL http://www.metoffice.com/research/interproj/nwpsaf/rtm.

DRAFT

A CloudCover object and method definitions

Figure A.1: CRTM_CloudCover_type object definition.

```
TYPE :: CRTM_CloudCover_type
  ! Housekeeping
  LOGICAL :: Is_Allocated = .FALSE.                   ! Allocation indicator
  INTEGER :: n_Layers = 0                             ! K dimension.
  ! Data
  INTEGER  :: Overlap           = DEFAULT_OVERLAP_ID  ! Overlap type identifier
  REAL(fp) :: Total_Cloud_Cover = ZERO                ! Cloud cover used in RT
  REAL(fp), ALLOCATABLE :: Cloud_Fraction(:)          ! K. The physical cloud fraction
  REAL(fp), ALLOCATABLE :: Cloud_Cover(:)             ! K. The overlap cloud cover
  ! Copy of the individual cloud type data
  TYPE(CRTM_Cloud_type), ALLOCATABLE :: Cloud(:)      ! Cloud information
  ! Intermediate results
  TYPE(iVar_type) :: iVar                             ! FWD results for TL/AD
CONTAINS
  PRIVATE
  PROCEDURE, PUBLIC, PASS(self)    :: Overlap_Id
  PROCEDURE, PUBLIC, PASS(self)    :: Overlap_Name
  PROCEDURE, PUBLIC, PASS(self)    :: Compute_CloudCover
  PROCEDURE, PUBLIC, PASS(self_TL) :: Compute_CloudCover_TL
  PROCEDURE, PUBLIC, PASS(self_AD) :: Compute_CloudCover_AD
  PROCEDURE, PUBLIC, PASS(self)    :: Is_Usable
  PROCEDURE, PUBLIC, PASS(self)    :: Destroy
  PROCEDURE, PUBLIC, PASS(self)    :: Create
  PROCEDURE, PUBLIC, PASS(self)    :: Inspect
  PROCEDURE, PUBLIC, PASS(self)    :: Set_To_Zero
  PROCEDURE :: Equal
  PROCEDURE :: NotEqual
  PROCEDURE :: Compare_
  GENERIC, PUBLIC :: OPERATOR(==) => Equal
  GENERIC, PUBLIC :: OPERATOR(/=) => NotEqual
  GENERIC, PUBLIC :: OPERATOR(.Compare.) => Compare_
END TYPE CRTM_CloudCover_type
```

DRAFT

A.1 Compute_CloudCover interface

```
NAME:
  Compute_CloudCover

PURPOSE:
  Function method to compute the cloud cover profile given a supplied
  Atmosphere object, and populate the CloudCover object with the
  results.

CALLING SEQUENCE:
  err_stat = cc_obj%Compute_CloudCover( &
               atmosphere       , &
               Overlap = overlap  )

OBJECTS:
  cc_obj:      Cloud cover object which is to be populated with cloud
               cover results.
               UNITS:      N/A
               CLASS:      CRTM_CloudCover_type
               DIMENSION:  Scalar
               ATTRIBUTES: INTENT(OUT)

INPUTS:
  atmosphere:  Atmopshere object containing the layer cloud fraction
               profile, and the actual cloud profiles for when cloud
               water content averaging of the cloud cover is selected.
               UNITS:      N/A
               TYPE:       CRTM_Atmosphere_type
               DIMENSION:  Scalar
               ATTRIBUTES: INTENT(IN)

OPTIONAL INPUTS:
  overlap:     Set this argument to a flag defining the cloud coverage
               algorithm used. Supplied module functions providing valid flag
               output are:
                 CloudCover_Maximum_Overlap(): Use maximum overlap method.
                 CloudCover_Random_Overlap() : Use random overlap method.
                 CloudCover_MaxRan_Overlap() : Use maximum-random overlap method.
                 CloudCover_Average_Overlap(): Use cloud content weighted averaged method. [DEFAULT]
               If not specified, the default is the cloud content weighted
               averaged method
               UNITS:      N/A
               TYPE:       INTEGER
               DIMENSION:  Scalar
               ATTRIBUTES: INTENT(IN), OPTIONAL

FUNCTION RESULT:
  err_stat:    The return value is an integer defining the error status.
               The error codes are defined in the Message_Handler module.
                 If == SUCCESS, the computation was successful
                    == FAILURE, an unrecoverable error occurred.
               UNITS:      N/A
               TYPE:       INTEGER
               DIMENSION:  Scalar
```

DRAFT

A.2 Compute_CloudCover_TL interface

```
NAME:
  Compute_CloudCover_TL

PURPOSE:
  Function method to compute the tangent-linear cloud cover profile for
  supplied forward model results and a Atmosphere perturbation, and populate
  the tangent-linear CloudCover object with the results.

CALLING SEQUENCE:
  err_stat = cc_obj_TL%Compute_CloudCover_TL( &
               cc_FWD       , &
               atmosphere_TL  )

OBJECTS:
  cc_obj_TL:      The tangent-linear cloud cover object which is to be
                  populated with perturbed cloud cover results.
                  UNITS:      N/A
                  CLASS:      CRTM_CloudCover_type
                  DIMENSION:  Scalar
                  ATTRIBUTES: INTENT(OUT)

INPUTS:
  cc_FWD:         The forward model cloud cover object.
                  UNITS:      N/A
                  CLASS:      CRTM_CloudCover_type
                  DIMENSION:  Scalar
                  ATTRIBUTES: INTENT(IN)

  atmosphere_TL:  The tangent-linear atmosphere object containing the layer
                  cloud fraction perturbation profile, and the cloud amount
                  perturbation profiles for when cloud water content averaging
                  of the cloud cover is selected.
                  UNITS:      N/A
                  TYPE:       CRTM_Atmosphere_type
                  DIMENSION:  Scalar
                  ATTRIBUTES: INTENT(IN)

FUNCTION RESULT:
  err_stat:       The return value is an integer defining the error status.
                  The error codes are defined in the Message_Handler module.
                    If == SUCCESS, the computation was successful
                       == FAILURE, an unrecoverable error occurred.
                  UNITS:      N/A
                  TYPE:       INTEGER
                  DIMENSION:  Scalar
```

DRAFT

A.3 Compute_CloudCover_AD interface

```
NAME:
  Compute_CloudCover_AD

PURPOSE:
  Function method to compute the adjoint cloud cover profile for supplied
  forward model results and populate the Atmosphere adjoint object
  with the results.

CALLING SEQUENCE:
  err_stat = cc_obj_AD%Compute_CloudCover_AD( &
               cc_FWD       , &
               atmosphere_AD  )

OBJECTS:
  cc_obj_AD:      The adjoint cloud cover object. This object contains
                  data on input, but is zeroed out upon output.
                  UNITS:      N/A
                  CLASS:      CRTM_CloudCover_type
                  DIMENSION:  Scalar
                  ATTRIBUTES: INTENT(IN OUT)

INPUTS:
  cc_FWD:         The forward model cloud cover object.
                  UNITS:      N/A
                  CLASS:      CRTM_CloudCover_type
                  DIMENSION:  Scalar
                  ATTRIBUTES: INTENT(IN)

OUTPUTS:
  atmosphere_AD:  The adjoint atmosphere object containing the results of the
                  adjoint calculation. The adjoint cloud fraction profile
                  will be modified on output. For the weighted average cloud
                  clover method, the adjoint cloud water amounts will also be
                  modified.
                  UNITS:      N/A
                  TYPE:       CRTM_Atmosphere_type
                  DIMENSION:  Scalar
                  ATTRIBUTES: INTENT(IN OUT)

FUNCTION RESULT:
  err_stat:       The return value is an integer defining the error status.
                  The error codes are defined in the Message_Handler module.
                    If == SUCCESS, the computation was successful
                       == FAILURE, an unrecoverable error occurred.
                  UNITS:      N/A
                  TYPE:       INTEGER
                  DIMENSION:  Scalar
```

A.4 == interface

DRAFT

```
NAME:
  ==

PURPOSE:
  Operator method to test the equality of two CloudCover objects.

CALLING SEQUENCE:
  IF ( x == y ) THEN
    ...
  END IF

OBJECTS:
  x, y:  Two cloud cover object to be compared.
         UNITS:      N/A
         TYPE:       CRTM_CloudCover_type
         DIMENSION:  Scalar or any rank
         ATTRIBUTES: INTENT(IN)
```

A.5 /= interface

```
NAME:
  /=

PURPOSE:
  Operator method to test the inequality of two CloudCover objects.

CALLING SEQUENCE:
  IF ( x /= y ) THEN
    ...
  END IF

OBJECTS:
  x, y:  Two cloud cover objects to be compared.
         UNITS:      N/A
         TYPE:       CRTM_CloudCover_type
         DIMENSION:  Scalar or any rank
         ATTRIBUTES: INTENT(IN)
```

A.6 .Compare. interface

```
NAME:
  .Compare.

PURPOSE:
  Operator method to compare two CloudCover objects.

  This procedure performs similarly to the == operator, but is non-elemental
  to allow for informational output when a difference is found between the
  two objects being compared.
```

DRAFT

```
    Mostly used for debugging.

  CALLING SEQUENCE:
    IF ( x .Compare. y ) THEN
      ...
    END IF

  OBJECTS:
    x, y:      The cloud cover objects to compare.
               UNITS:      N/A
               CLASS:      CRTM_CloudCover_type
               DIMENSION:  Scalar
               ATTRIBUTES: INTENT(IN)
```

A.7 Create interface

```
  NAME:
    Create

  PURPOSE:
    Elemental subroutine method to create instances of CloudCover objects.

  CALLING SEQUENCE:
    CALL cc_obj%Create( n_Layers, &
                        Forward       = Forward, &
                        Error_Message = Error_Message )

  OBJECTS:
    cc_obj:         Cloud cover object
                    UNITS:      N/A
                    CLASS:      CRTM_CloudCover_type
                    DIMENSION:  Scalar or any rank
                    ATTRIBUTES: INTENT(OUT)

  INPUTS:
    n_Layers:       Number of layers for which there is cloud data.
                    Must be > 0.
                    UNITS:      N/A
                    TYPE:       INTEGER
                    DIMENSION:  Conformable with object.
                    ATTRIBUTES: INTENT(IN)

  OPTIONAL INPUTS:
    Forward:        Set this optional logical flag to allocate the sub-object to
                    hold the intermediate forward model results.
                    IF .FALSE. - the subobject is NOT allocated [DEFAULT]
                       .TRUE.  - the subobject is allocated
                    UNITS:      N/A
                    TYPE:       CHARACTER(*)
                    DIMENSION:  Conformable with object.
                    ATTRIBUTES: INTENT(IN), OPTIONAL
```

DRAFT

```
OPTIONAL OUTPUTS:
  Error_Message:  If an error occurred creating the object, this
                  argument will contain error information.
                  UNITS:      N/A
                  TYPE:       CHARACTER(*)
                  DIMENSION:  Conformable with object.
                  ATTRIBUTES: INTENT(OUT), OPTIONAL
```

A.8 Destroy interface

```
NAME:
  Destroy

PURPOSE:
  Elemental subroutine method to re-initialize CloudCover objects.

CALLING SEQUENCE:
  CALL cc_obj%Destroy()

OBJECTS:
  cc_obj:  Re-initialized cloud cover object(s).
           UNITS:      N/A
           CLASS:      CRTM_CloudCover_type
           DIMENSION:  Scalar or any rank
           ATTRIBUTES: INTENT(OUT)
```

DRAFT

A.9 Inspect interface

```
NAME:
  Inspect

PURPOSE:
  Subroutine method to display the contents of a CloudCover object.

CALLING SEQUENCE:
  CALL cc_obj%Inspect( Hires=hires, Unit=unit, Verbose=Verbose )

OBJECTS:
  cc_obj:  Cloud cover object
           UNITS:      N/A
           CLASS:      CRTM_CloudCover_type
           DIMENSION:  Scalar
           ATTRIBUTES: INTENT(IN)

OPTIONAL INPUTS:
  Hires:   Set this logical argument to output object contents with
           more significant digits.
           If == .FALSE., output format is 'es13.6' [DEFAULT].
              == .TRUE.,  output format is 'es22.15'
```

```
          If not specified, default is .FALSE.
          UNITS:      N/A
          TYPE:       LOGICAL
          DIMENSION:  Scalar
          ATTRIBUTES: INTENT(IN), OPTIONAL

 Unit:    Unit number for an already open file to which the output
          will be written.
          If the argument is specified and the file unit is not
          connected, the output goes to stdout.
          UNITS:      N/A
          TYPE:       INTEGER
          DIMENSION:  Scalar
          ATTRIBUTES: INTENT(IN), OPTIONAL

 Verbose: Set this logical argument to output the intermediate variable
          sub-object contents if they are available.
          If == .FALSE., the intermediate variables are NOT output [DEFAULT].
             == .TRUE.,  the intermediate variables are output if available
          If not specified, default is .FALSE.
          UNITS:      N/A
          TYPE:       LOGICAL
          DIMENSION:  Scalar
          ATTRIBUTES: INTENT(IN), OPTIONAL
```

A.10 Is_Usable interface

```
NAME:
  Is_Usable

PURPOSE:
  Elemental function method to test the status of CloudCover objects to
  determien if they are usable.

CALLING SEQUENCE:
  status = cc_obj%Is_Usable( Include_iVar = Include_iVar )

OBJECTS:
  cc_obj:       Cloud cover object which is to have its usability tested.
                UNITS:      N/A
                CLASS:      CRTM_CloudCover_type
                DIMENSION:  Scalar or any rank
                ATTRIBUTES: INTENT(IN)

OPTIONAL INPUTS:
  Include_iVar: Set this optional logical flag to alos check the status
                of the intermediate variable sub-object.
                IF .FALSE. - the subobject is NOT tested [DEFAULT]
                   .TRUE.  - the subobject is tested
                UNITS:      N/A
                TYPE:       CHARACTER(*)
                DIMENSION:  Conformable with object.
                ATTRIBUTES: INTENT(IN), OPTIONAL
```

DRAFT

```
FUNCTION RESULT:
  status:        The return value is a logical value indicating the
                 usable status of the object.
                   .TRUE.  - if the object is usable.
                   .FALSE. - if the object is NOT usable.
                 UNITS:      N/A
                 TYPE:       LOGICAL
                 DIMENSION:  Same as object
```

A.11 Overlap_Id interface

```
NAME:
  Overlap_Id

PURPOSE:
  Function method to return the overlap methodology identifier of a
  CloudCover object.

CALLING SEQUENCE:
  id = cc_obj%Overlap_Id()

OBJECTS:
  cc_obj:  Cloud cover object for which the overlap methodology identifier
           is required.
           UNITS:      N/A
           CLASS:      CRTM_CloudCover_type
           DIMENSION:  Scalar
           ATTRIBUTES: INTENT(OUT)

FUNCTION RESULT:
  id:      The return value is an integer defining the overlap methodology.
           The actual number value of these integers in a CRTM release can
           change at any time based upon code updates.
           UNITS:      N/A
           TYPE:       INTEGER
           DIMENSION:  Scalar
```

A.12 Overlap_Name interface

```
NAME:
  Overlap_Name

PURPOSE:
  Function method to return a string description of the overlap methodology
  that has been set for a CloudCover object.

CALLING SEQUENCE:
  name = cc_obj%Overlap_Name()
```

DRAFT

```
OBJECTS:
  cc_obj:  Cloud cover object for which the overlap methodology descriptor
           is required.
           UNITS:      N/A
           CLASS:      CRTM_CloudCover_type
           DIMENSION:  Scalar
           ATTRIBUTES: INTENT(IN)

FUNCTION RESULT:
  name:    Character variable containing a short descriptor of the overlap
           methodology. If the object's overlap methodology identifier is
           invalid, the returned string is "Invalid".
           UNITS:      N/A
           TYPE:       CHARACTER(*)
           DIMENSION:  Scalar
```

A.13 Set_To_Zero interface

```
NAME:
  Set_To_Zero

PURPOSE:
  Elemental subroutine method to zero out the data arrays in a
  CloudCover object.

CALLING SEQUENCE:
  CALL cc_obj%Set_To_Zero()

OBJECTS:
  cc_obj:  Cloud cover object
           UNITS:      N/A
           CLASS:      CRTM_CloudCover_type
           DIMENSION:  Scalar or any rank
           ATTRIBUTES: INTENT(IN OUT)

COMMENTS:
  - The dimension components of the object are *NOT* set to zero.
  - The overlap methodology identifier component is *NOT* reset.
```

DRAFT

B `Options` object and method definitions

DRAFT

Figure B.1: CRTM_Options_type structure definition.

```
TYPE :: CRTM_Options_type
  ! Allocation indicator
  LOGICAL :: Is_Allocated = .FALSE.

  ! Input checking on by default
  LOGICAL :: Check_Input = .TRUE.

  ! User defined MW water emissivity algorithm
  LOGICAL :: Use_Old_MWSSEM = .FALSE.

  ! Antenna correction application
  LOGICAL :: Use_Antenna_Correction = .FALSE.

  ! NLTE radiance correction is ON by default
  LOGICAL :: Apply_NLTE_Correction = .TRUE.

  ! RT Algorithm is set to ADA by default
  INTEGER(Long) :: RT_Algorithm_Id = RT_ADA

  ! Aircraft flight level pressure
  ! Value > 0 turns "on" the aircraft option
  REAL(Double) :: Aircraft_Pressure = -ONE

  ! User defined number of RT solver streams (streams up + streams down)
  LOGICAL       :: Use_n_Streams = .FALSE.
  INTEGER(Long) :: n_Streams = 0

  ! Scattering switch. Default is for
  ! Cloud/Aerosol scattering to be included.
  LOGICAL :: Include_Scattering = .TRUE.

  ! Cloud cover overlap id is set to averaging type by default
  INTEGER(Long) :: Overlap_Id = DEFAULT_OVERLAP_ID

  ! User defined emissivity/reflectivity
  ! ...Dimensions
  INTEGER(Long) :: n_Channels = 0  ! L dimension
  ! ...Index into channel-specific components
  INTEGER(Long) :: Channel = 0
  ! ...Emissivity optional arguments
  LOGICAL :: Use_Emissivity = .FALSE.
  REAL(Double), ALLOCATABLE :: Emissivity(:)  ! L
  ! ...Direct reflectivity optional arguments
  LOGICAL :: Use_Direct_Reflectivity = .FALSE.
  REAL(Double), ALLOCATABLE :: Direct_Reflectivity(:) ! L

  ! SSU instrument input
  TYPE(SSU_Input_type) :: SSU

  ! Zeeman-splitting input
  TYPE(Zeeman_Input_type) :: Zeeman

END TYPE CRTM_Options_type
```

DRAFT

B.1 CRTM_Options_SetValue interface

```
NAME:
  CRTM_Options_SetValue

PURPOSE:
  Elemental subroutine to set the values of the non-dimensional,
  non-contained-object CRTM_Options object components.

CALLING SEQUENCE:
  CALL CRTM_Options_SetValue( &
         Options                                             , &
         Check_Input               = Check_Input             , &
         Use_Old_MWSSEM            = Use_Old_MWSSEM          , &
         Use_Antenna_Correction    = Use_Antenna_Correction  , &
         Apply_NLTE_Correction     = Apply_NLTE_Correction   , &
         Set_ADA_RT                = Set_ADA_RT              , &
         Set_SOI_RT                = Set_SOI_RT              , &
         Include_Scattering        = Include_Scattering      , &
         Set_Maximum_Overlap       = Set_Maximum_Overlap     , &
         Set_Random_Overlap        = Set_Random_Overlap      , &
         Set_MaxRan_Overlap        = Set_MaxRan_Overlap      , &
         Set_Average_Overlap       = Set_Average_Overlap     , &
         Set_Average_Overlap       = Set_Average_Overlap     , &
         Use_Emissivity            = Use_Emissivity          , &
         Use_Direct_Reflectivity = Use_Direct_Reflectivity, &
         n_Streams                 = n_Streams               , &
         Aircraft_Pressure         = Aircraft_Pressure         )

OBJECTS:
  Options:                  Options object for which the indicated component
                            values are to be set.
                            UNITS:      N/A
                            TYPE:       CRTM_Options_type
                            DIMENSION:  Scalar or any rank
                            ATTRIBUTES: INTENT(IN OUT)

OPTIONAL INPUTS:
  Check_Input:              Set this logical argument to control checking of
                            the CRTM input data.
                            If == .TRUE. , the CRTM input data is checked [DEFAULT]
                               == .FALSE., no input data checking is done.
                            UNITS:      N/A
                            TYPE:       LOGICAL
                            DIMENSION:  Conformable with Options object
                            ATTRIBUTES: INTENT(IN), OPTIONAL

  Use_Old_MWSSEM:           Set this logical argument to invoke the previous version
                            of the microwave sea surface emissivity model.
                            If == .TRUE. , the old model is used.
                               == .FALSE., the current model is used [DEFAULT]
                            UNITS:      N/A
                            TYPE:       LOGICAL
                            DIMENSION:  Conformable with Options object
                            ATTRIBUTES: INTENT(IN), OPTIONAL
```

DRAFT

```
Use_Antenna_Correction:   Set this logical argument to apply an antenna correction
                          to the computed brightness temperatures for certain
                          microwave instruments (AMSU-A/B, MHS)
                          If == .TRUE. , antenna correction is applied
                             == .FALSE., no correction is applied [DEFAULT]
                          UNITS:      N/A
                          TYPE:       LOGICAL
                          DIMENSION:  Conformable with Options object
                          ATTRIBUTES: INTENT(IN), OPTIONAL

Apply_NLTE_Correction:    Set this logical argument to apply an non-LTE correction
                          to shortwave infrared radiances.
                          If == .TRUE. , non-LTE correction is applied [DEFAULT]
                             == .FALSE., no correction is applied
                          UNITS:      N/A
                          TYPE:       LOGICAL
                          DIMENSION:  Conformable with Options object
                          ATTRIBUTES: INTENT(IN), OPTIONAL

Set_ADA_RT:
Set_SOI_RT:               Set this logical argument to use the specified algorithm
                          for scattering radiative transfer.
                          If == .TRUE. , the corresponding RT algorithm is used.
                          Note: - By default, the ADA algorithm is used.
                                - If MORE THAN ONE argument is specified, the
                                  the default ADA algorithm is used.
                          UNITS:      N/A
                          TYPE:       LOGICAL
                          DIMENSION:  Conformable with Options object
                          ATTRIBUTES: INTENT(IN), OPTIONAL

Include_Scattering:       Set this logical argument to control the inclusion of
                          cloud and aerosol scattering in the radiative transfer.
                          If == .TRUE. , scattering calculations are performed [DEFAULT]
                             == .FALSE., only cloud/aerosol absorption is considered.
                          UNITS:      N/A
                          TYPE:       LOGICAL
                          DIMENSION:  Conformable with Options object
                          ATTRIBUTES: INTENT(IN), OPTIONAL

Set_Maximum_Overlap:
Set_Random_Overlap:
Set_MaxRan_Overlap:
Set_Average_Overlap:      Use these logical arguments to set the cloud overlap
                          methodology for fractionally cloudy input profiles.
                          If == .TRUE. , the corresponding overlap method is used.
                          Note: - By default, the average overlap method is used.
                                - If MORE THAN ONE overlap argument is specified,
                                  the default overlap method is used.
                          UNITS:      N/A
                          TYPE:       LOGICAL
                          DIMENSION:  Conformable with Options object
                          ATTRIBUTES: INTENT(IN), OPTIONAL

Use_Emissivity:           Set this logical argument to control the use of the emissivity
```

DRAFT

```
                             spectrum included in the object.
                             If == .TRUE. , use the included emissivity spectrum
                                == .FALSE., let the CRTM compute the emissivity spectrum
                             Note: - This argument is ignored if the object does not
                                     contain any emissivity data
                                   - See the CRTM_Options_SetEmissivity() procedure for
                                     loading emissivity data into an Options object.
                             UNITS:      N/A
                             TYPE:       LOGICAL
                             DIMENSION:  Conformable with Options object
                             ATTRIBUTES: INTENT(IN), OPTIONAL

    Use_Direct_Reflectivity: Set this logical argument to control the use of the direct
                             reflectivity spectrum included in the object.
                             If == .TRUE. , use the included direct reflectivity spectrum
                                == .FALSE., let the CRTM compute the direct reflectivity spectrum
                             Note: - This argument is ignored if the object does not
                                     contain any direct reflectivity data
                                   - See the CRTM_Options_SetEmissivity() procedure for
                                     loading direct relfectivity data into an Options object.
                             UNITS:      N/A
                             TYPE:       LOGICAL
                             DIMENSION:  Conformable with Options object
                             ATTRIBUTES: INTENT(IN), OPTIONAL

    n_Streams:               Set this integer argument to the number of streams (up + down)
                             to use in the radiative transfer solver for scattering
                             atmospheres.
                             By default, a channel-specific value is selected based
                             on the Mie parameter.
                             UNITS:      N/A
                             TYPE:       INTEGER
                             DIMENSION:  Conformable with Options object
                             ATTRIBUTES: INTENT(IN), OPTIONAL

    Aircraft_Pressure:       Set this real argument to aircraft pressure level to use
                             for an aircraft instrument simulation.
                             Note: This option has not been rigorously tested.
                             UNITS:      hPa
                             TYPE:       REAL(fp)
                             DIMENSION:  Conformable with Options object
                             ATTRIBUTES: INTENT(IN), OPTIONAL
```

DRAFT