An Introduction to the JCSDA Community Radiative Transfer Model (CRTM)

Yong Han
NOAA/NESDIS
JCSDA

Outline

- The role of a fast radiative transfer model in satellite data assimilation
- CRTM main modules
- Radiative transfer (RT) equation applied
- CRTM surface emissivity/reflectivity models
- RT solution in clear sky environment
- CRTM fast transmittance model
- RT solution in cloudy/aerosol environment
- CRTM Forward, Tangent-linear, Adjoint and K-Matrix models
- CRTM user interface

Why need a fast RT model?

Variational Analysis: the cost function

$$J(x) = \frac{1}{2}(x - x_b)^T B^{-1}(x - x_b) + \frac{1}{2}[y(x) - y_m]^T (E_m + E_F)^{-1}[y(x) - y_m]$$

- Background model state x_b and its covariance B
- Measurement y_m and its error covariance E_m
- Radiative transfer (RT) model y(x) and its error E_F

Minimization:

$$\frac{\partial J}{\partial x} = B^{-1}(x - x_b) - H(x)^T E^{-1}\{y_m - y(x)\} = 0 \quad \Longrightarrow \quad \boldsymbol{x}$$

$$H(x) = \{h_{i,j}\} = \left\{\frac{\partial y_i}{\partial x_j}\right\}$$

Jacobian matrix with respect to state variables

- An RT model maps state variables to radiances and it can be used to retrieve information about the state variables from the radiance measurements.
- CRTM is a fast RT model, which employs many computational efficient algorithms to meet the requirements of the operational data assimilation system.

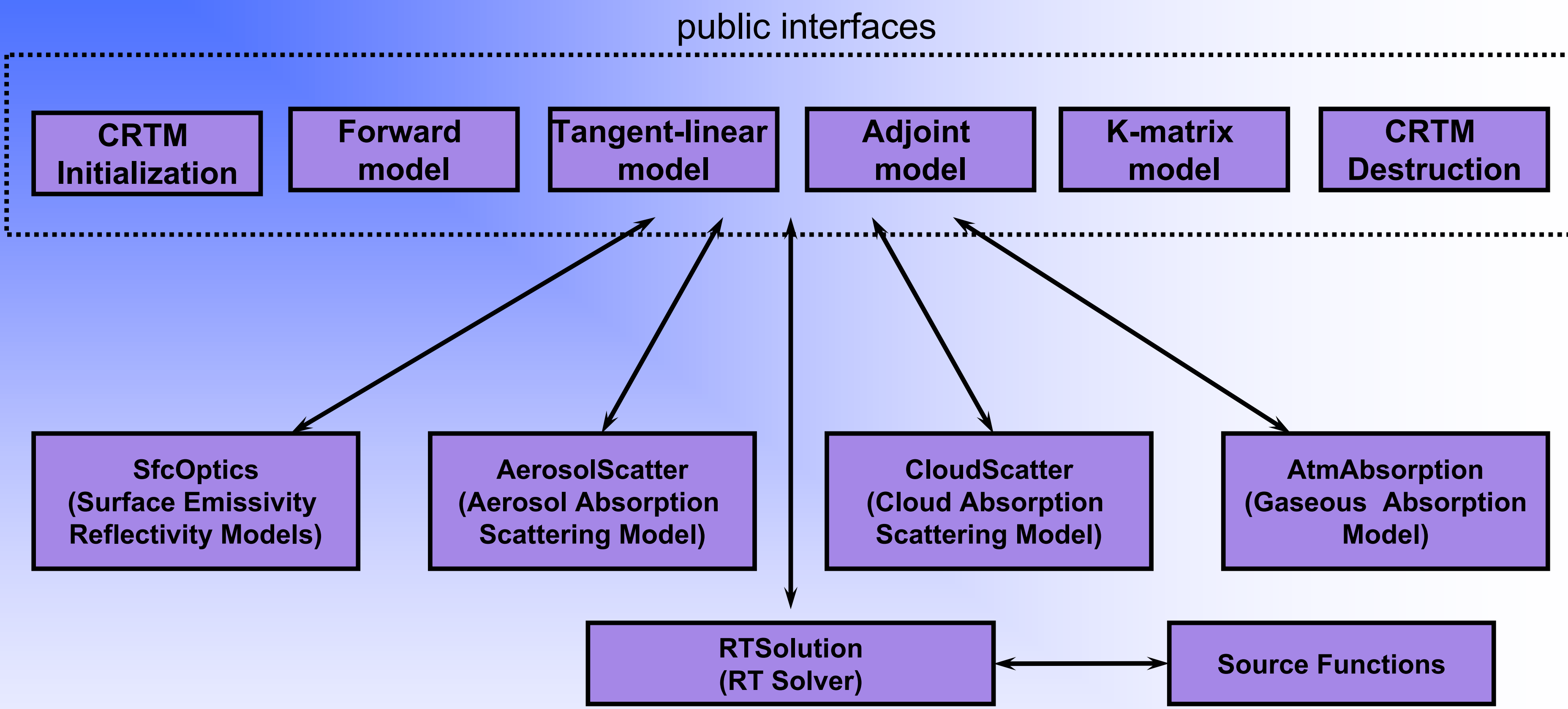
Community Contributions

- Community Research: Radiative transfer science
  - UWisc – Successive Order of Iteration
  - University of Colorado –DOTLRT
  - UCLA – Delta 4 stream vector radiative transfer model
  - Princeton Univ – snow emissivity model improvement
  - NESDIS – Advanced doubling and adding scheme, surface emissivity models, LUT for aerosols, clouds, precip
  - AER – Optimal Spectral Sampling (OSS) Method
  - UMBC – SARTA
- Core team (ORA/EMC): Smooth transition from research to operation
  - Maintenance of CRTM
  - CRTM interface
  - Benchmark tests for model selection
  - Integration of new science into CRTM

What CRTM does?

- Compute satellite radiances (Forward model)
- Compute radiance responses to the perturbations of the state variables (Tangent-linear model)
- Compute Adjoint (Adjoint model)
- Compute Jacobians (K-matrix model)

CRTM Major Modules



Supported Instruments (IR & MW)

TIROS-N to NOAA-18 AVHRR
TIROS-N to NOAA-18 HIRS
GOES-8 to 13 Imager channels
GOES-8 to 13 sounder channel 08-13
GOES-R ABI
Terra/Aqua MODIS Channel 1-10
METEOSAT-SG1 SEVIRI

Aqua AIRS
Aqua AMSR-E
Aqua AMSU-A
Aqua HSB
NOAA-15 to 18 AMSU-A
NOAA-15 to 17 AMSU-B
NOAA-18 MHS
TIROS-N to NOAA-14 MSU

DMSP F13,15 SSM/T1
DMSP F14,15 SSM/T2
DMSP F16 SSMIS
NPP ATMS
Coriolis Windsat
METOP-A AMSUA, MHS, HIRS, AVHRR

Two of the fundamental variables in CRTM

Radiance:

Specific intensity:

I_ν mW/(m^2.sr.cm^{-1})

the flux of energy in a given direction per second per unit frequency range per unit solid angle per unit area perpendicular to the given direction

Brightness Temperature:

BT_ν Kelvin

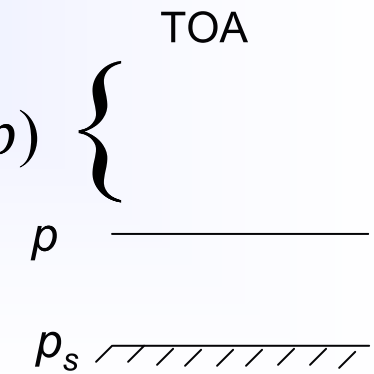
which is the Inverse of the Planck function $B_\nu(T) = \dfrac{2h\nu^3}{c^2(e^{h\nu/kT}-1)}$

$$BT_\nu = B_\nu^{-1}(I_\nu)$$

Atmospheric transmittance:

$$\tau_\nu(p) = e^{-\int_0^p \sigma(p')dp'}$$

$\tau_\nu(p)$



e.g. if I_s is a upward radiance emitted from the surface, then $I_s\tau_\nu(p_s)$ is the amount of energy contribution from the surface received by a satellite sensor.

Radiative Transfer Equation

- Stokes vector: $\{I, Q, U, V\}^T$ describes the intensity, phase and polarization of the radiation field.

  where $I = I_v + I_h,\ Q = \ldots$

- The current operational CRTM does not model the Stokes vector; instead CRTM solves the following equation for a scalar intensity:

$$\mu \frac{dI(\tau,\Omega)}{d\tau} = -I(\tau,\Omega) + \frac{\omega}{4\pi}\int P(\tau,\Omega,\Omega')I(\tau,\Omega')d\Omega'$$

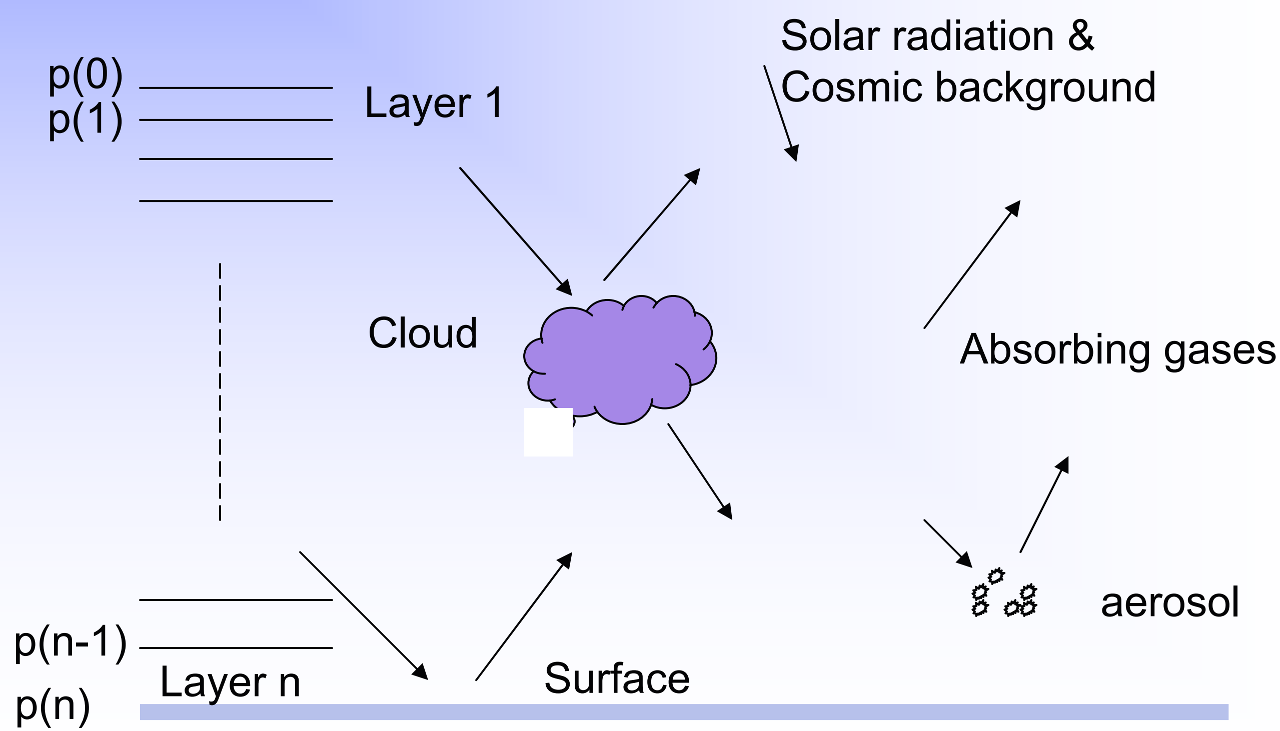
$$+\,\omega P(\tau,\Omega,\Omega_{\oplus})(F_{\oplus}/4\pi)e^{-\tau/\mu_{\oplus}} + (1-\omega)B(T(\tau))$$

τ --- optical path

- The current operational CRTM assumes:

(1) the atmospheric radiation is unpolarized (exception: O2 Zeeman absorption and emission in the mesosphere)

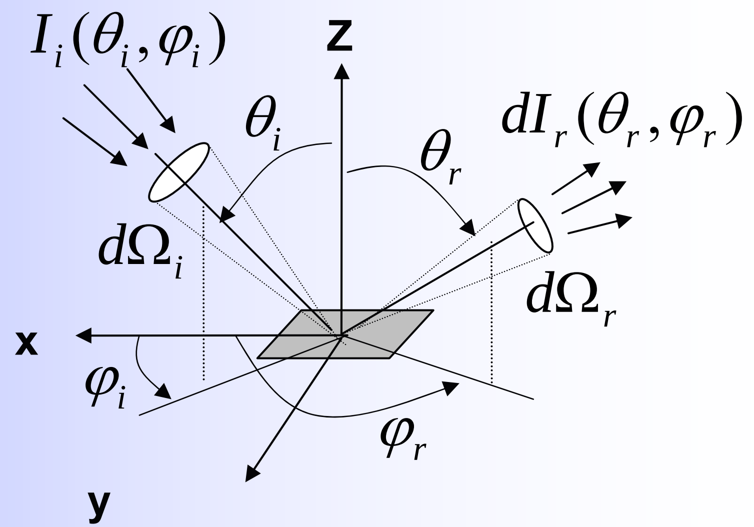
(2) Polarization is neglected for IR sensors.



CRTM surface emissivity/reflectivity models

Bidirectional reflectance distribution function (BRDF):

$$BRDF(\theta_i,\varphi_i,\theta_r,\varphi_r) \equiv \frac{dI_r(\theta_r,\varphi_r)}{I_i(\theta_i,\varphi_i)\cos(\theta_i)d\Omega_i(\theta_i,\varphi_i)}$$



Reflectance:

$$\rho(\theta_i,\varphi_i) = \int BRDF(\theta_i,\varphi_i,\theta_r,\varphi_r)\cos(\theta_r)d\Omega_r$$

Emissivity (by Kirchhoff's law):

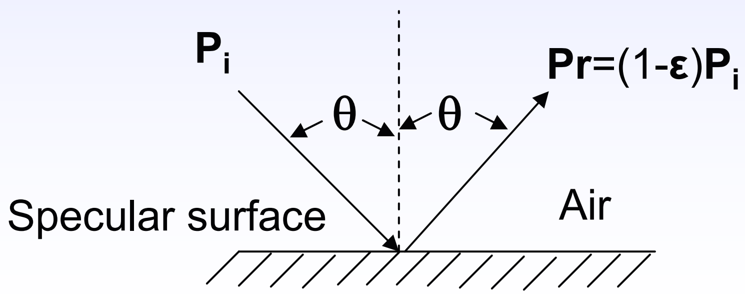
$$\varepsilon(\theta_i,\varphi_i) = 1-\rho(\theta_i,\varphi_i)$$

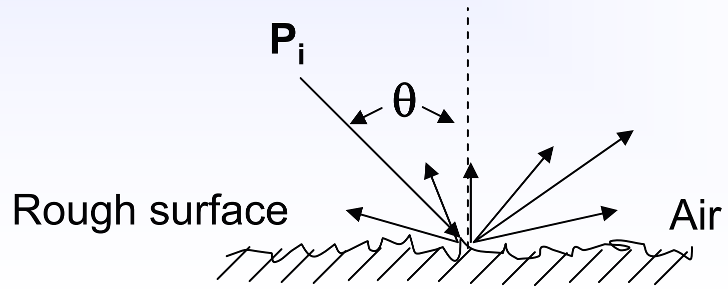
Specular surface:

$$BRDF(\theta_i,\varphi_i,\theta_r,\varphi_r) = \rho(\theta_i)\delta(\theta_r-\theta_i)\delta(\varphi_r-\varphi_i)/(\cos(\theta_r)\sin(\theta_r))$$

Lambertian (perfect rough) surface: $BRDF(\theta_i,\varphi_i,\theta_r,\varphi_r) = \dfrac{\rho}{\pi}$

CRTM assumes a specular reflection for MW sensors and Lambertian reflection for IR sensors, and uses relationship: $\rho(\theta) = 1-\varepsilon(\theta)$ (**but** $\varepsilon(\theta)$ **is not limited to the two special cases**)

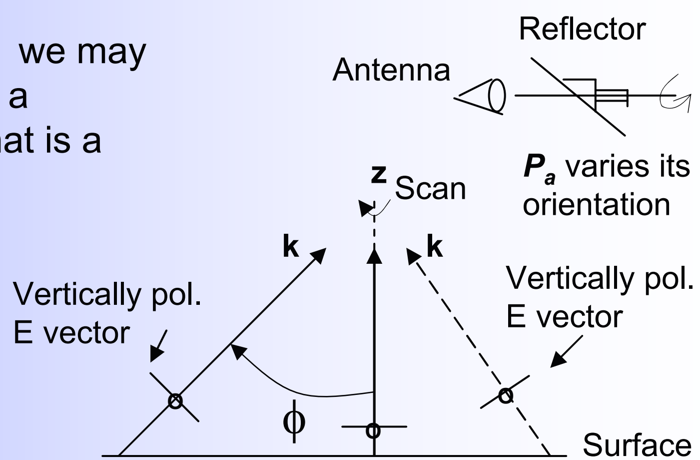




Polarization in surface emission and scattering (MW)

- Emission and scattering by a surface is usually polarization-dependent.
- A radiometer with a linearly polarized antenna would measure the same amount of atmospheric emission (randomly polarized) regardless of the direction of the antenna polarization vector $\mathbf{p_a}$; but would receive a different amount of radiation from the surface when the $\mathbf{p_a}$ vector changes orientation

- Since the atmospheric radiation is assumed unpolarized, we may still use a scalar RT equation to compute a radiance with a polarization $\mathbf{p_a}$ by using a surface emissivity/reflectivity that is a combination of polarization components.



The circles represent the horizontal polariztion perpendicular to the (**k**,**z**) plane)

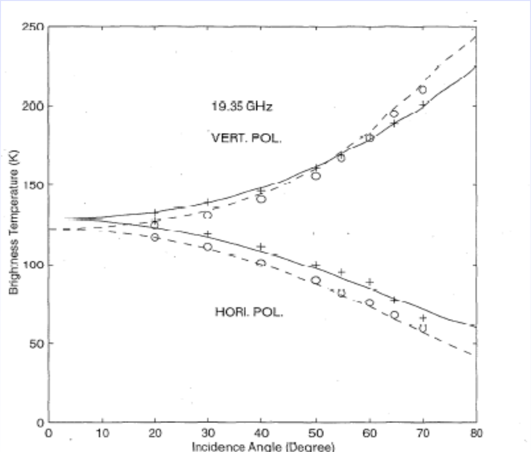
Example: the AMSU sensor has channels with linear polarized antennas. For v polarized antennas (when viewing nadir direction, $\mathbf{p_a}$ is in the scanning plane):

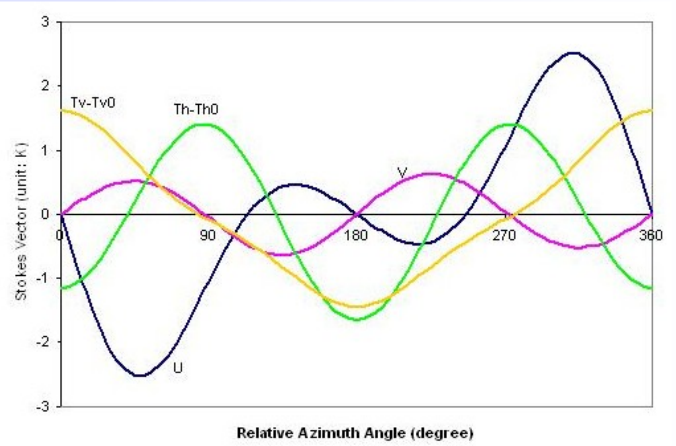
$$\varepsilon_{p_a}(nadir\ V\ \ pol.) = \varepsilon_V \cos^2(\phi) + \varepsilon_h \sin^2(\phi)$$

and for h polarization ($\mathbf{p_a}$ is perpendicular to the scanning plane:

$$\varepsilon_{p_a}(nadir\ h\ \ pol.) = \varepsilon_V \sin^2(\phi) + \varepsilon_h \cos^2(\phi)$$

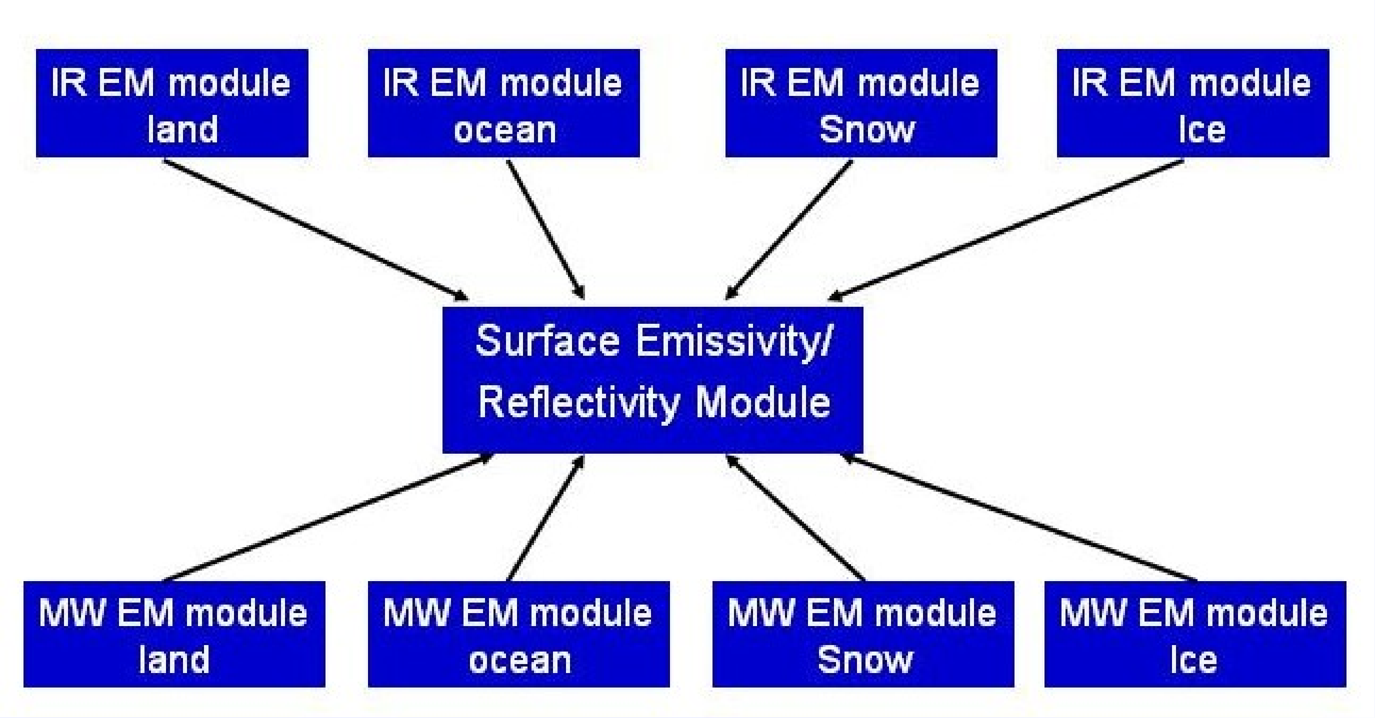
Model calculations:
solid line for wind speed = 13.5 m/s; dashed line for 3.5 m/s.
Measured:
crosses for wind speed = 13.5 m/s; circles for 0.5 m/s





Over ocean surface

CRTM surface emissivity/reflectivity module



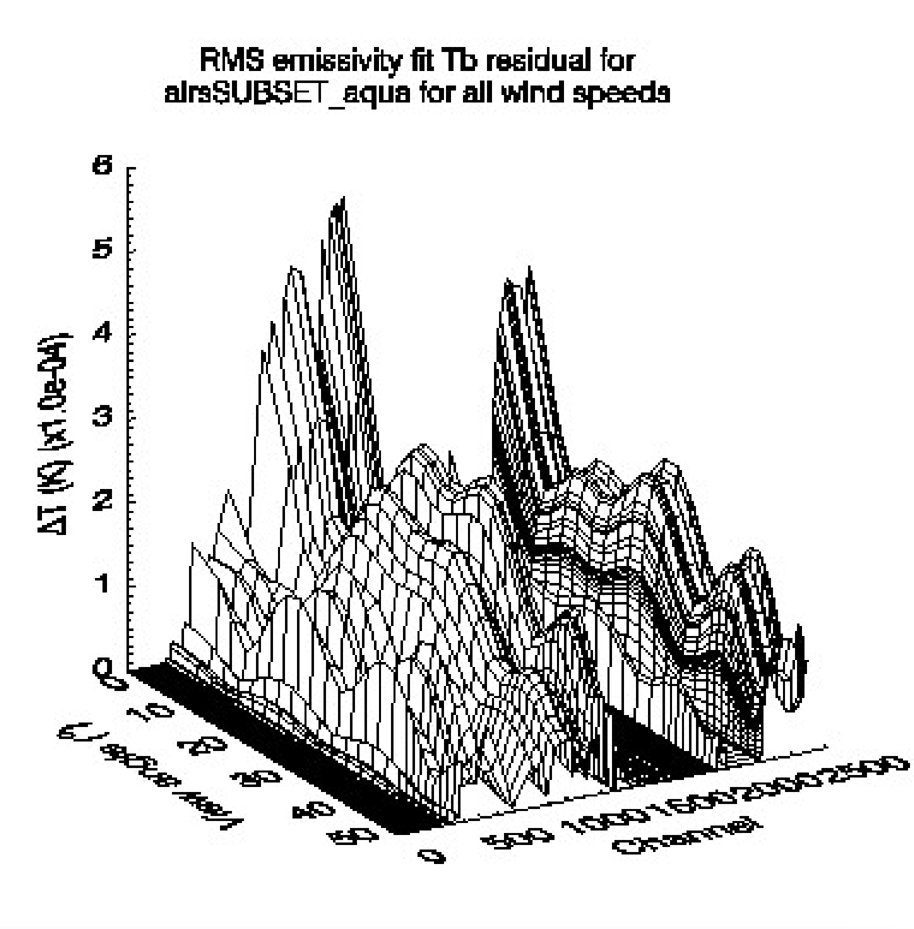
The CRTM surface emissivity module is split into 8 sub-modules to accommodate new model implementations and model improvement.

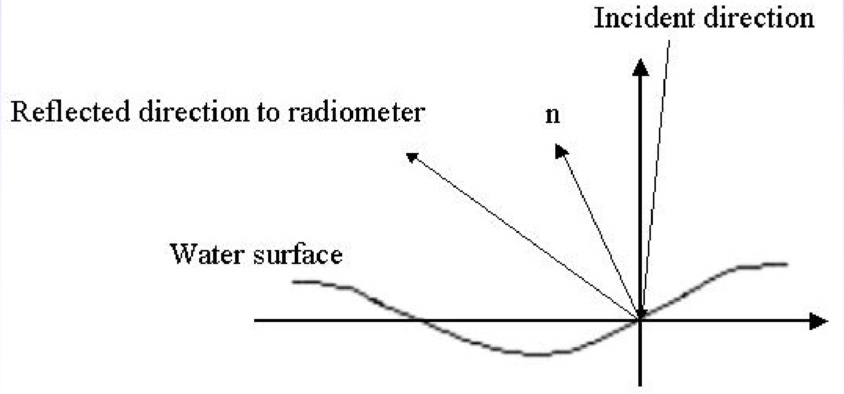
IR Sea Surface Emission Model (IRSSE)

$$\varepsilon(\theta, w, v) = c_0(w, v) + c_1(w, v)\hat{\theta}^{c_2(w,v)} + c_3(w, v)\hat{\theta}^{c_4(w,v)}$$

c_0 – c_4 are regression coefficients, obtained through regression against Wu-Smith model.

The IRSSE model is a parameterized Wu-Smith model for rough sea surface emissivity





IR emissivity lookup table for land surfaces

Surface types included in the IR emissivity database (Carter et al., 2002):

Surface Type	
Compacted soil	Grass scrub
Tilled soil	Oil grass
Sand	Urban concrete
Rock	Pine brush
Irrigated low vegetation	Broadleaf brush
Meadow grass	Wet soil
Scrub	Scrub soil
Broadleaf forest	Broadleaf(70)/Pine(30)
Pine forest	Water
Tundra	Old snow
Grass soil	Fresh snow
Broadleaf/Pine forest	New ice

Microwave Ocean Emissivity Model

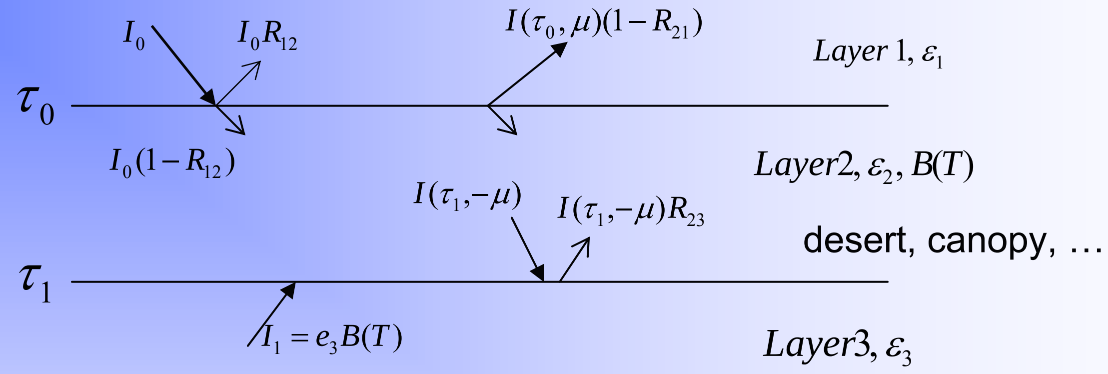
FASTEM-1 (English and Hewison, 1998):

Model inputs:
satellite zenith angle, water temperature, surface wind speed, and frequency

Model outputs:
emissivity (Vertical polarization) and emissivity (horizontal polarization)

NESDIS Microwave Land Emission Model (LandEM)

(1) Three layer medium:



(2) Emissivity derived from a two-stream radiative transfer solution and modified Fresnel equations for reflection and transmission at layer interfaces:

$$\varepsilon = \alpha R_{12} + (1-R_{21})\frac{(1-\beta)[1+\gamma e^{-2k(\tau_1-\tau_0)}] + \alpha(1-R_{12})[\beta - \gamma e^{-2k(\tau_1-\tau_0)}]}{(1-\beta R_{21}) - (\beta - R_{21})\gamma e^{-2k(\tau_1-\tau_0)}}$$

Weng, et al, 2001

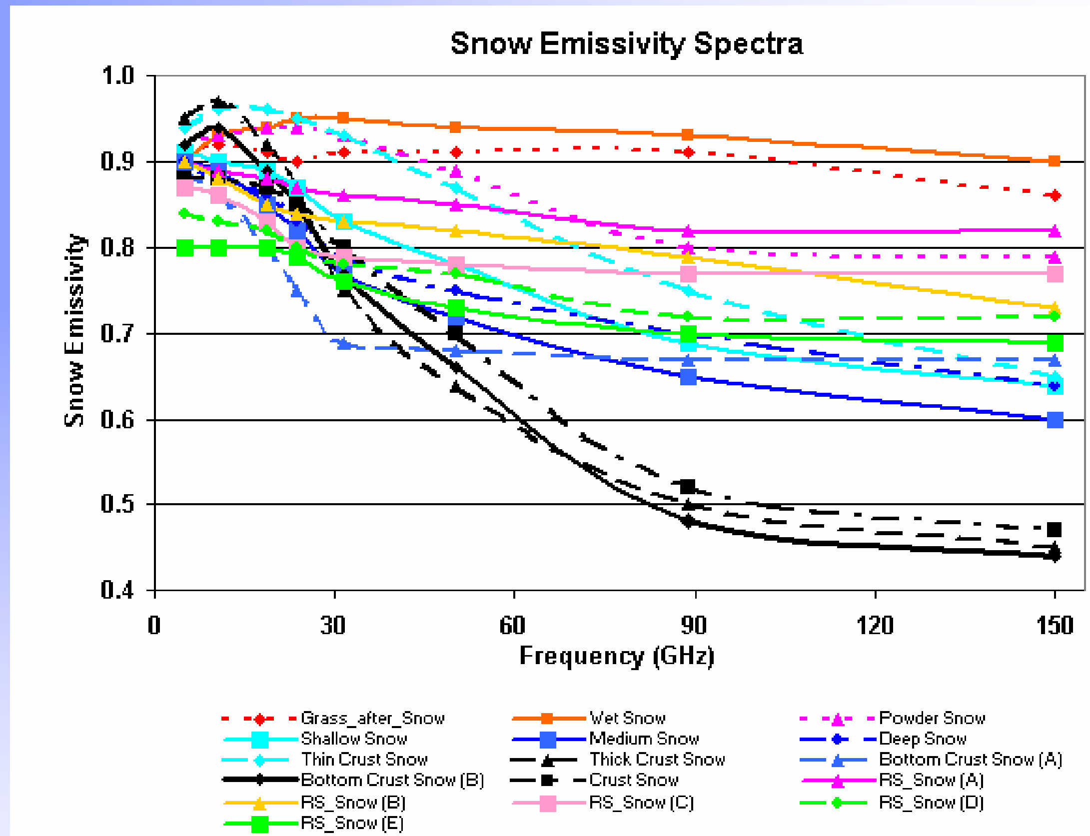
Conditions using LandEM:

over land: f < 80 GHz, use LandEM; f >= 80 GHz, e_v = e_h = 0.95

over snow: f < 80 GHz, use LandEM; f >= 80 GHz, e_v = e_h = 0.90

Microwave empirical snow and ice surface emissivity model

(1) Emissivity Database:



(2) Snow type discriminators are used to pick up snow type and emissivity:

$$DI_i = a_0 + \sum_{j=1}^{N} a_{1j} T_{Bj} + \sum_{j=1}^{N} a_{2j} T_{Bj}^2 + (a_3 T_S) + a_3 \cos \vartheta$$

$T_{b,j}$ – e.g. AMSU window channel measurements

(3) Supported sensors: AMSU, AMSRE, SSMI, MSU, SSMIS

Radiative transfer solution under clear sky conditions

$$I_\nu = \sum_{k=1}^{n} (\tau_{\nu,k-1} - \tau_{\nu,k}) B_\nu(T_k) + \varepsilon_\nu B_\nu(T_s)\tau_{\nu,n} + (1-\varepsilon_\nu)\tau_{\nu,n} I_s{}^{\downarrow}{}_\nu + \rho_\nu \tau_\nu (p_s, \theta_{sun})(F_{0,\nu}/\pi)\cos\theta_{sun}$$

(1) (2) (3) (4)

Transmittance at the kth level: $\tau_k = e^{-\sum_{j=1}^{k} \sigma_j / \cos(\theta)}$

σ_k – optical depth of the kth layer

(1) Contribution from the atmospheric absorbing gases;
(2) Contribution from surface emission attenuated by the atmosphere
(3) Surface reflected downwelling radiation from the atmosphere and space cosmic background attenuated by the atmosphere.

$$I^{\downarrow}{}_{s,\nu}(\theta_d) = \sum_{k=1}^{n} (\tau^{\downarrow}{}_{\nu,k}(\theta_d) - \tau^{\downarrow}{}_{\nu,k-1}(\theta_d)) B_\nu(T_k) + \tau^{\downarrow}{}_{\nu,n}(\theta_d) I_c$$

MW: θ_d – satellite zenith angle (specular surface reflection)
IR : θ_d – diffuse angle (Lambersian surface reflection)

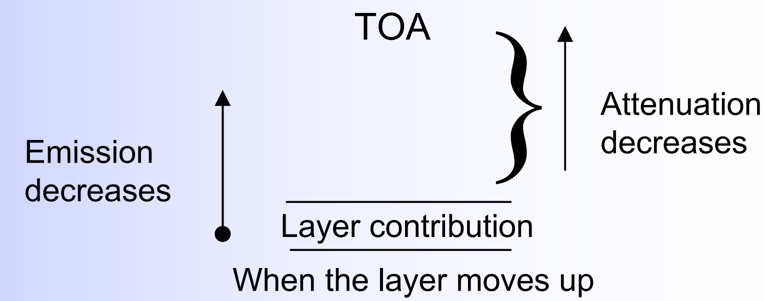
(4) Surface reflected solar radiation attenuated by the atmosphere

Weight function and radiance Jacobian

Weighting function:

$$w_k = (\tau_{\nu,k-1} - \tau_{\nu,k})/(z_{k-1} - z_k)$$

or $$w_k = (\tau_{\nu,k-1} - \tau_{\nu,k})/(\ln(p_k) - \ln(p_{k-1}))$$



Jacobian (radiance derivative with respect to state variables):

$$\frac{\partial I}{\partial x_k}$$ x_k - air temperature, water vapor mixing ratio, etc

(1) The atmosphere contribution (the first term of the RT solution in the previous slide) can be express as

$$I^{atm}{}_{\nu} = \sum_{k=1}^{n} \Delta z_k w_{k,\nu} B_{\nu}(T_k)$$

(a) The atmospheric contribution is the weighted sum of the source functions;
(b) The weighting function tells the relative importance of the contribution from each atmospheric layer.
(c) Weighting function does not depend on the layer thickness.

(2) The Jacobian is the radiance response to a unit perturbation of the state variable; it depends on the layer thickness.

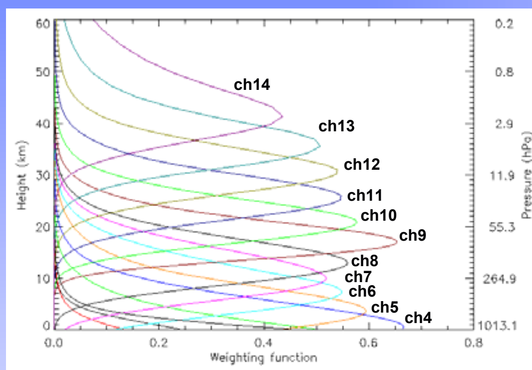
Weighting function and Jacobian profiles

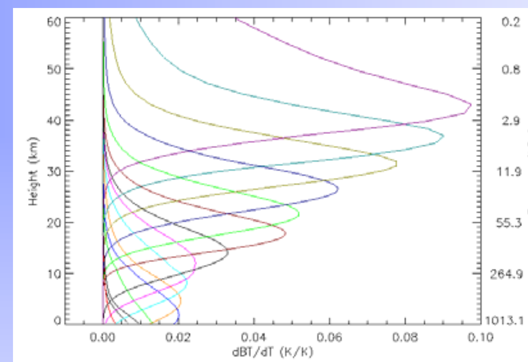
Weighting function

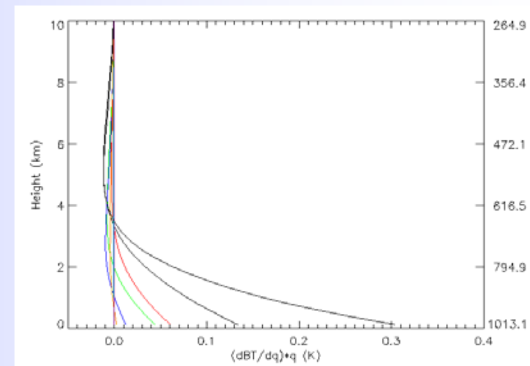
Jacobian with respect to temperature

Jacobian with respect to H2O mixing ratio times layer mixing ratio

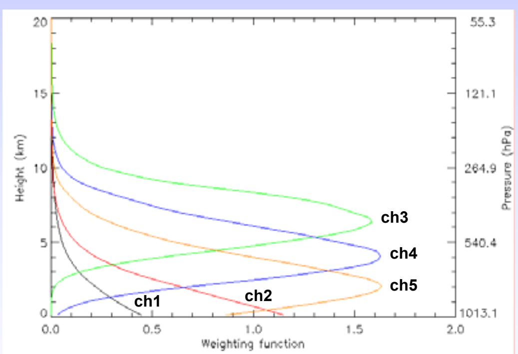
AMSUA

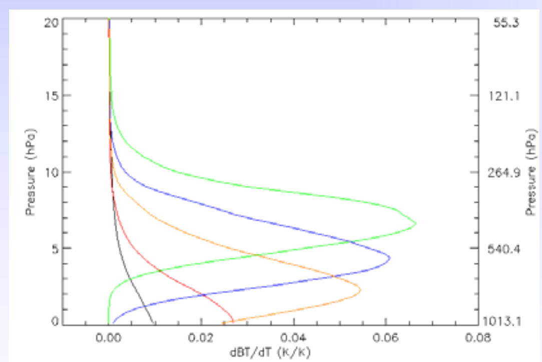


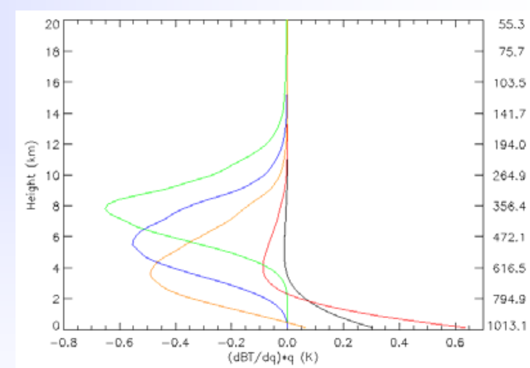




MHS







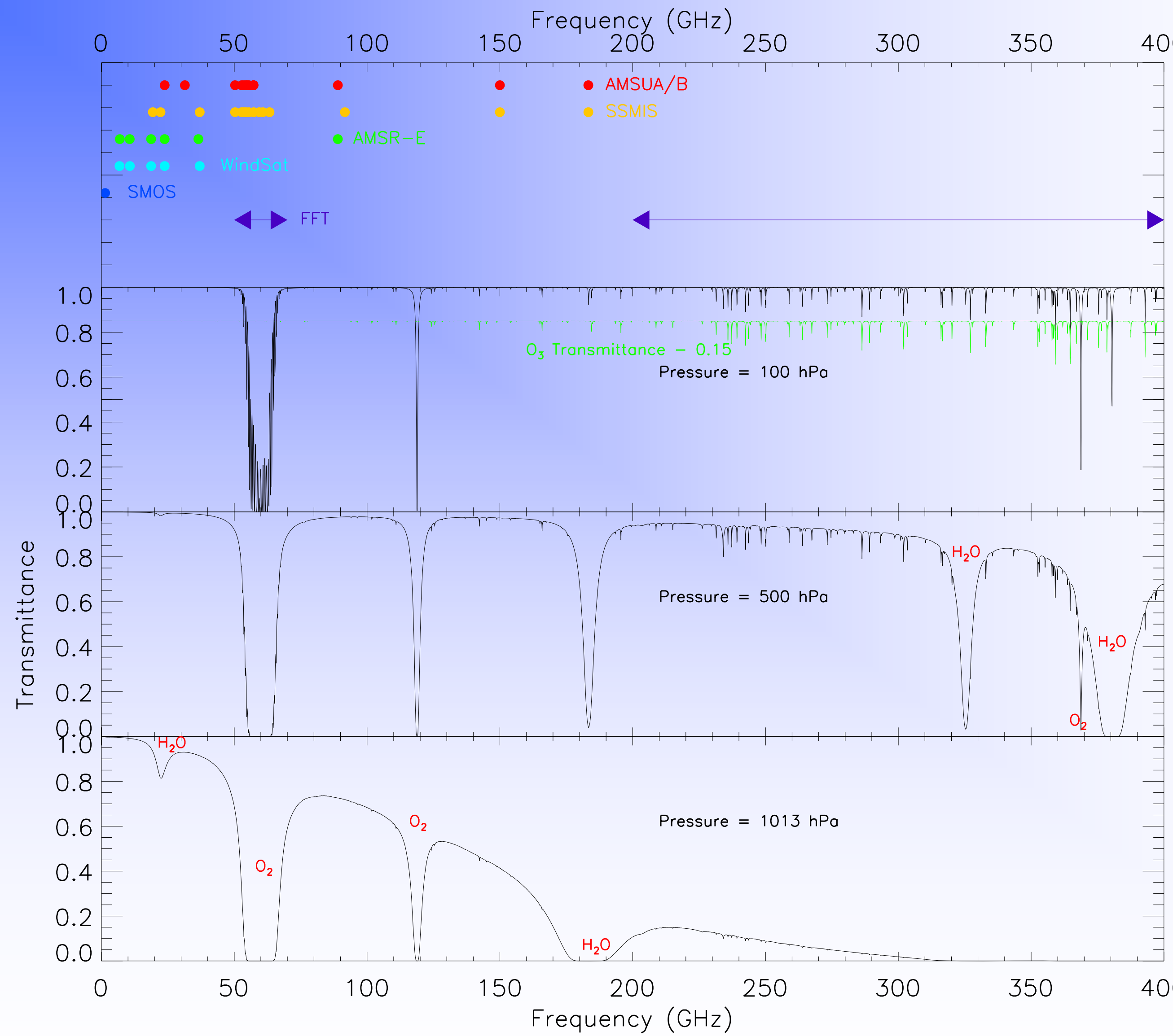
Computed with CRTM for US standard atmosphere over ocean surface

CRTM fast transmittance model

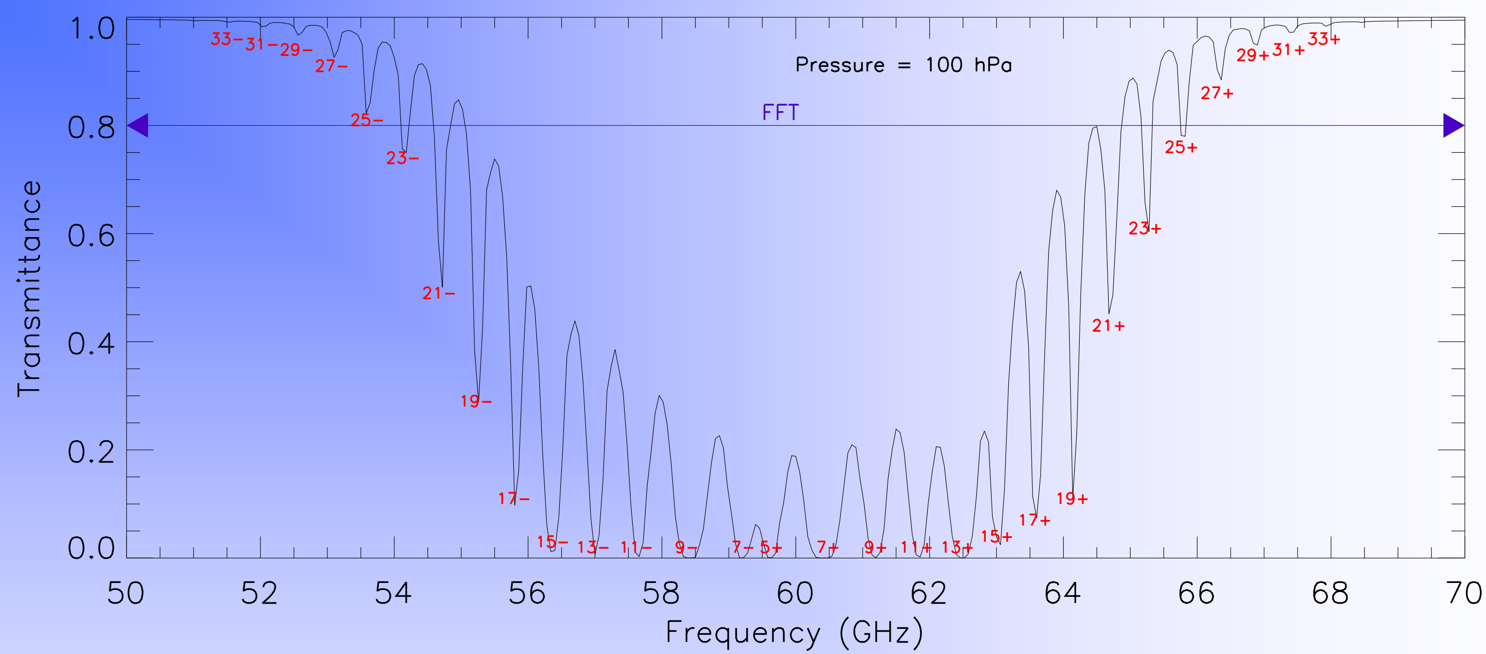
- Microwave and infrared spectra
- Why we need fast algorithm
- Transmittance parameterization

There are other well known transmittance models: RTTOV (UK), OSS (AER, Inc) and SARTA (UMBC). The JCSDA RT team is integrating them into CRTM.

Microwave transmittance spectrum

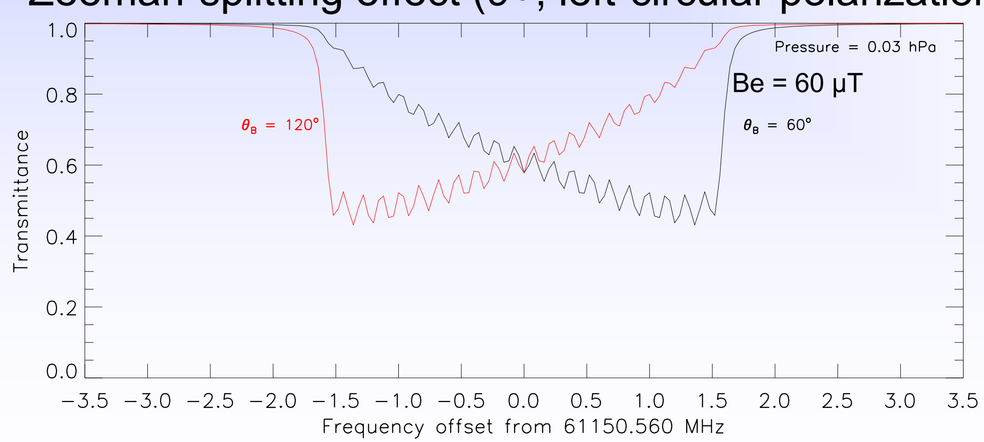


MW O_2 transmittance near 60 GHz

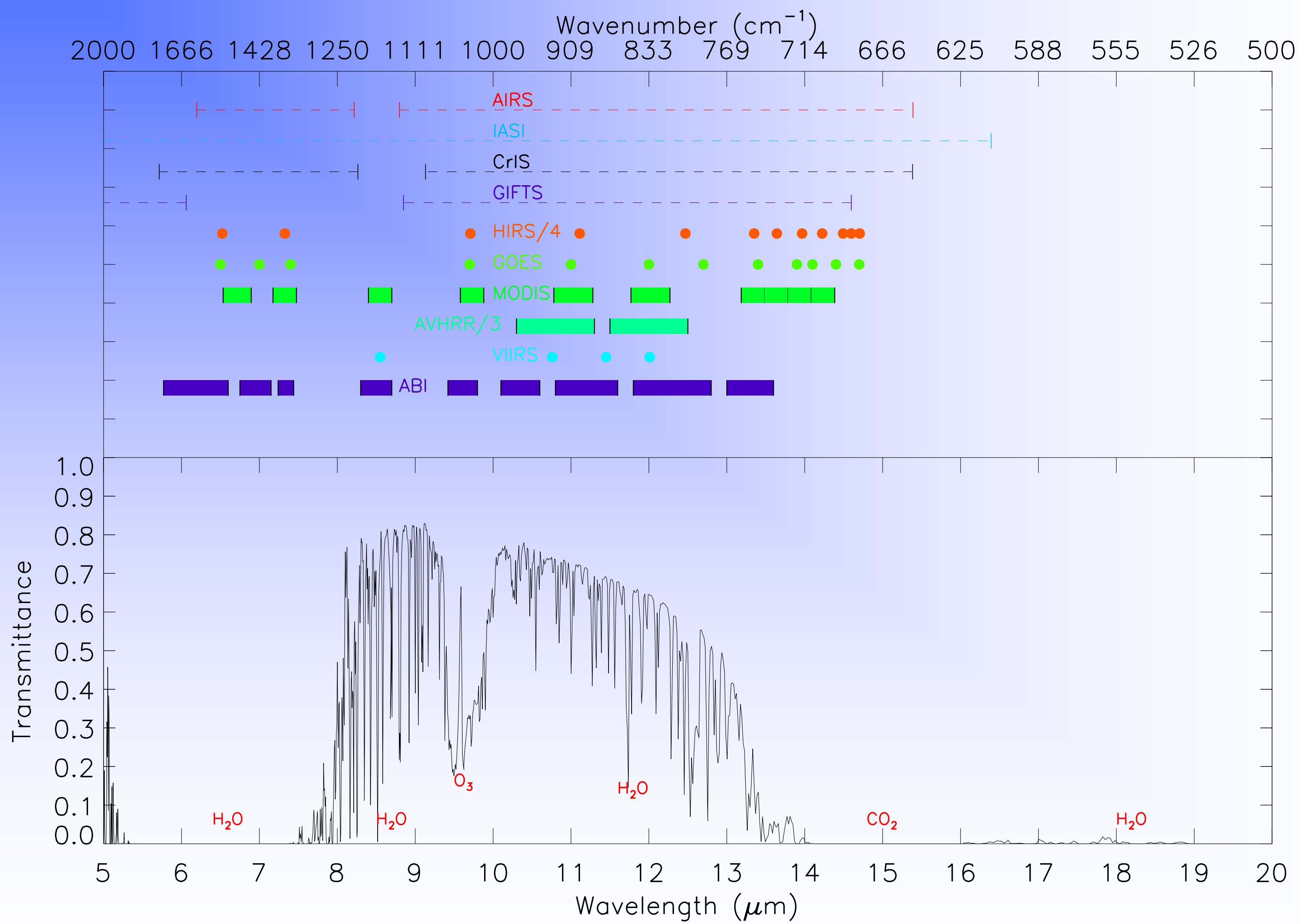


Zeeman-splitting effect (9+, left-circular polarization)

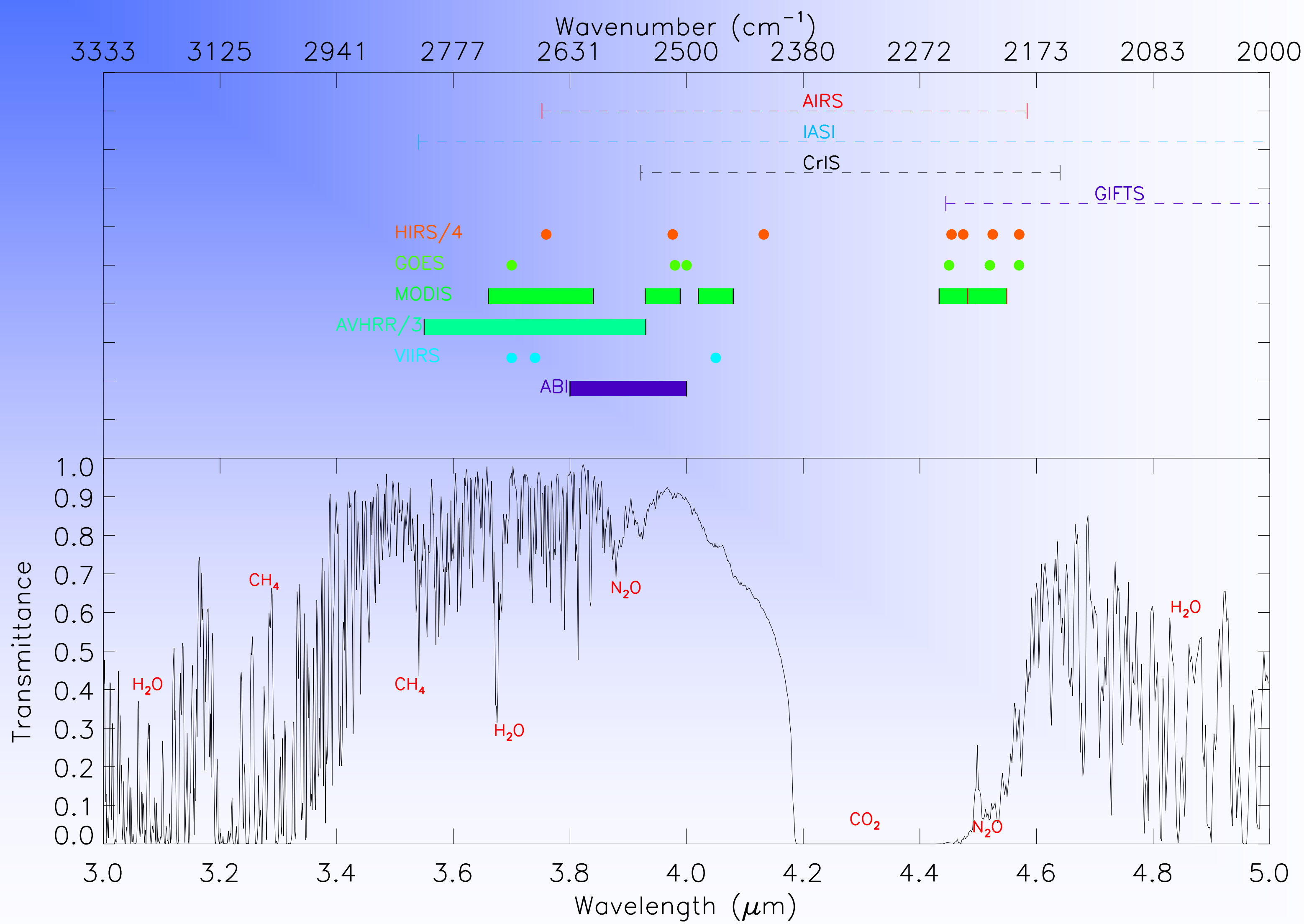
B_e – Earth's magnetic field strength.
θ_B – angle between the observation direction and the Earth's magnetic field.



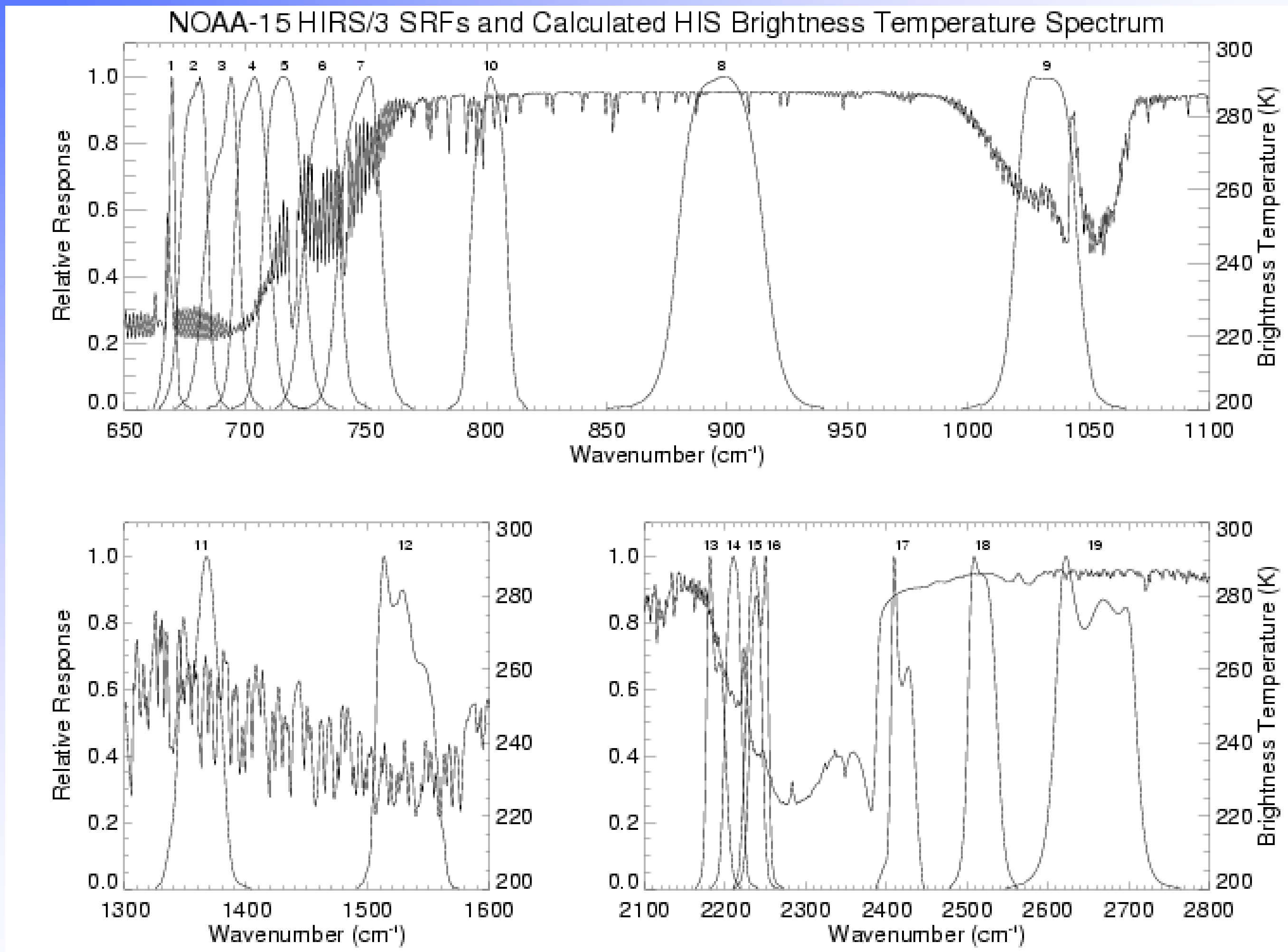
IR total transmittance spectrum (1)



IR total transmittance spectrum (2)



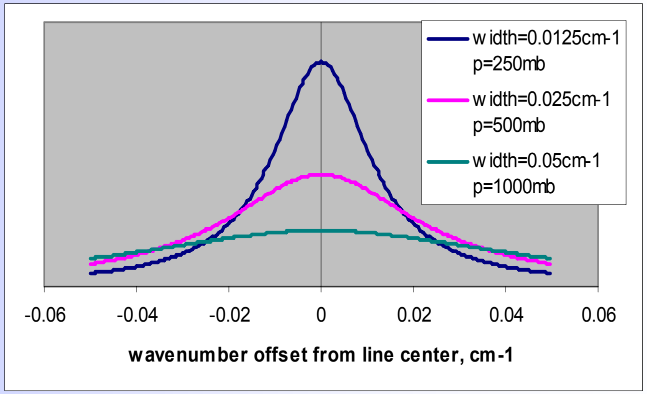
HIRS/3 spectral response functions (SRFs)



Fast transmittance module

Why need fast algorithm:

In the atmosphere, the absorption line width α_L is about 0.05 cm^{-1} at 1000mb and 0.0125 cm^{-1} at 250 mb. So, e.g., for a sensor with 1 cm^{-1} passband, a large number of monochromatic radiance calculations are needed for channel radiance simulation:



From Goody

$$I_{ch} = \sum_{i=1}^{N} I_{\nu_i} \phi_{\nu_i}$$

The channel's spectral response function (SRF)

Current operational systems can not handle such computation.

Solution: parameterize the optical depth $\sigma_{ch,k}$, defined as

$$\sigma_{ch,k} \equiv \ln(\tau_{ch,k-1} / \tau_{ch,k}) \quad \text{and} \quad \tau_{ch,k} \equiv \sum_{i=1}^{N} \tau_{\nu_i,k} \phi_{\nu_i}$$

Parameterization:

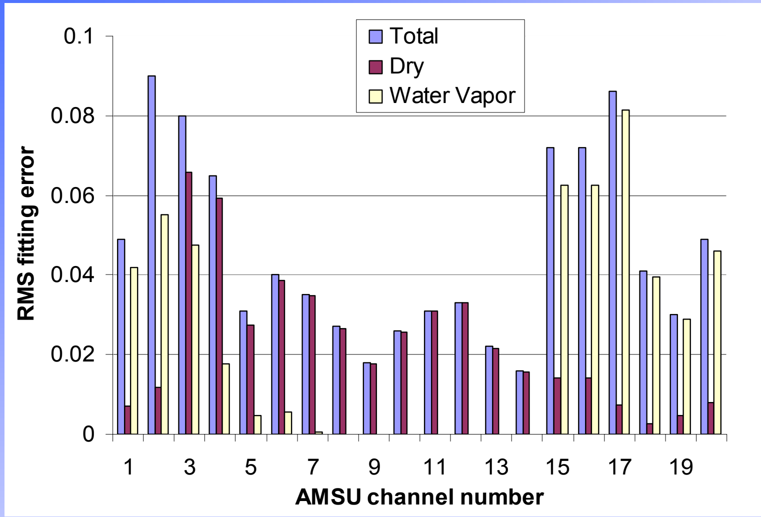
Predictors, such as T and water vapor

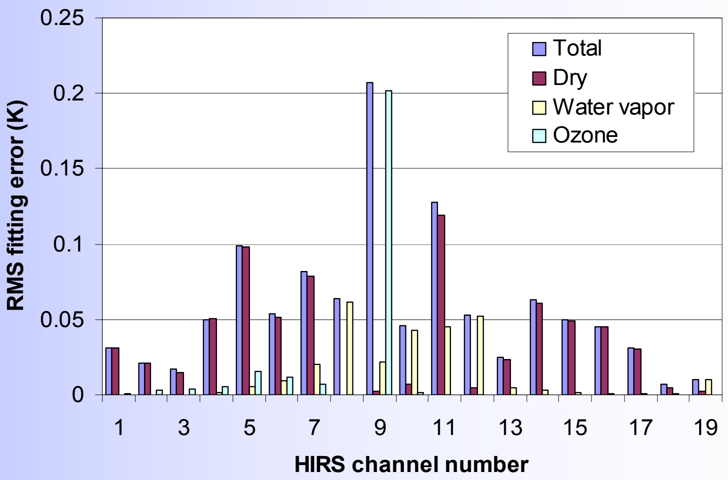
$$\sigma_{ch,k} = c_{0,k} + \sum_{i=1}^{n} c_{i,k} x_{i,k}$$

The radiance is then computed with the regular RT equation without the need for spectral integration.

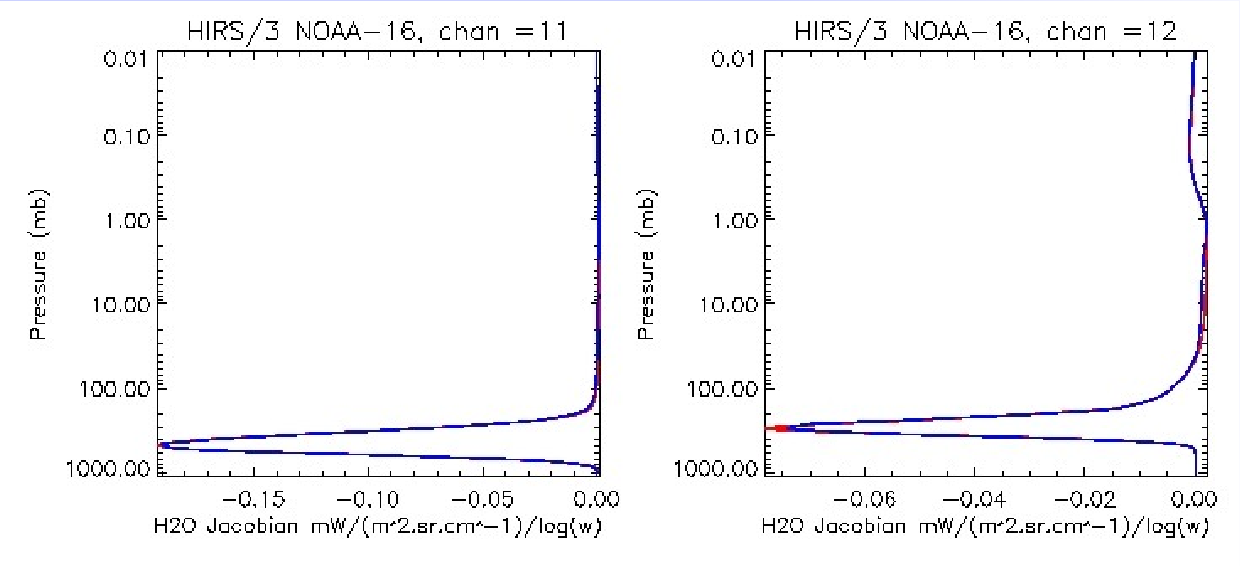
CRTM currently is implemented with the OPTRAN transmittance algorithm

Radiance errors due to transmittance model uncertainty

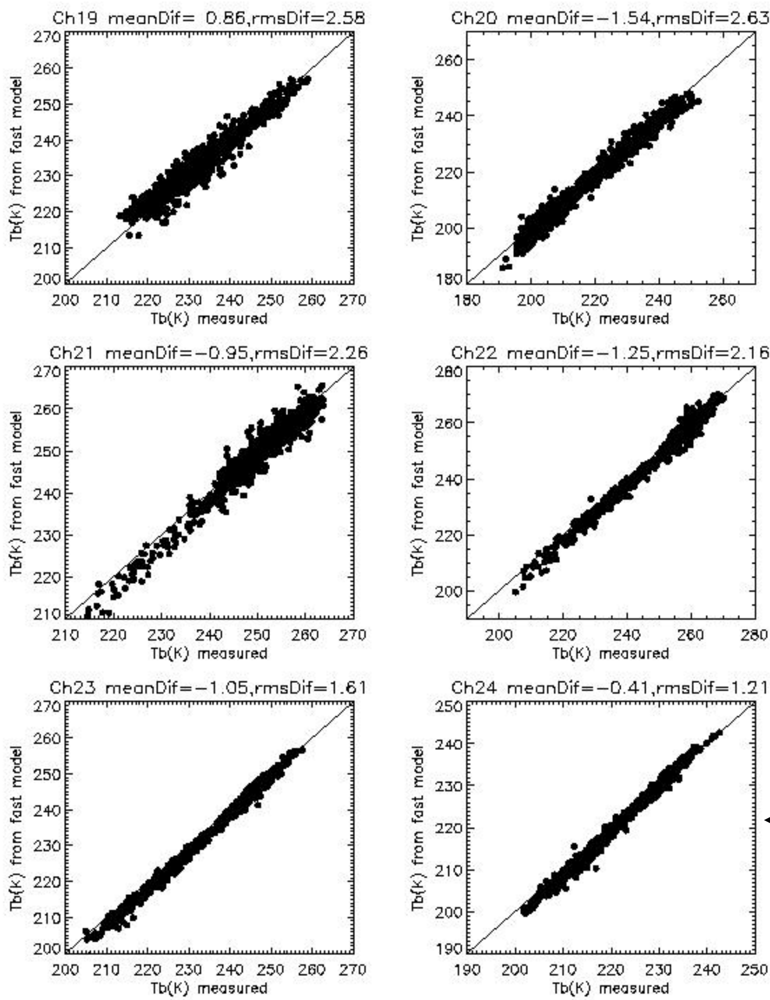


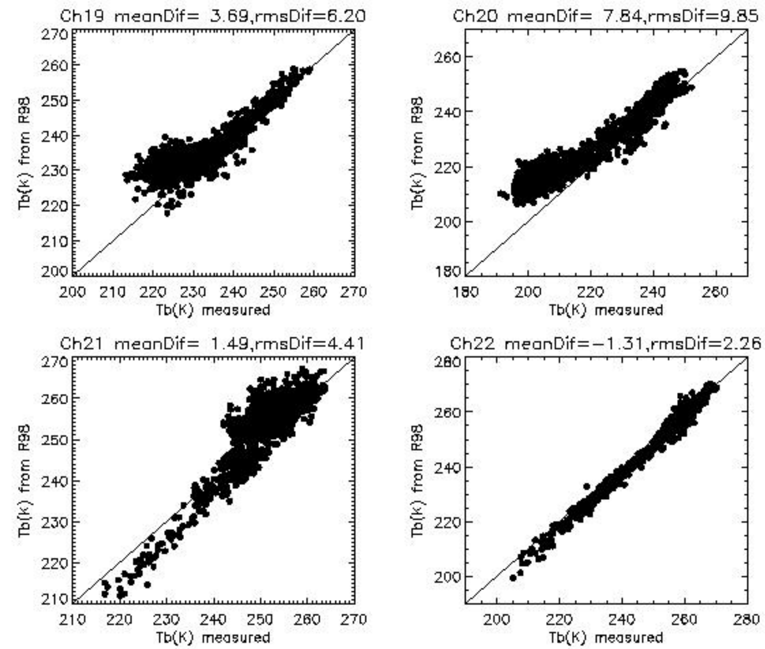


Radiance Jacobians with respect to water vapor, compared with LBLRTM



Comparison between SSMIS observations and simulations with/without Zeeman-effect





Without including Zeeman-effect in the model.

Channels 23 & 24 are not affected by Zeeman-splitting

Collocated temperature profiles for model input are retrievals form the SABER experiment.

Sample size: 1097 samples

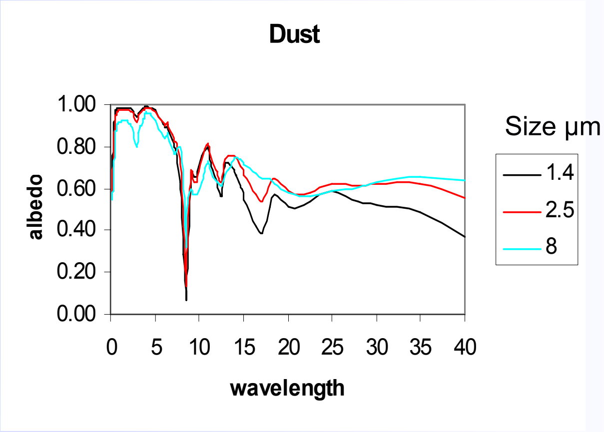
CRTM cloud absorption/scattering, aerosol absorption/scattering and RT solution modules

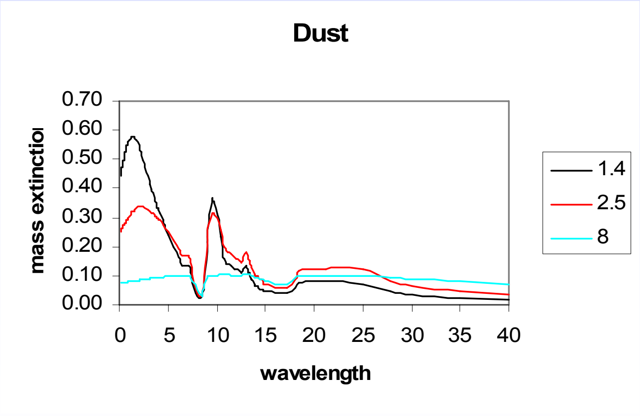
Cloud absorption/Scattering module (provide cloud optical parameters for RT solution module)

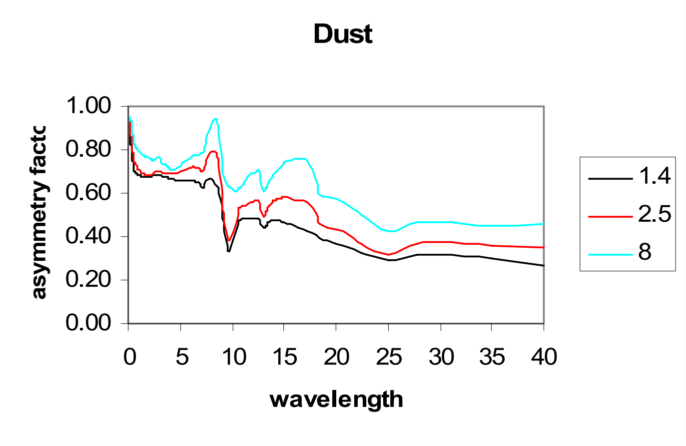
- Six cloud types: water, ice, rain, snow, graupel and hail
- NESDIS/ORA lookup table (Liu et al., 2005): mass extinction coefficient, single scattering albedo, asymmetric factor and Legendre phase coefficients. Sources:
  - IR: spherical water cloud droplets (Simmer, 1994); non-spherical ice cloud particles (Liou and Yang, 1995; Macke, Mishenko et al.; Baum et al., 2001).
  - MW: spherical cloud, rain and ice particles (Simmer, 1994).

Aerosol absorption/Scattering module (provide aerosol optical parameters for RT solution module)

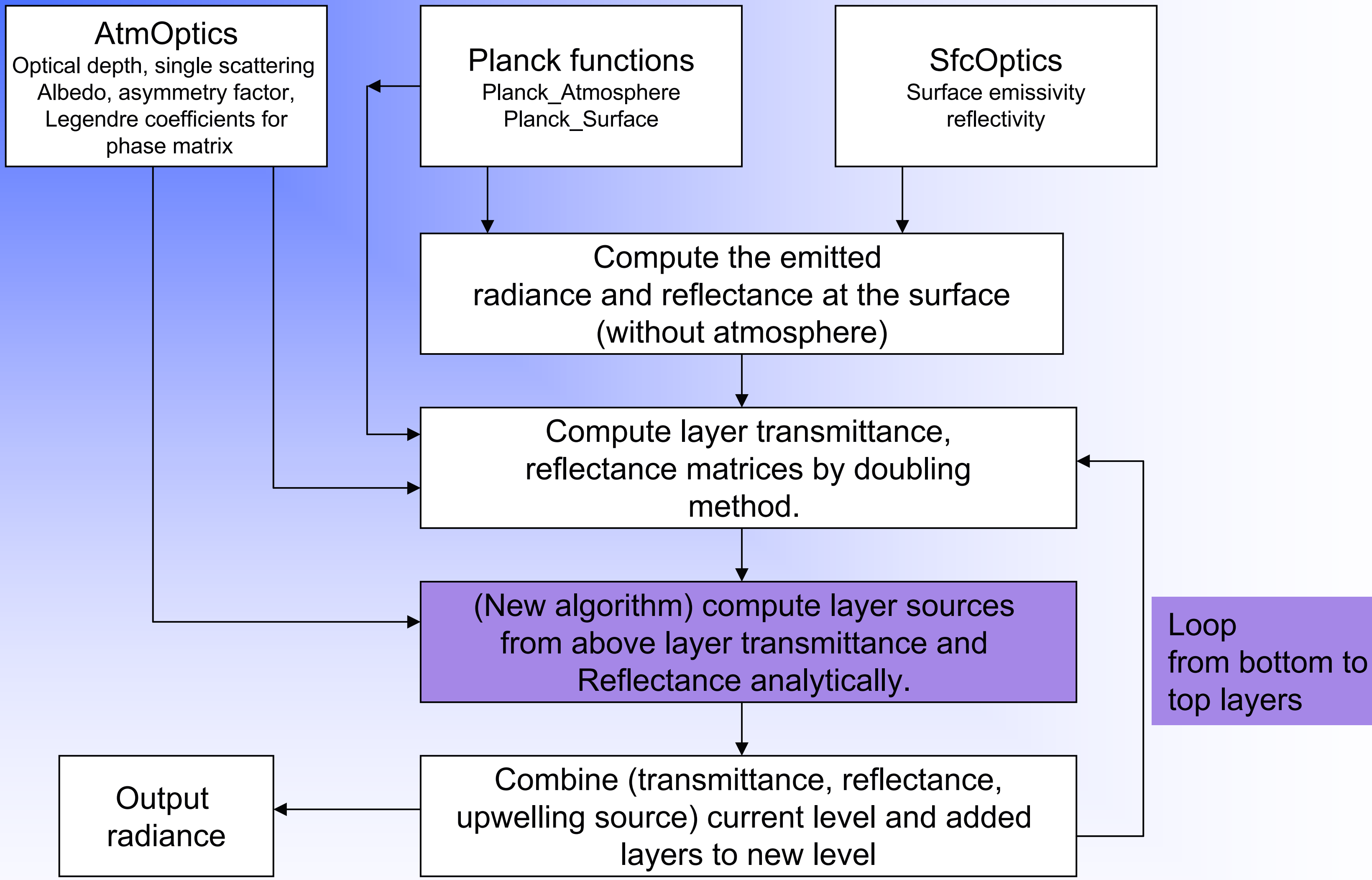
- GOCART aerosol profiles: Dust, Sea Salt, Organic carbon, Black carbon, Sulfate.
- Optical parameter lookup table.







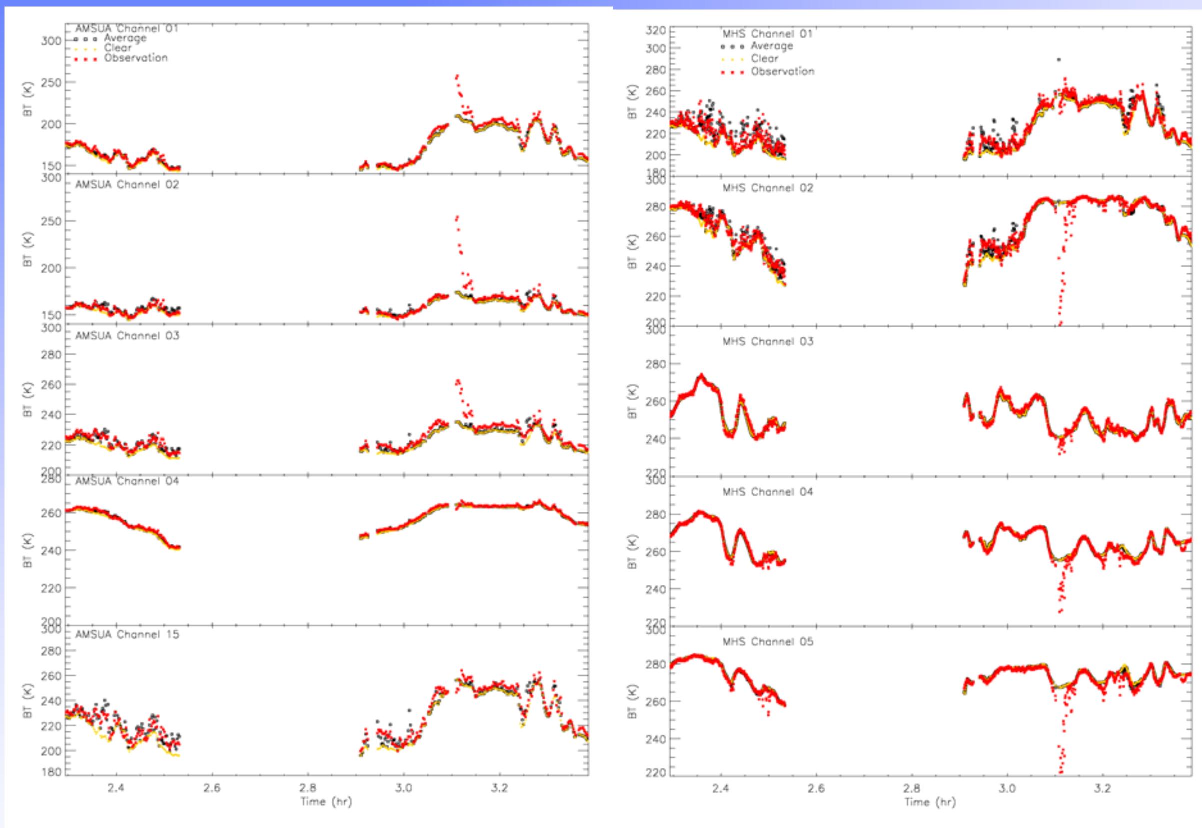
RT solution for cloud/aerosol scattering environment: Advanced Doubling-Adding Method (ADA)

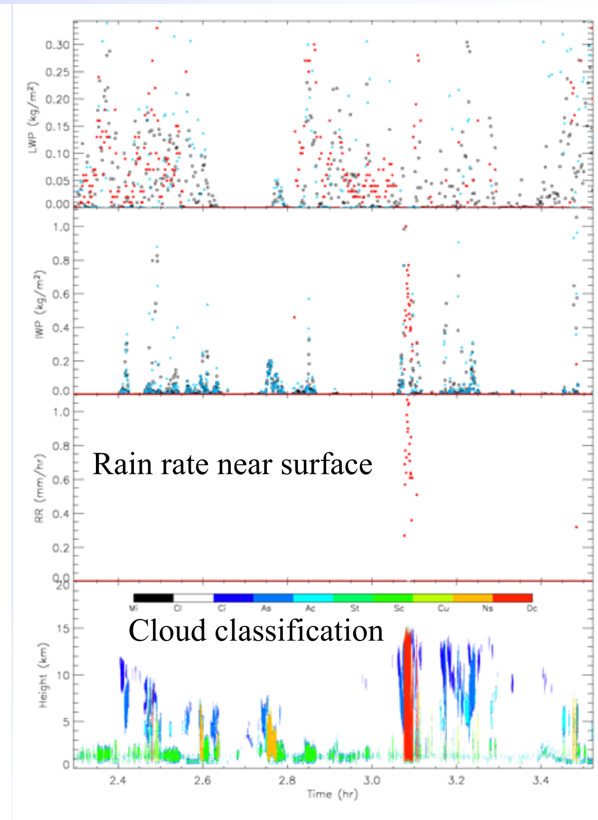


1.7 times faster then VDISORT; 61 times faster than DA
Maximum differences between ADA,VDISORT and DA are less than 0.01 K.

Liu and Weng, 2006

Time series of AMSU-A, MHS observations versus CRTM simulations using CloudSat data (non-precipitating weather)



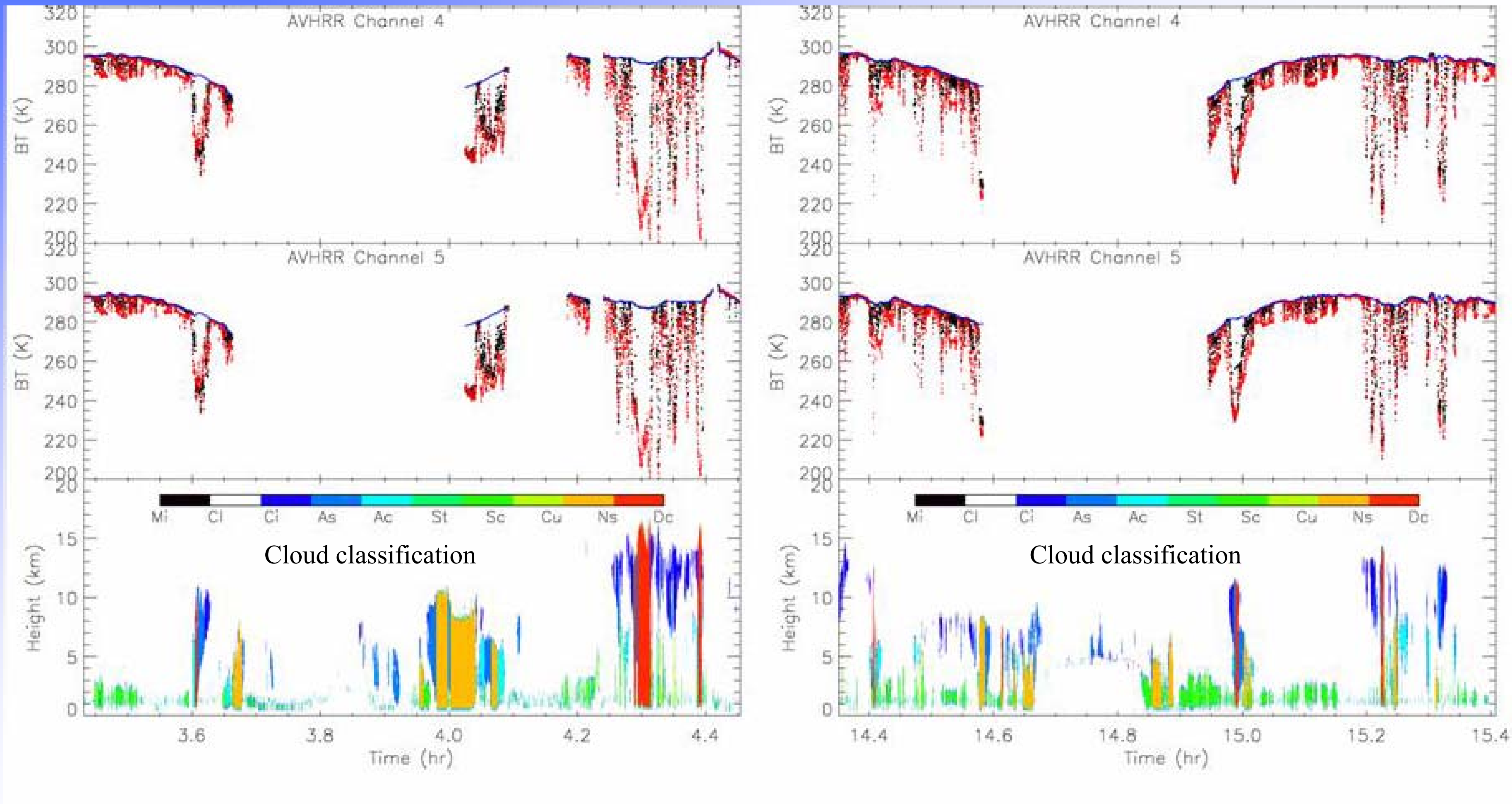


Model simulations with cloud component on (black) and off (yellow);
AMSU-A and MHS observations (red), 07/27/2006
Model input: cloud liquid/ice content and particle size profiles from CloudSat

Upper two panels:
Red – AMSUA+MHS retrievals
Black – derived from CloudSat Radar

Large differences between Observations and simulations near 3.1 are due to CloudSat data that exclude precipitation.

Time series of AVHRR observations versus CRTM simulations using CloudSat data

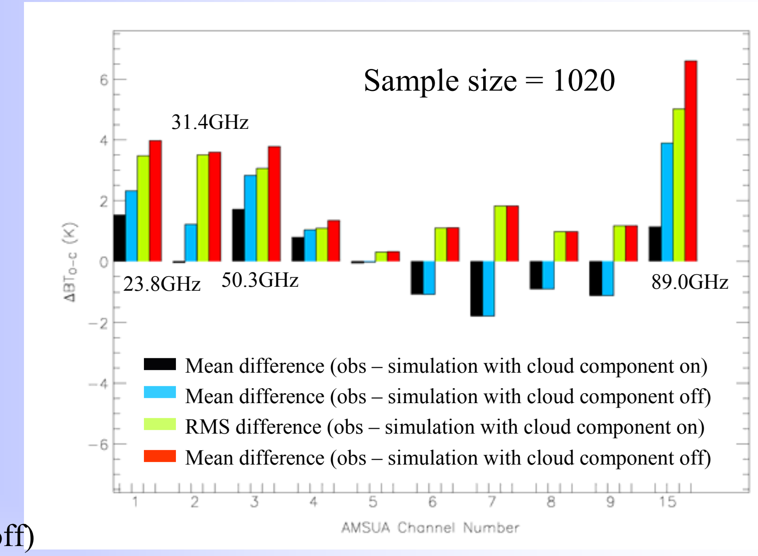


- **Model simulations with cloud component on (black) and off (blue)**
- **AVHRR observations (red)**
- **Model input: cloud liquid/ice content and particle size profiles from CloudSat**

Mean & RMS difference between AMSU-A, MHS & AVHRR observations and simulations under cloudy conditions

The differences between the observations and simulations are significantly reduced with the inclusion of modeling cloud absorption and scattering

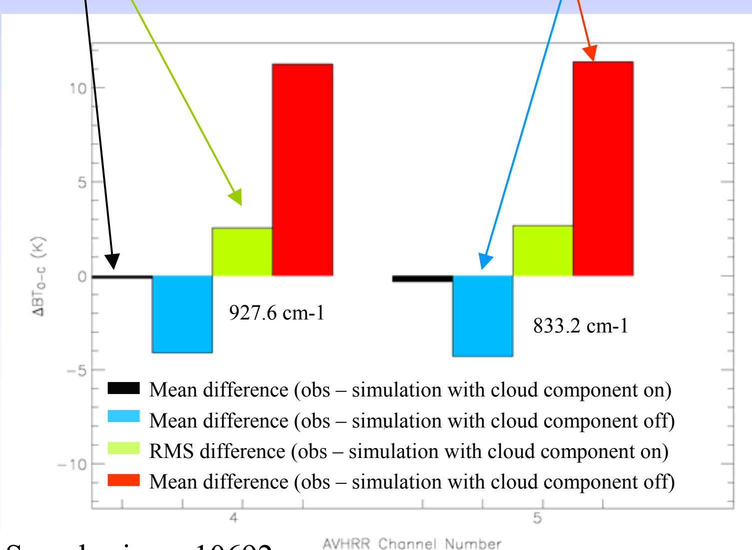
AMSU-A



Mean & RMS Diff. (obs. – simulation w. cloud on)

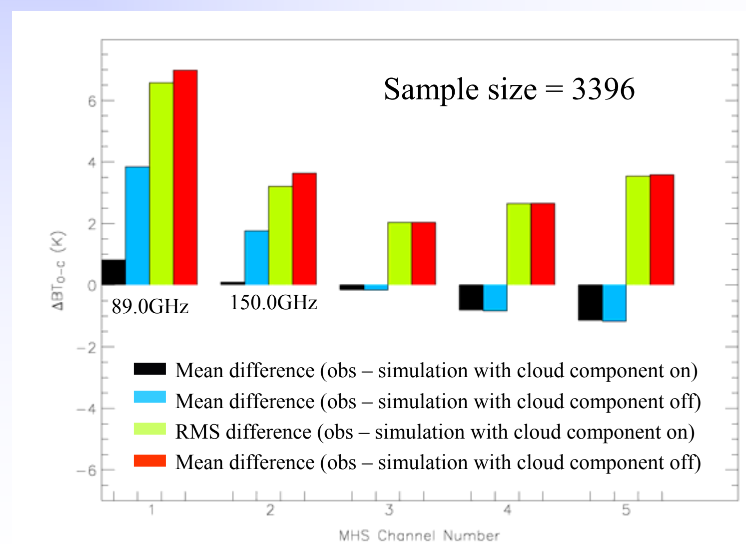
Mean & RMS Diff. (obs. – simulation w. cloud off)

AVHRR



Sample size = 10692

MHS



CRTM Tangent-linear, Adjoint and K-Matrix (Jacobian) models

Order of developing these models:

$$FW \Rightarrow TL \Rightarrow AD \Rightarrow K\_Matrix$$

- Forward model:

$$Y = F(X)$$

$$X = \{x_1, x_2, ..., x_n\}^T$$ State vector (n state variables)

$$Y = \{y_1, y_2, ..., y_m\}^T$$ Radiance vector (m channels)

FW model may be considered as a composition of a set of K functions:

$$F(X) = F^K(...F^2[F^1(Z^0 = X)]) \qquad (1)$$

or expressed with the help of the intermediate variables Z^l as

$$Z^1 = F^1(Z^0 = X), ..., Z^l = F^l(Z^{l-1}), ..., Z^k = F^k(Z^{k-1}) = Y \qquad (2)$$

In CRTM, many of the functions $Z^l = F^l(Z^{l-1})$ are coded explicitly with subroutines or functions:

Z^{l-1} – input variable vector
Z^l – output variable vector

- Tangent-linear (TL) model:

$$\delta Y = H(X)\delta X$$

δX perturbation of ***X***

δY perturbation of ***Y***

H(***X***)***,*** a matrix with elements:

$$H_{i,j} = \{\frac{\partial y_i}{\partial x_j}, i = 1, m;\ j = 1, n\}$$ Jacobians (derivatives of the radiance with respect to a state variable)

Applying the chain rule to (1):

$$\delta Y = H^K(Z^{K-1})H^{K-1}(Z^{K-2})\cdots H^l(Z^{l-1})\cdots H^1(Z^0)\delta Z^0$$

or expressed as

$$\delta Z^1 = H^1(Z^0 = X)\delta Z^0, \ldots\ldots, \delta Z^l = H^l(Z^{l-1})\delta Z^{l-1}, \delta Z^k = H^k(Z^{k-1})\delta Z^{k-1}$$

where $H^l(Z^{l-1}) = \{H^l{}_{i,j}\} = \{\frac{\partial F^l{}_i}{\partial Z^{l-1}{}_j}\}$

In CRTM, $\delta Z^l = H^l(Z^{l-1})\delta Z^{l-1}$ is developed by differentiating the FW counterpart $Z^l = F^l(Z^{l-1})$.

$Z^{l-1}, \delta Z^{l-1}$ Inputs

δZ^l Output

Naming convention: (1) If ***Z*** is a FW variable name, then ***δZ*** is named is ***Z_TL***; (2) if ***Sub*** is the FW subroutine (or function) name, then the corresponding Tangent-linear subroutine (or function) is named as ***Sub_TL***.

- Adjoint (AD) model:

Let the function *J*(.) transforms the radiance vector ***Y*** (=***F***(***X***)) into a scalar:

$J = J(\boldsymbol{Y})$ (Note, CRTM does not include this function)

The gradient of *J* with respect to ***X***:

$$\nabla_X J = \nabla_X Y \nabla_Y J$$ --- AD model

where $\nabla_X J = [\frac{\partial J}{\partial x_1},...,\frac{\partial J}{\partial x_n}]^T$ $\nabla_Y J = [\frac{\partial J}{\partial y_1},...,\frac{\partial J}{\partial y_m}]^T$ both vectors

$\nabla_X Y = \{\frac{\partial y_i}{\partial x_j}, j = 1, n; i = 1, m\}$ transpose of the Jacobian matrix

Since $Z^l = F^l(Z^{l-1}), ..., Z^0 = X$ and $Y = F(X) = F^k(F^{k-1}[F^{k-1}...F^l(Z^l)])$

we have $J = J(F^k(F^{k-1}(F^{k-2}(...F^l(Z^l)))))$ So *J* is also a function ***Z^l***

Define Adjoint variables:

$$\delta^* Z^l \equiv \nabla_{Z^l} J, \quad \delta^* X \equiv \nabla_X J, \quad \delta^* Y \equiv \nabla_Y J$$

Adjoint (AD) model (Cont.):

So we can write $\nabla_X J = \nabla_X Y \nabla_Y J$ Into the form:

$$\delta^* X = H(X)^T \delta^* Y$$
$$= (H^1)^T (H^2)^T \cdots (H^K)^T \delta^* Y$$

or

$$\delta^* Z^{K-1} = (H^K)^T \delta^* Y, \delta^* Z^{K-2} = (H^{K-1})^T \delta^* Z^{K-1}, \cdots, \delta^* Z^1 = (H^2)^T \delta^* Z^2, \delta^* X = (H^1)^T \delta^* Z^1$$

In CRTM, $\delta^* Z^{l-1} = (H^l)^T (Z^{l-1}) \delta^* Z^l$ is developed by reversing the order of the operations in the TL counterpart $\delta Z^l = H^l (Z^{l-1}) \delta Z^{l-1}$, following a set of rules (Giering and Kaminski, 1998)

Z^{l-1}	$\delta^* Z^l$	Input
	$\delta^* Z^{l-1}$	Output

Naming convention: if ***Z*** is the FW variable, the corresponding AD variable is named as ***Z_AD***; if ***Sub*** is the FW routine, then ***Sub_AD*** is named as the AD routine.

- The outputs of the TL and AD models:

TL model provides: $\delta Y = H(X)\delta X$

or $$\delta y_i = \frac{\partial y_i}{\partial x_1}\delta x_1 + \frac{\partial y_i}{\partial x_2}\delta x_2 + \ldots + \frac{\partial y_i}{\partial x_n}\delta x_n$$

$i = 1, m$

A summation over the state variable dimension

AD model provides: $\delta^* X = H(X)^T \delta^* Y$

or $$\delta^* x_j = \frac{\partial y_1}{\partial x_j}\delta^* y_1 + \frac{\partial y_2}{\partial x_j}\delta^* y_2 + \ldots + \frac{\partial y_m}{\partial x_j}\delta^* y_m$$

$j = 1, n$

A summation over the channel dimension

(1) $\{\delta^* y_i, i = 1, m\}$ are the AD model input variables. If they hold the values $\nabla_Y J (= \{\partial J / \partial y_i, i = 1, m\})$, then the output $\delta^* x_j$ hold the values $\nabla_X J$ --- the gradient of the *J* function.

(2) The TL and AD models do not provide Jacobians unless you set one of the inputs δx_j = 1 for the TL model and $\delta^* y_i$ = 1 for the AD model and set the reset of the delta inputs to zero and loop over the range of *i* or *j* --- a slow process.

- K-matrix model (compute the Jacobian matrix H):

K-matrix model is a slightly modified version of the AD model;
The modification is to separate the terms $(\partial y_i / \partial x_j)\delta^* y_i$ in the summation expression (see previous side). The K-matrix model may be expressed as

$$X\_K = [h_1 \delta^* y_1, h_2 \delta^* y_2, ..., h_m \delta^* y_m]$$

where $h_i = [\frac{\partial y_i}{\partial x_1}, \frac{\partial y_i}{\partial x_2}, ..., \frac{\partial y_i}{\partial x_n}]^T$

$$X\_K = \{\delta x^*_{i,j}; i = 1, m; j = 1, n\} = \{\frac{\partial y_i}{\partial x_j} \delta^* y_i, i = 1, m; j = n\}$$

$\delta^* y_1$, …, $\delta^* y_m$ are part of the K-matrix model input variables

To obtain *Jacobians,* set all the inputs $\delta^* y_1$, …, $\delta^* y_m$ *to one.*

Naming convention: in CRTM K-Matrix model, $\delta^* x_{i,j}$ is named as *x_K* and $\delta^* y_i$ as *y_K*.

CRTM user interface (1)

- CRTM software characteristic:
  - A set of Fortran subroutines and functions; users call CRTM from their application programs.
  - Structure variables (derived types) are used as input and output variables in the routine interfaces.
    - Benefits: (1) clean, (2) adding variables does not change the interfaces
    - Disadvantages: (1) variables are hidden under the structure variable name (e.g. X.x1 and X.x2 are hidden in the interface where only X appears; (2) it is easy to forget initialize those variables that are not interested to the users.
  - There is a set of coefficient data that are loaded during CRTM initialization stage. These data are included in several files, some of which are sensor/channel independent and some are sensor/channel dependent.

CRTM User Interface (2)

- CRTM Calling procedure (an example):

CRTM Initialization (load coefficient data)
In this process, the set of sensors supported in the following calls are determined.

↓

Memory allocation (using CRTM routines) for structure pointer array members

↓

Sensor selection (among the sensors specified during CRTM initialization)

↓

Set CRTM input variables

or / or / or / or

CRTM FW model | **CRTM TL model** | **CRTM AD model** | **CRTM KM model**

May be repeated

↓

Memory deallocation (using CRTM routines) for allocated structure pointer array members

↓

CRTM destruction (deallocate memory for internal variables)

Ending process

CRTM User Interface (3)

- CRTM initialization:

```
Error_status =  CRTM_Init( ChannelInfo          , &  ! Output
                           SensorID             , &  ! input
                           CloudCoeff_File      , &  ! input
                           AerosolCoeff_File    , &  ! input
                           EmisCoeff_File )          ! Input
```

SensorID : string array containing a list of the sensor IDs (e.g. hirs3_n17, amsua_n18, etc); array size = n_sensors specified by the user. The sensor IDs are used to construct the files for the SpcCoeff and TauCoeff file names:

e.g. hirs3_n17 will be used for hirs3_n17.SpcCoeff.bin and hirs3_n17.TauCoeff.bin

CloudCoeff_File, ArosolCoeff, EmisCoeff: file names for coefficient data file used for cloud, aerosol, surface emission and scattering calculations; the data are currently not sensor specific.

ChannelInfo : structure array (type ChannelInfor_type) containing sensor/channel information when returned from this function, array size = n_sensors.

- CRTM destruction:

Error_status = **CRTM_Destroy**(ChannelInfo)

(1) The memory for ChannelInfo is released, (2) all dynamically allocated memory for internal variables is released.

CRTM User Interface (4)

- Calling CRTM FW model (example):

```
Error_Status = CRTM_Forward( Atmosphere        , &  ! Input
                             Surface           , &  ! Input
                             GeometryInfo      , &  ! Input
                             ChannelInfo(2:2)  , &  ! Input
                             RTSolution )           ! Output
```

Atmosphere: structure array (type Atmosphere_type) containing data describing atmospheric state; array size = n_profiles (specified by user).
e.g. Atmospere(1)%temperature is an array of size n_layers (specified by user) containing the temperature profile for the first atmospheric profile.
Surface: structure array (type Surface_type) of size n_profiles containing data describing surface state. e.g. Surface(1)%water_temperature is a scalar variable for the water surface temperature.
GeometryInfo: structure array (type GeometryInfo_type) of size n_profiles containing satellite/sensor geometry information. e.g. the scalar GeometryInfo%Sensor_Zenith_Angle describes the sensor's zenith angle.
ChannelInfo: explained in the previous slide; ChannelInfo(2:2) is a one-element array containing sensor/channel information for the second sensor, which the user specified with the SensorID array during CRTM initialization.
RTSolution: structure array (type RTSolution_type) of size ChannelInfo(2:2)%n_Channels x n_profiles. e.g. the scalar variable RTSolution(1,1)%brightness_temepratures is the BT for the first channel and first profile.

CRTM User Interface (5)

- Calling CRTM TL model (example):

```
Error_Status = CRTM_Tangent_Linear( &
                 Atmosphere,      Surface        &  ! Input
                 Atmosphere_TL    Surface_TL   , &  ! Input
                 GeometryInfo                  , &  ! Input
                 ChannelInfo(2:2)              , &  ! Input
                 RTSolution, RTSolution_TL)         ! Output
```

Atmosphere_TL: structure array of the same type and dimension as Atmosphere containing TL variables corresponding to those in Atmosphere.
Surface_TL: structure array of the same type and dimension as Surface containing TL variables corresponding to those in Surface.
RTSolution_TL: structure array of the same type and dimension as RTSolution containing the resulting TL variables corresponding to those in RTSolution.

e.g. Atmosphere_TL(1)%temperature(10) is the temperature perturbation at layer 10 of the first profile and RTSolution_TL(1,1)%brightness_temperature is the response to the perturbations of those variables in Atmosphere_TL and Surface_TL for the first channel and first profile.

Note, when calling this routine, make sure all the perturbations are correctly set. E.g. Surface%wind_speed has a default value of 5 m/s. Since Surface_TL has the same type Surface_type as Surface, Surface_TL%wind_speed (perturbation) is automatically set to 5 m/s. So it must be set to zero or the intended value.

CRTM User Interface (6)

- Calling CRTM AD model (example):

```
Error_status = CRTM_Adjoint( Atmosphere, Surface                  , &  ! Input
                             RTSolution_AD                        , &  ! Input
                             GeometryInfo, ChannelInfo(2:2)       , &  ! Input
                             Atmosphere_AD, Surface_AD            , &  ! Output
                             RTSolution)                               ! Output
```

RTSolution_AD: structure array of the same type and dimension as RTSolution containing the AD variables corresponding to those in RTSolution.
Atmosphere_AD: structure array of the same type and dimension as Atmosphere containing AD variables corresponding to those in Atmosphere.
Surface_AD: structure array of the same type and dimension as Surface containing AD variables corresponding to those in Surface.

CRTM User Interface (7)

- Calling CRTM K-Matirx model (example):

```
Error_Status = CRTM_K_Matrix( Atmosphere, Surface              , &  ! Input
                              RTSolution_K                     , &  ! Input
                              GeometryInfo, ChannelInfo(2:2)   , &  ! Input
                              Atmosphere_K, Surface_K          , &  ! Output
                              RTSolution )                          ! Output
```

RTSolution_K: structure array of the same type and dimension as RTSolution containing the K variables corresponding to those in RTSolution.
Atmosphere_K: structure array of the type Atmosphere_type with dimension n_Channels x n_profiles, containing K variables corresponding to those in Atmosphere.
Surface_K: structure array of the type Surface_type with dimension n_Channels x n_profiles, containing K variables corresponding to those in Surface.

Note:
(1) compared with Atmosphere_AD and Surface_AD in the AD model, Atmosphere_K and Surface_K have an additional dimension n_Channels (as mentioned earlier, this dimension separates term $\frac{\partial y_i}{\partial x_j}\delta^* y_i$ from the other terms.

(2) To request a brightness temperature Jacobians $\partial BT_i / \partial x_j$, in this example, set RTSolution_K(i,1)%brightness_temperature = 1.0 and RTSolution_K%radiance = 0.0, i=1,n_Channels.

Explain why need setting one of Radiance_K & BT_K to one and the other to zero

In the FW routine Compute_FW(**X**, Rad, BT):

```
CALL Compute_Radiance(X, Rad)
CALL Compute_BrightnessTmperature(…, Rad, BT)
```

In TL routine Compute_TL(**X**, **X_TL**, Rad_TL, BT_TL):

```
CALL Compute_Radiance_TL(…, X_TL, Rad_TL)
CALL Compute_BrightnessTemperature_TL(…, Rad_TL, BT_TL)
```

In AD routine Compute_AD(**X**, **Rad_AD**, **BT_AD**, X_AD):

```
CALL Compute_BrightnessTemperature_AD(…, BT_AD, Rad_AD)
CALL Compute_Radiance_AD(Rad_AD, X_AD)
```

In Compute_BrightnessTemperature_AD(…, **BT_AD**, Rad_AD):

```
….
Rad_AD = Rad_AD + a*BT_AD
BT_AD = 0.
```

In Compute_Radiance_AD(**Rad_AD**, X_AD):

```
….
X_AD = X_AD + b*Rad_AD
Rad_AD = 0
```

Red color – input variables
Black color – output variables