A First Look at the CRTM Transmittance Coefficient Generation Package

Dr.-Ing. Patrick Stegmann

CRTM Monthly Meeting

July 30, 2020

Contributors:

- Code:
  - Isaac Moradi
  - Will McCarty
  - Bryan Karpowicz
  - Jim Rosinski
  - Haixia Liu
  - Yanqiu Zhu
  - Benjamin Johnson
- Testing:
  - Cheng Dang
- Visible/UV Code:
  - Mark Liu

Introduction

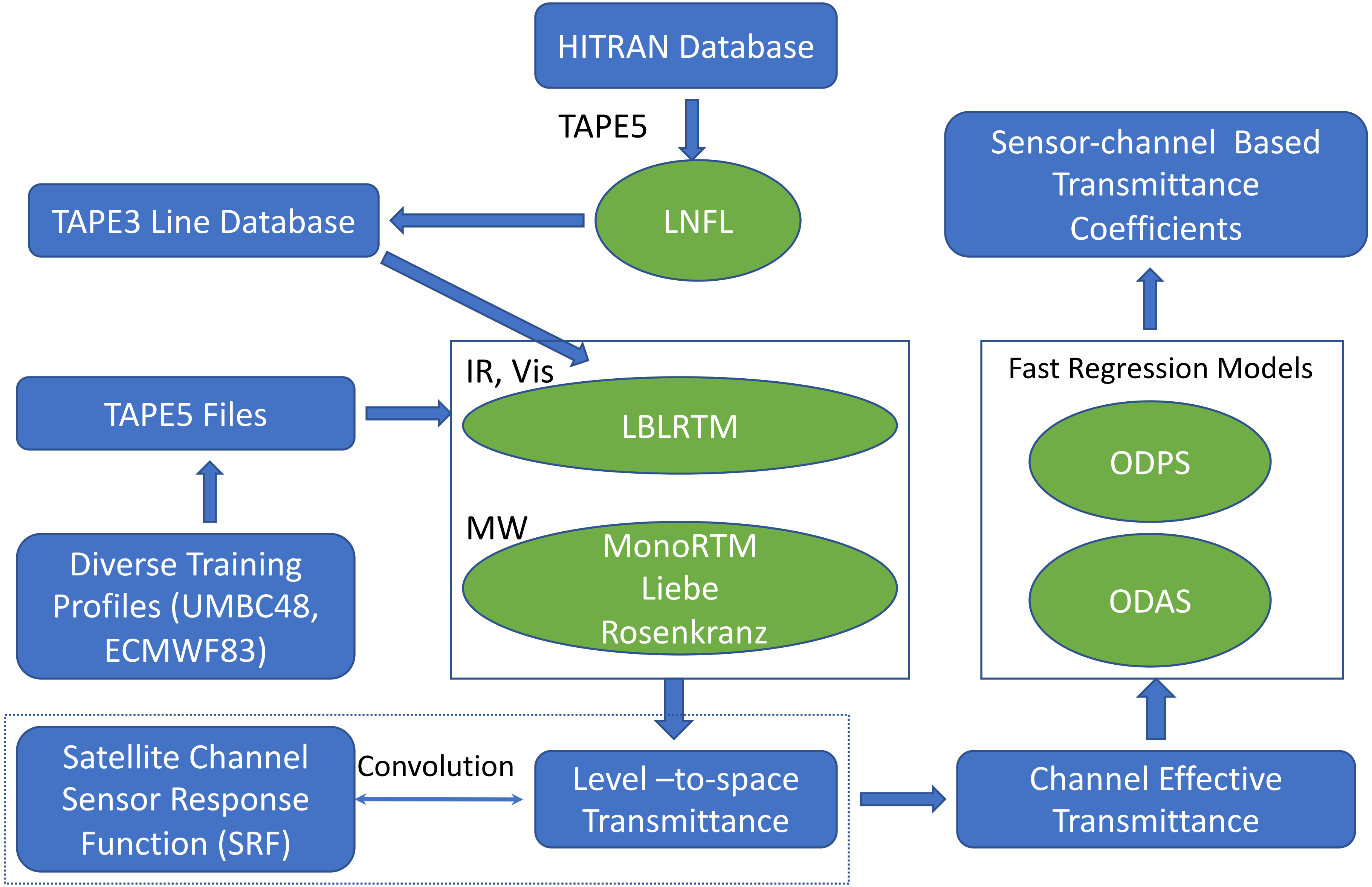
Development Issues

- Legacy Code is a Programming Paradigm of its own.
- No access to a working version of the Coefficient Generation code.
- Poor Documention.
- Available code in the CRTM SVN trunk was effectively broken and heavily fragmented:
  - Missing files
  - Mismatching interfaces
  - Deprecated data file formats
- Using LBLRTM requires expert knowledge.
- Clear case in favor of the JEDI approach:
  - Agile Development
  - Mandatory version control
  - Up-to-date software

High Level Process Overview

The CRTM transmittance coefficient generation process is conceptually easy and straightforward.

CRTM Coefficient Training Process in more Detail:

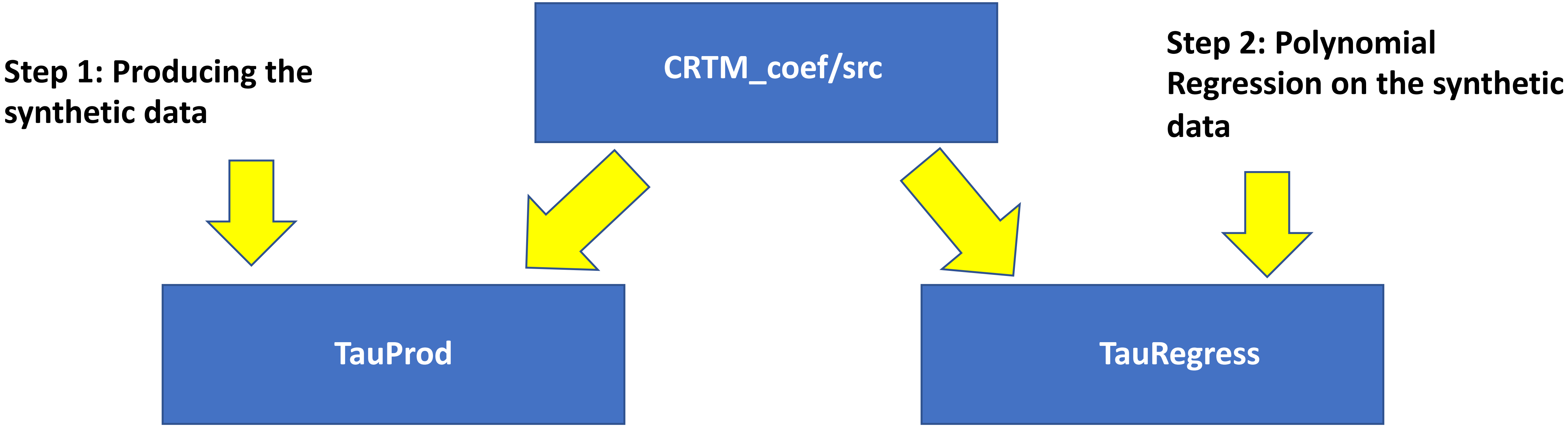


Limitations of the code from the CRTM SVN trunk

- Extremely complicated to set up and run.
- Completely inflexible:
    - No custom profile sets
    - No custom bands
    - No predictor variation between bands
    - Computation parameters both specified in configuration file and hard coded in Fortran module.
- Requires access to HPC infrastructure.
- Processing a single IR sensor takes at least one full workday on S4.

Repository Overview

Folder Structure (1/2)



- Liebe & Rosenkranz for MW transmittance
- LBLRTM I/O code for IR

- ODPS regression
- ODAS

Folder Structure (2/2)

- Working directory CRTM_coef/workdir
- Documentation in CRTM_coef/doc/UserGuide (work in progress)

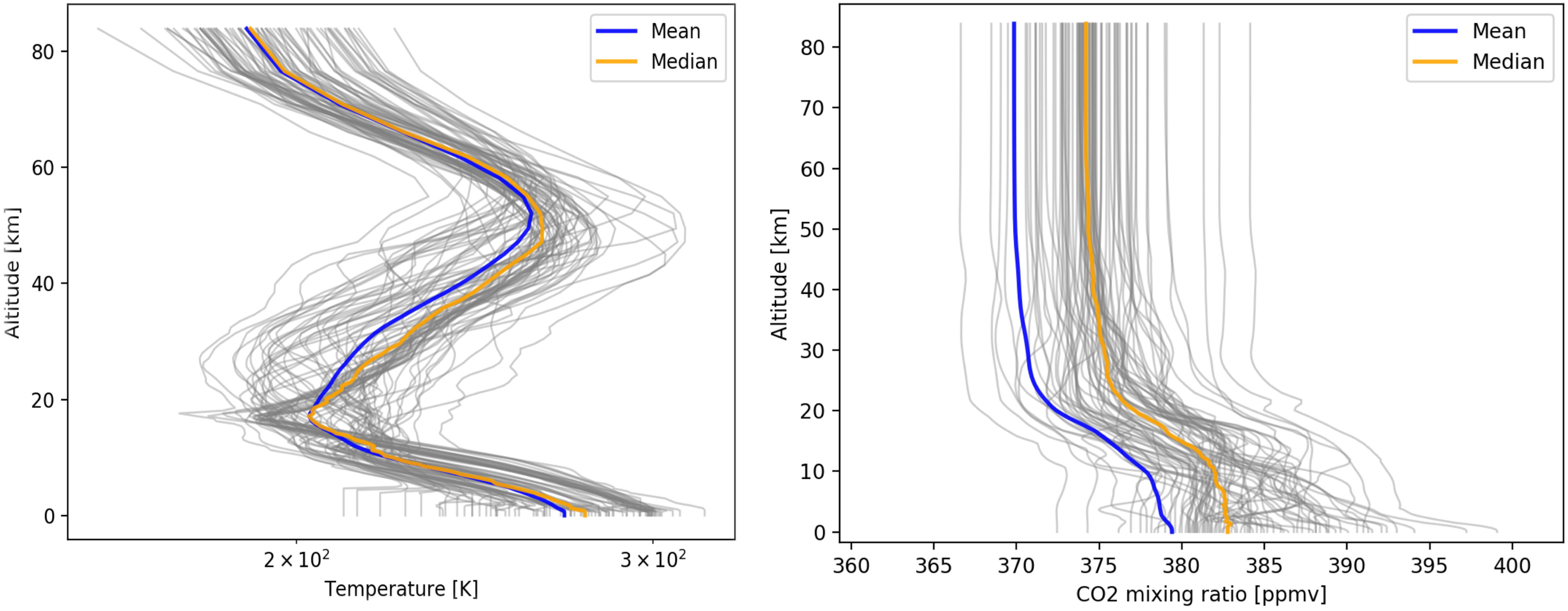
The IR Process, Step by Step

Part 1: Generating the Predictands as Synthetic Data

The LBLRTM step

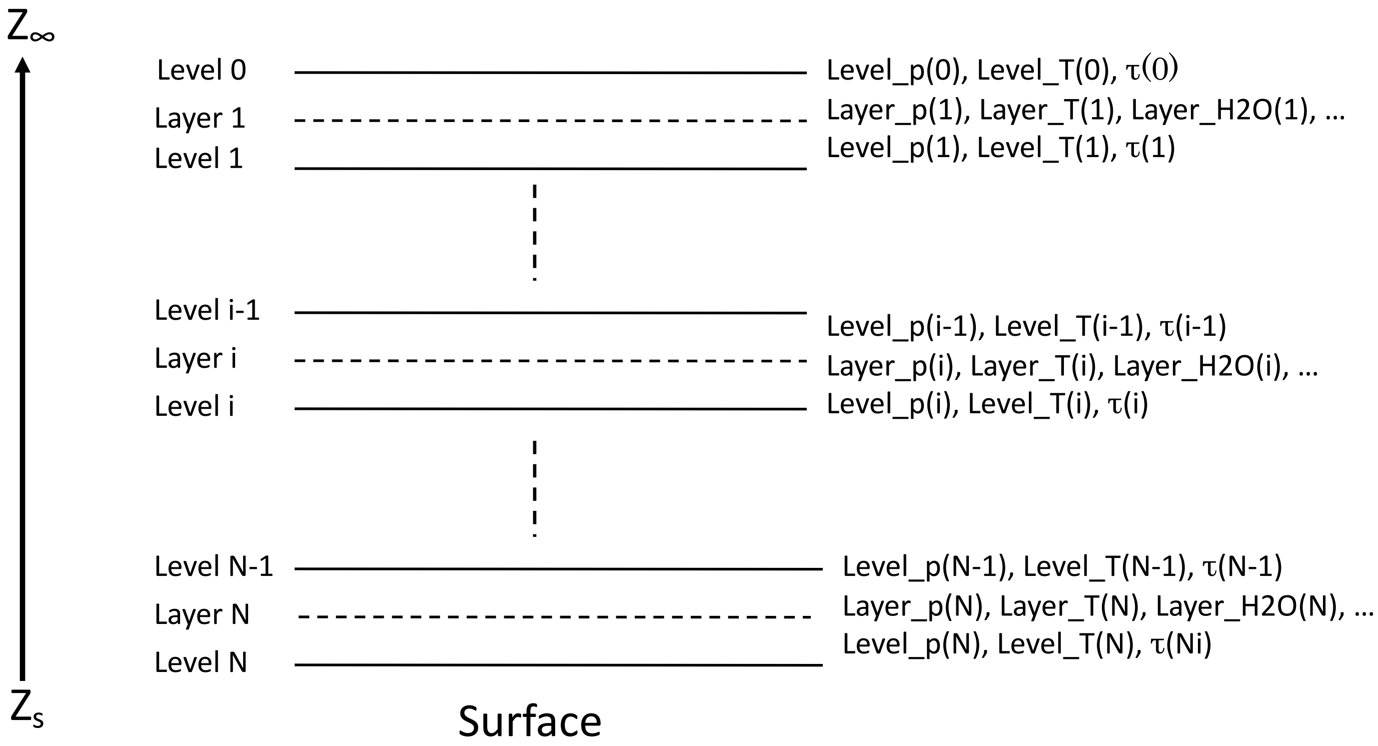
- Download and compile LNFL and LBLRTM from AER (tested with *LBLRTM v12.8, LNFL v3.7*)
- Clone *CRTM_dev*; Compile the *CRTM* and *LBLRTMIO* libraries.
- Compile and run *Create_LBLRTM_Input_Files* to transform the atmospheric profiles from NetCDF to *TAPE5* input files.
- Link the *TAPE5* files into the working directory.
- Run *run-lnfl* in the working directory to create *TAPE3* line data.
- Link the instrument *oSRF file* into the working directory.
- Adjust the configuration file *Transmittance_Production.processing_defaults*
- Submit the batch job *submit_process_tape5* to run LBLRTM.

Predictor Training Profile Set (ECMWF83)



Why not use e.g. reanalyis data instead?

Atmosphere Profile Layering Scheme



Example: AIRS science team level pressure definition (101 levels):

$$P_{lev}(i) = (Ai^2 + Bi + C)^{7/2}$$

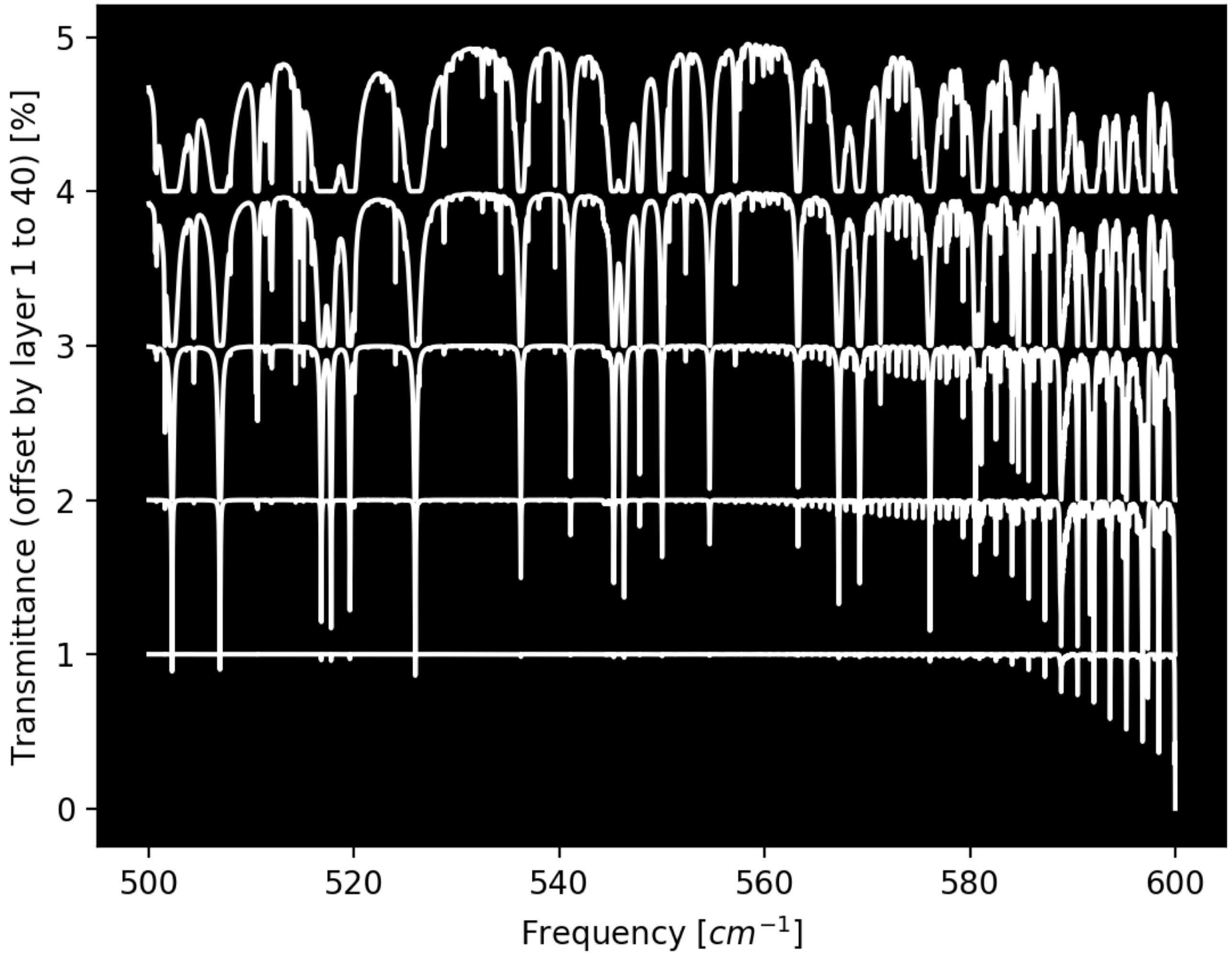
$$P_{lev}(1) = 0.005 \qquad P_{lev}(38) = 300.0 \qquad P_{lev}(101) = 1100.0$$

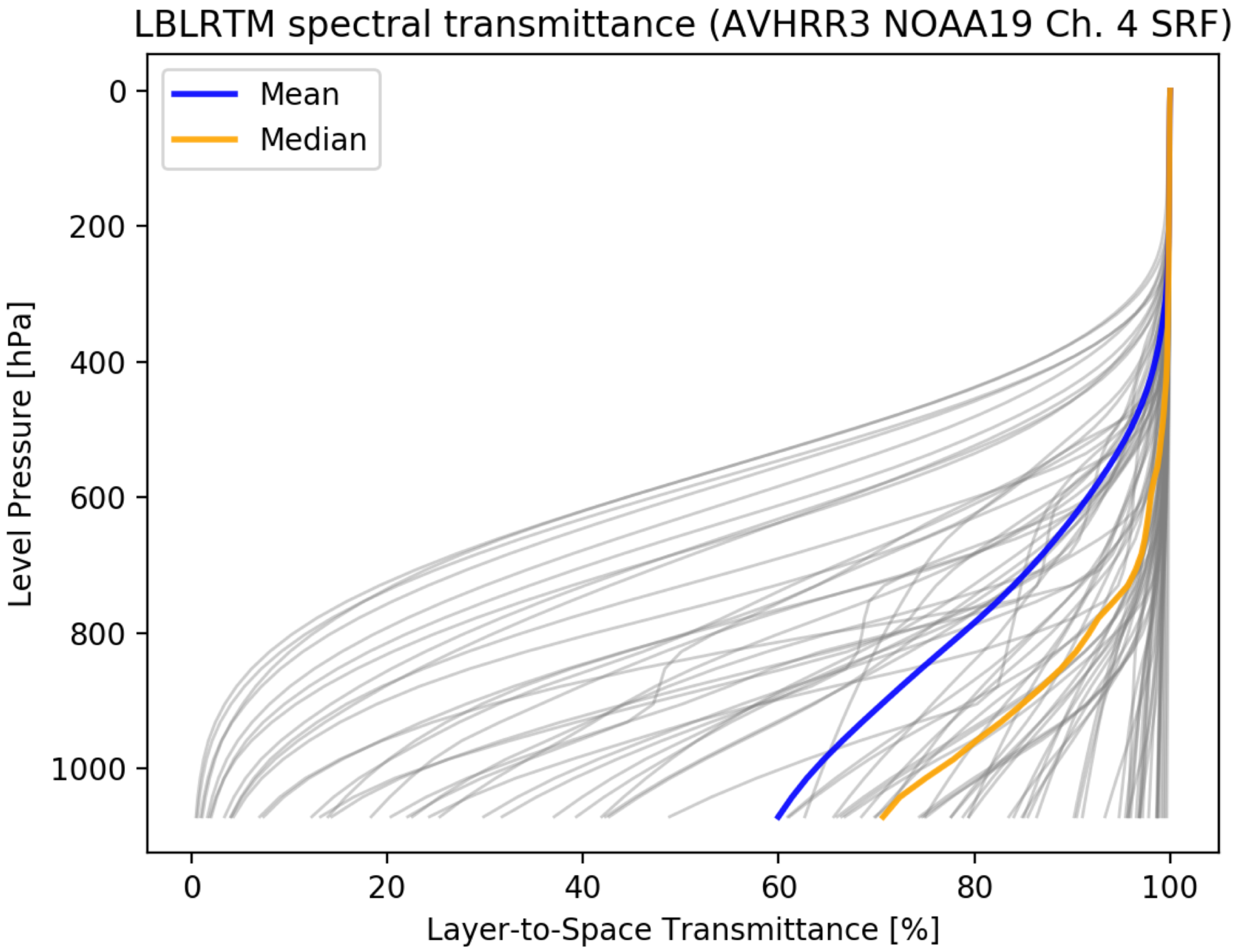
A=-1.55 × 10^{-4}, B=-1.55 × 10^{-4}, and C=7.45

Layer pressure definition:

$$P_{layer}(1:N) = (P_{lev}(2:N) - P_{lev}(1:N-1)) / \log(P_{lev}(2:N) / P_{lev}(1:N-1))$$

LBLRTM Monochromatic Layer-to-Space Transmittance Output at Fixed Frequency Intervals (TAPE20)

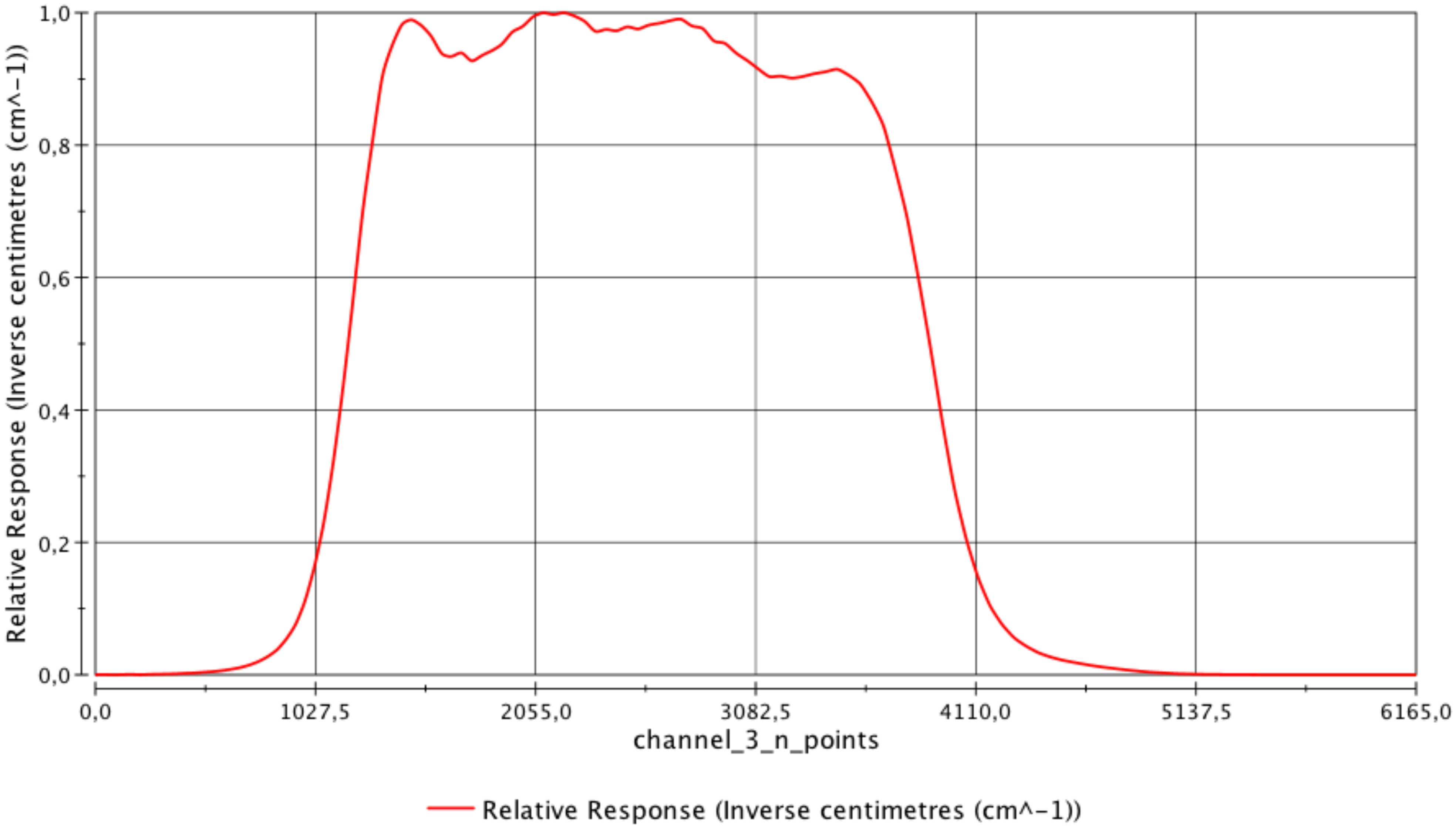


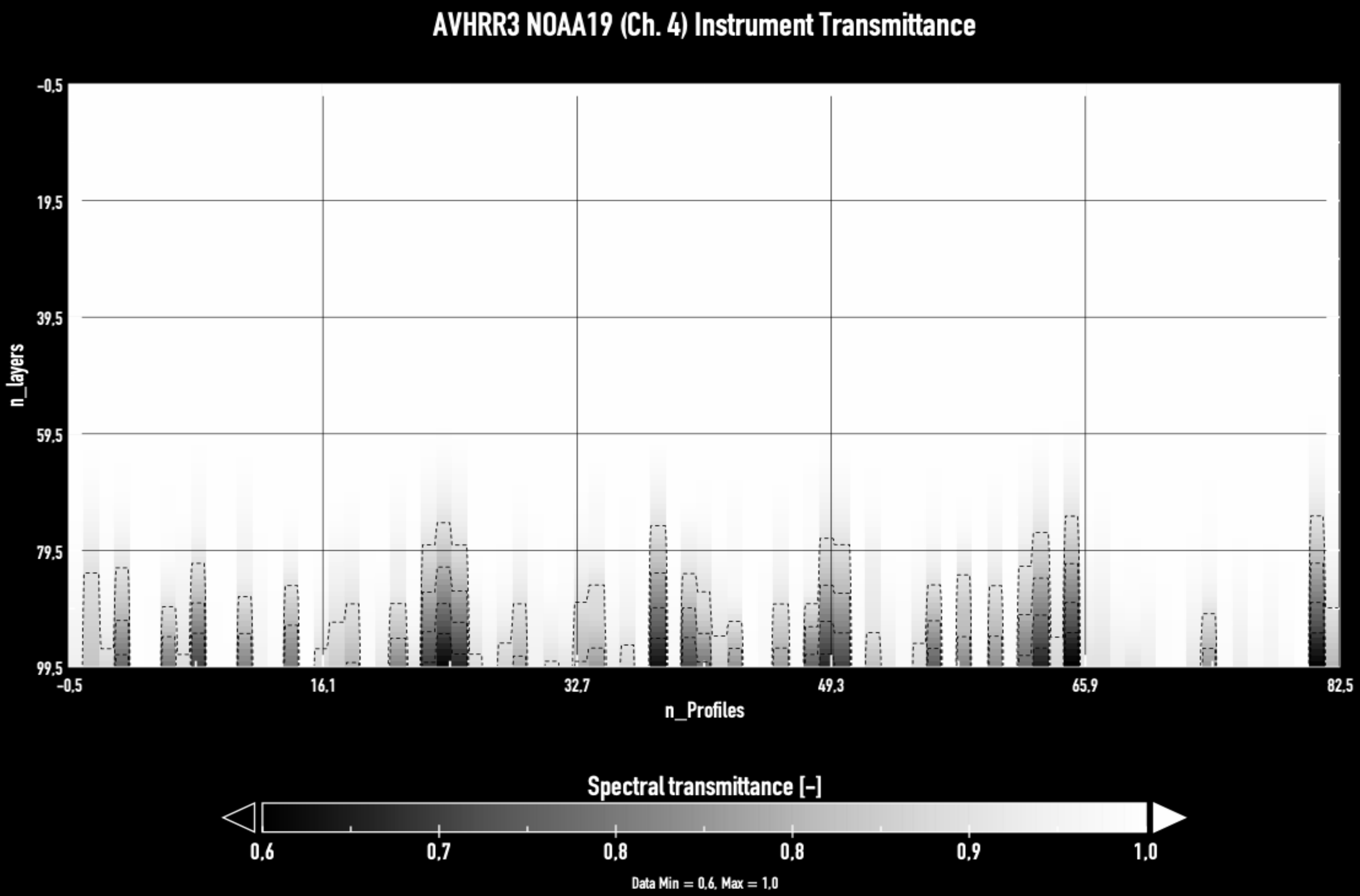


The SRF Convolution Step

- Compile *Create_ProcessControl_File, Convolve_TauSpc_with_SRF, and Check_ProcessControl_File*.
- Submit the *submit_process_TauSpc* script to perform the actual SRF convolution.
- Create a ProcessControl file called „*pc.generic*“.
- Run the *process_TauProfile_files* script to assemble all individual transmittance files into a single NetCDF file.

SRF Example: AVHRR3-NOAA19, Channel 3

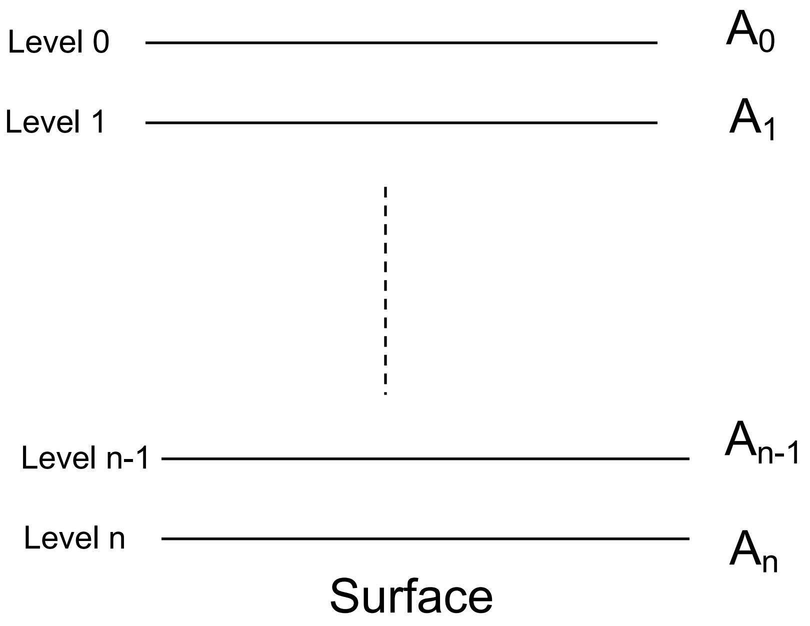



AVHRR3 NOAA19 (Ch. 4) Instrument Transmittance
-0,5
19,5
39,5
59,5
79,5
99,5
n_layers
-0,5
16,1
32,7
49,3
65,9
82,5
n_Profiles
Spectral transmittance [-]
0,6
0,7
0,8
0,8
0,9
1,0
Data Min = 0,6, Max = 1,0

The IR Process, Step by Step

Part 2: Performing the Regression

Gaseous Transmittance Model 1 (AtmAbsorption)
Compact OPTRAN (ODAS)



$$A_i = \int_0^p \frac{r_i}{g\cos(\theta)}\,dp'$$

$$\ln(k_{ch}(A)) = c_0(A) + \sum_{j=1}^{6} c_j(A)P_j(A)$$

$$c_j(A) = \sum_{m=0}^{n} a_{j,m} \ln(A)^m,\ j = 0{,}6,\ n < 10$$

$$\tau_{ch} = \exp(-k_{ch}\delta A)$$ estimate layer transmittance

$$\tau_{ch} := \int \tau_\nu \phi_\nu d\nu$$ Channel transmittance definition

ϕ_ν – spectral response function

K – absorption coefficient of an absorber

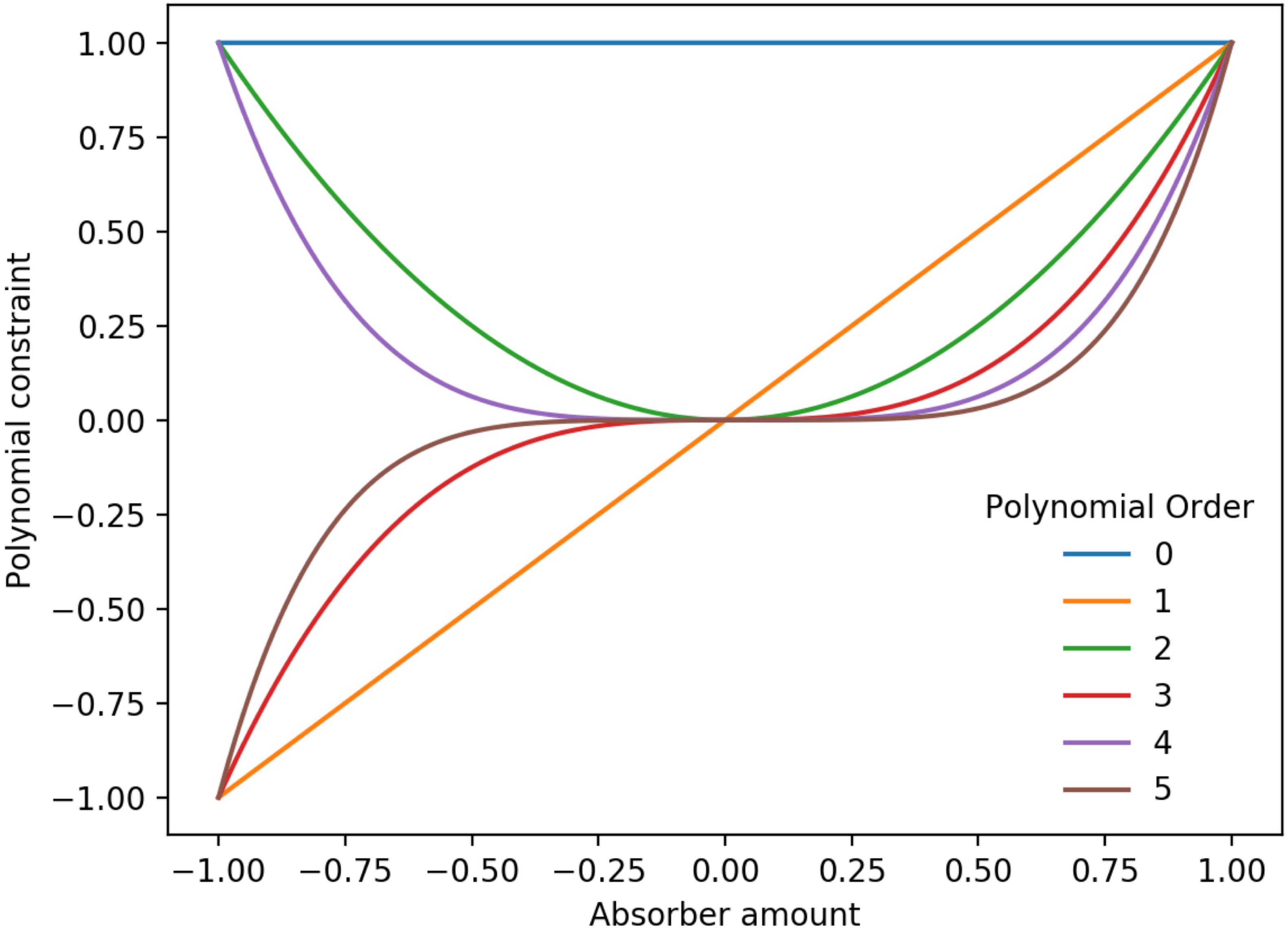
A – integrated absorber amount

P_j – predictors

a_j – constants obtained from regression

- Currently H_2O and O_3 are the only variable trace gases and other trace gases are “fixed”.
- The model provides good Jacobians and is very efficient in using computer memory

ODAS Polynomial Constraints



Gaseous Transmittance Model 2 (AtmAbsorption) ODPS (Optical Depth in Pressure Space)

Regression formulation:

$$d_i - d_{i-1} = \sum_{j=1}^{N_p} c_{i,j} X_{i,j} ,$$

$(d_i - d_{i-1})$ – the layer optical depth

d_i – the level to space optical depth from level *i*

N_p – the number of predictors at layer *i*

$c_{i,j}$ – the regression coefficients

$X_{i,j}$ – the predictors

The regression is actually performed in terms of its departure from a reference profile for all variable gases.

- Variable gases H_2O, CO_2, O_3, and can add absorbers N_2O, CO, and CH_4 for hyper-spectral IR sensors: IASI, AIRS, and CrIS.
- Other features:

(1) Water vapor line computed using ODAS (optional)

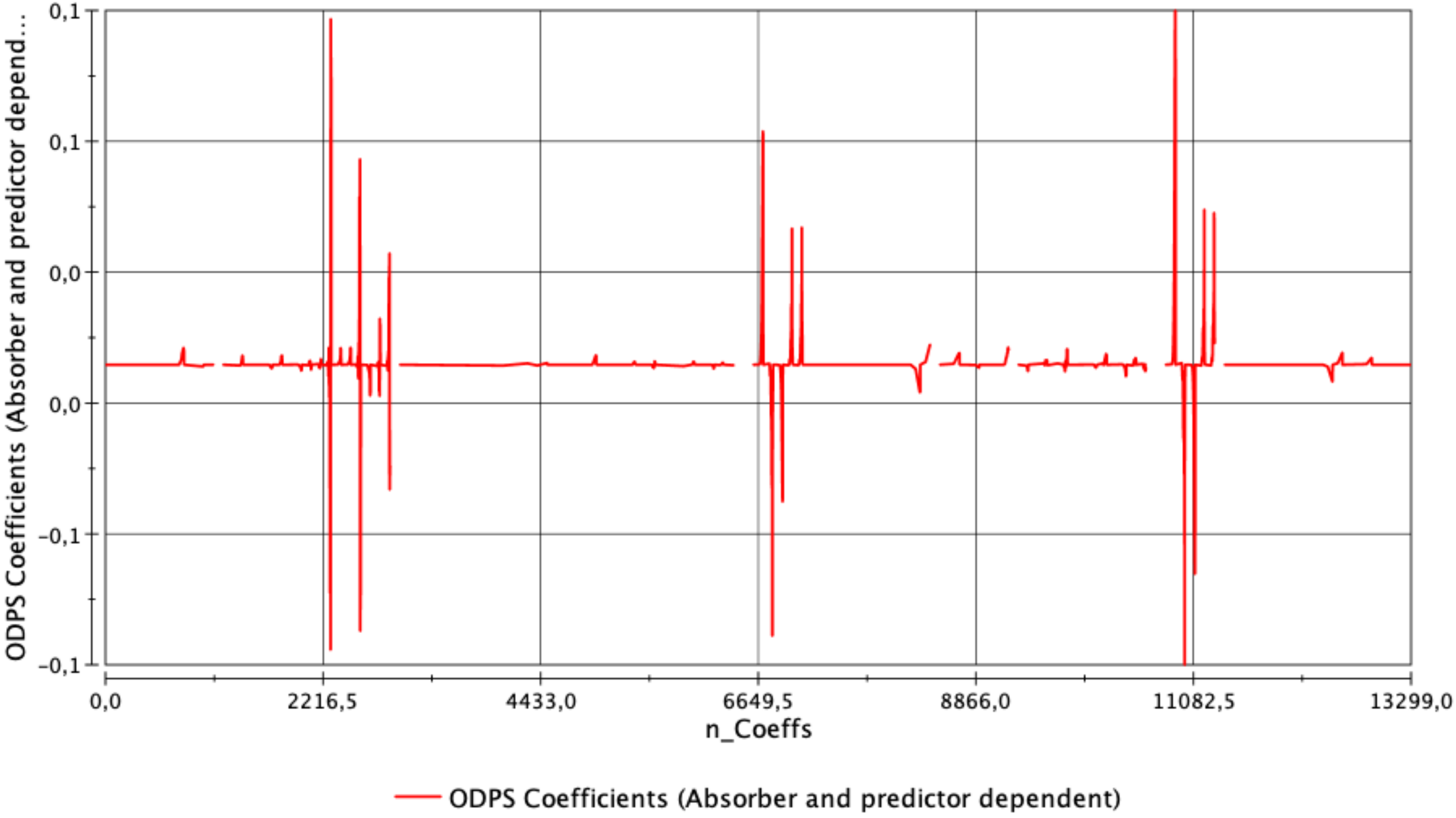
(2) Water vapor continua transmittance is treated separately from the water vapor line absorption.

(3) Have reference profile, and each absorber has associated min and max values.

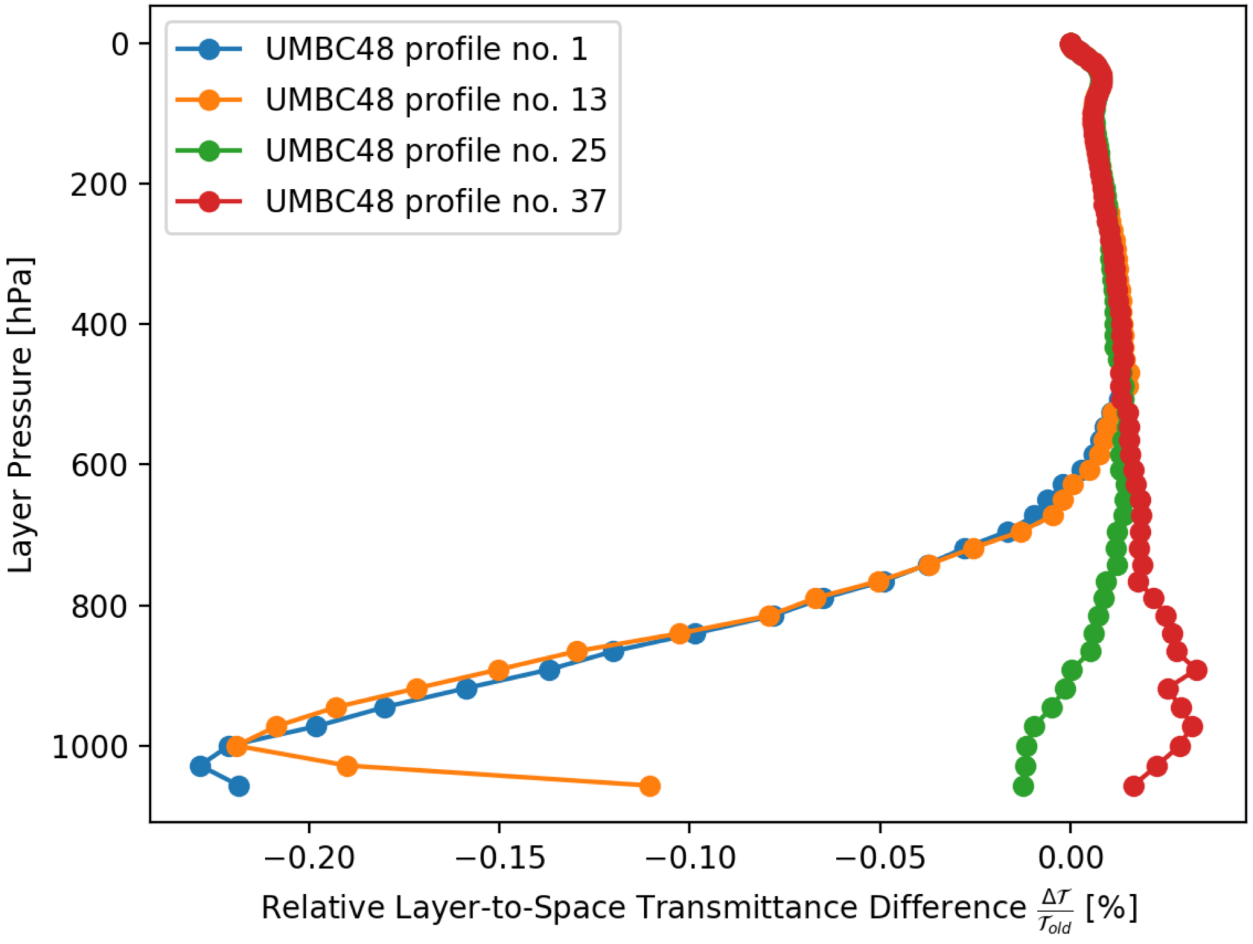
Running the ODPS Regression Code

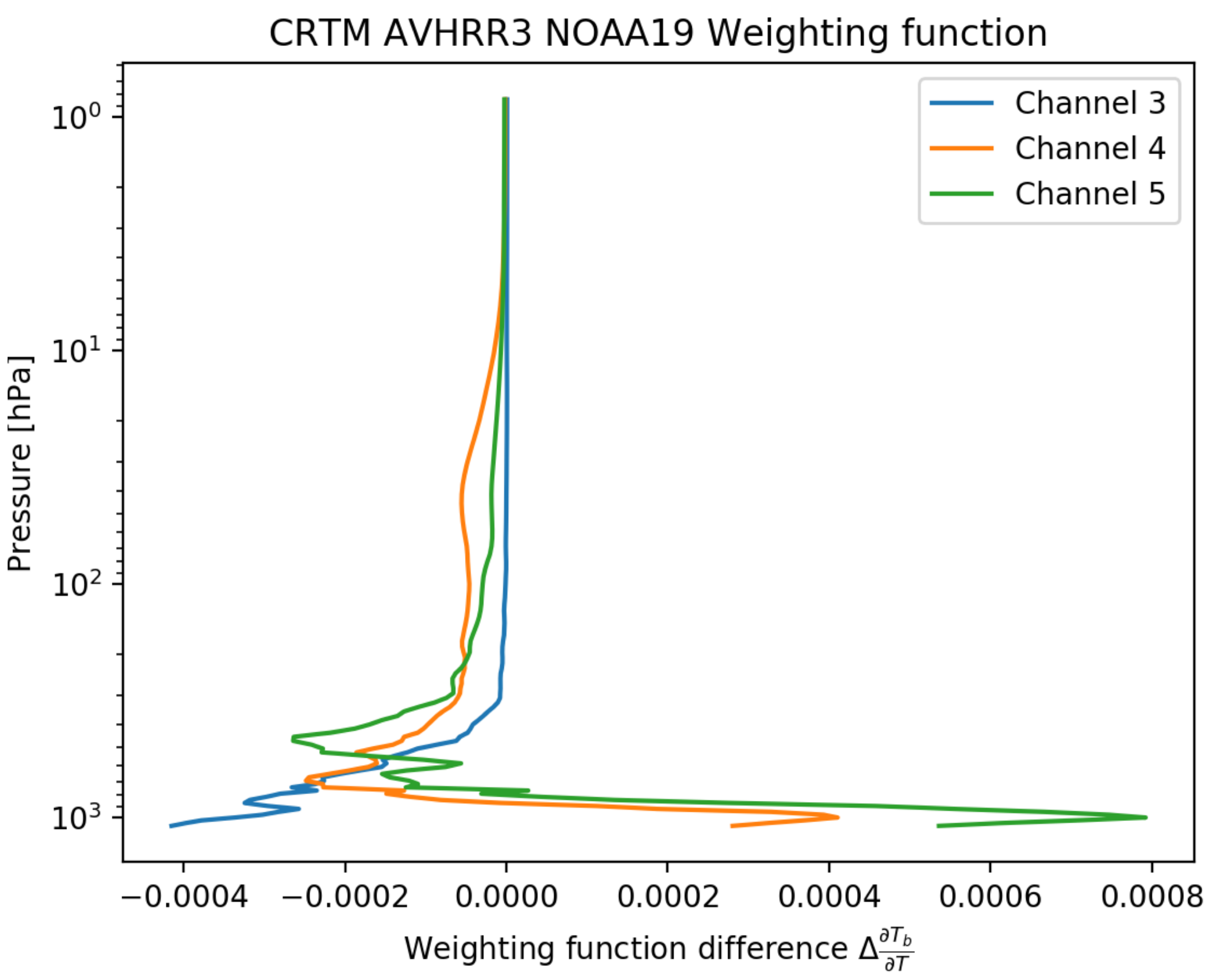
- Enter the *GenCoeff* directory.
- Add the desired sensor ID into the *sensor_list* file.
- Modify the *tau_coeff.parameters* configuration file to set the correct executable and data folders.
- Run all three steps in the *run_tau_coeff.sh* script in succession.
- Convert the NetCDF regression coefficients to the CRTM binary format.

Check ODPS coefficient output



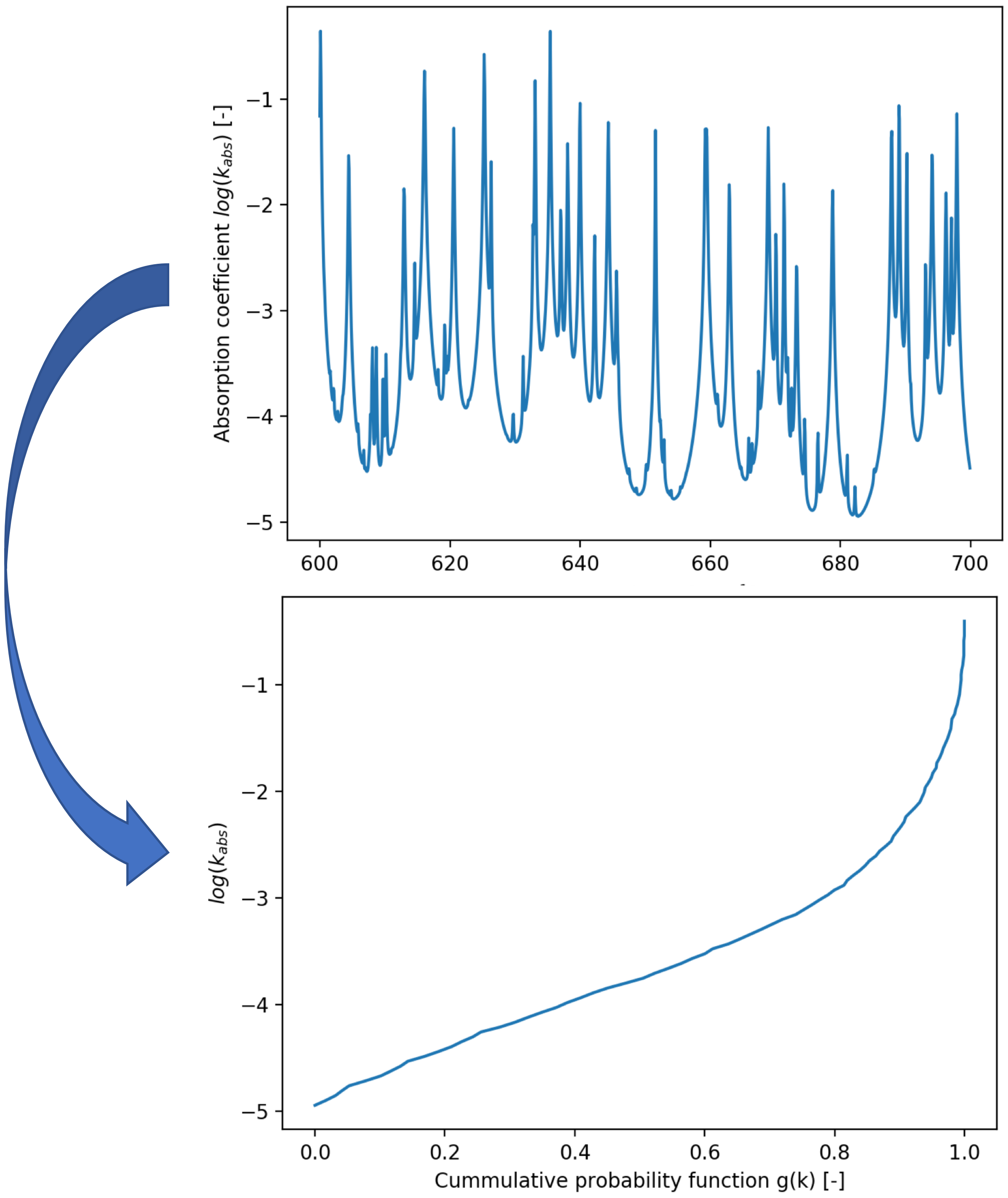
Check CRTM Integration of Coefficients





Alternatives?

- Pros:
  - The process needs to be much more user friendly.
  - More flexibility is necessary for future instruments and science.
  - Better integration with JEDI would be desirable.
- Cons:
  - Existing algorithms are already validated, very accurate and very fast.



Remaining Issues:

- Non-LTE
- Zeeman-Splitting for upper channels of AMSU-A, ATMS, SSMIS
- Apodization step for Interferometers such as IASI

Questions?

Comments?

Backup Slides

Code Dependencies

Dependencies

- HDF5
- netCDF4
- CRTM_dev
- LBLRTMIO

Linking the Libraries

```
NC4_DIR=/opt/netcdf4/4.6.2-intel-18.0.3
HDF_DIR=/opt/hdf5/1.8.21-intel-18.0.3
HDF4_DIR=/opt/hdf4/4.2.14-intel-18.0.3

INCLUDES = -I$(NC4_DIR)/include \
           -I$(HDF_DIR)/include

LIBRARIES = -L$(NC4_DIR)/lib -lnetcdf -lnetcdff\
                      -L$(HDF_DIR)/lib -lhdf5
```

Getting the CRTM

```
$ git clone https://github.com/JCSDA/CRTM_dev

$ cd CRTM_dev

$ git clone https://github.com/JCSDA/CRTM_fix
```

Building the CRTM

```
$ source configuration/gfortran.setup

$ source Set_CRTM_Environment.sh

$ cd scripts/shell

$ ./crtm_install_scripts.sh

$ cd Utility/

$ ./crtm_rebuild.sh
```

Linking the CRTM

```
INCLUDES = -I$HOME/local/CRTM/include

LIBRARIES = -L$HOME/local/CRTM/lib -lCRTM
```