USER GUIDELINE

"RETRIEVAL OF OPTICAL THICKNESS AND PARTICLE EFFECTIVE RADIUS OF LIQUID WATER CLOUDS USING AN OPTIMAL ESTIMATION TECHNIQUE"

Author : Trismono Candra Krisna
Institution : SRON Netherlands Institute for Space Research,
Sorbonnelaan 2, 3584 CA Utrecht, The Netherlands

Version : V2.2
Release date : 28 October 2020

Email : t.c.krisna@sron.nl

CHANGE LOG

Version	Date	Author	Reason for changes
1.0	11 May 2020	Trismono C. Krisna	Initial document
2.0	29 May 2020	Trismono C. Krisna	Implement multi-processing module in SIM_FWD, remove typos, change retrieval equation and gamma parameter in the codes, update manual guideline
2.1	02 June 2020	Trismono C. Krisna	Update perturbation coefficient and cost function threshold, add execution time for SIM_FWD and SIM_RET, update manual guideline
2.2	28 October 2020	Trismono C. Krisna	Update config file, screen interface, and user guideline. Configurable parameters are now placed in the config file in INPUT dir, while previously most of them are written in the codes.

INTRODUCTION

This document serves as a brief guideline to the end-to-end simulator (E2ES) package for retrieving cloud optical and microphysical properties. A modified optimal estimation (OE) approach is applied to derive bulk cloud properties, namely optical thickness *tau* and effective radius *reff*. For simplification, a homogeneous cloud vertical profile is assumed in the simulation.

CONTENT

The software is delivered as a full package consisting of several modules. Each is represented by a folder as follows,

1. 01_CREATE CASE
   To generate the measurement configuration.

2. 02_SIM_FWD
   To simulate line-by-line spectra (forward model) given the output of [1] as the input.

3. 03_L1B
   To generate synthetic measurement given the output of [2] as the input. This module does the convolution to the line-by-line spectra and generate additive noise applied on top. Eventually, it generates measured (noisy) spectra at instrument grids.

4. 04_SIM_RET
   To retrieve bulk cloud properties *tau* and *reff*. The measurement input is taken from the output of [3]. An OE with modified Levenberg-Marquardt (LM) algorithm is applied. Forward model and spectra convolution are also carried out within the module.

5. 05_ERROR_ANALYSIS
   To calculate retrieval error properties based on outputs of [4]. The analysis is carried out by taking Jacobian matrix and error covariance from the last iteration assuming a linear problem.

Additionally, DATA dir is aimed to store auxiliary data such as the spectral albedo (sea water) and the wavelength grid. Those folders must **be placed in parallel** because a relative path is commonly used by each module to determine the location of others. Currently, the modules work consecutively (end-to-end), that makes convenient for sensitivity studies. It is possible to pull out an individual module, such as [4] and [5] if the user wants to work on the retrieval based on real measurements.

HARDWARE AND SOFTWARE REQUIREMENTS

The codes were built and tested in a LINUX environment with 4 cores (CPU) and 8 GB RAM. This is the standard requirement. The number of core is important for running modules with multi process. **Prior running the module, please make sure that *libRadtran* has been installed**. The software is developed alongside *libRadtran* version 2.0.2. Older or newer

versions might be applicable. Please refer to *libRadtran* website http://www.libradtran.org/doc/libRadtran.pdf for more details.

The modules are coded in Python3.7 with standard libraries. Please check the availability of the required libraries in your local machine prior running. A Python3 compiler is necessary. All versions of Python3 compiler should be relevant, but using version 3.7 is recommended. The codes should also work on Python2 with some adjustments. However, this is **highly not recommended** considering its remaining lifetime.

HOW TO RUN MODULES

Brief steps to run each module is explained here. For more detailed information, please check the codes. They are quite self-explanatory. Descriptions have been made to explain each task, so the user can follow the workflow easily. Each module has corresponding MODULES dir, which is essential to store all the codes. To make changes, please make a backup version so it won't risk the original codes.

1. 01_CREATE_CASE
   Please go to MODULES dir. The main code is called 01_main_create_case.py. For this standard example, only *tau* and *reff* are varied. The other parameters, e.g., surface albedo, measurement geometry, cloud height, etc, remain constant but it's possible to be adjusted depending on the purpose. The code can be executed from the terminal,

   >> python3 01_main_create_case.py

   where python3 is the alias of Python3 compiler in your machine. Once the code has been executed, the outputs are stored in OUTPUT dir. Please make sure this folder exists prior execution (recommended), although the code is capable to generate it. If it is already exist and (or) has outputs from previous executions, it will not be overwritten.

2. 02_SIM_FWD
   The main codes are located in MODULES dir, namely 02_main_fwd_sp.py for serial-processing or 02_main_fwd_mp.py for multi-processing. Using the multi-processing module is highly recommended. Before running the main code, the config file must be generated. This can be performed by executing 01_create_config.py. From the terminal, this can be executed as follows,

   >> python3 01_create_fwd_config.py

   This will produce a config file written in INPUT dir. It summarizes the forward simulation settings e.g., *libRadtran* path, aerosol parameters, etc, as well as the number of core deployed for the multi-processing. Here, the number of core is set to 3, which can be extended as long as it does not exceed the number of CPU. Otherwise, the maximum performance won't be gained. Furthermore, do not forget to change the path of *libRadtran* based on the condition of the local machine and please only use an absolute path. Eventually, the main code can be executed from the terminal,

>> python3 02_main_fwd_<sp/mp>.py

Cloud files and *libRadtran* input files will be stored in INPUT dir, while *libRadtran* outputs and formatted radiative quantities are placed in OUTPUT dir. The main code has the capability to generate those folders, but it is recommended to create them before execution.

3. 03_L1B
   The main code is located in MODULES dir, namely 02_main_l1b.py. In this study, we use a simplified noise model based on the Signal-to-Noise Ratio (SNR) of the Moderate Resolution Imaging Spectroradiometer (MODIS, please refer to the link https://modis.gsfc.nasa.gov/about/specifications.php). Before running the main code, the config file must be first generated by 01_create_l1b_config.py. From the terminal, this can be executed as follows,

   >> python3 01_create_l1b_config.py

   This will produce a config file written in INPUT dir. Here, it is assumed that the instrument SNR is half of MODIS. This makes the degree of noise two times higher than MODIS. Please read the description given in the code for more detail. Once the config file has been generated, the main code can be executed from the terminal,

   >> python3 02_main_l1b.py

   The outputs will be stored in OUTPUT dir. The spectra information is split into the visible to near-infrared range (VNIR) and the short wavelength infrared (SWIR) window, following the format commonly used by the instrument. To define the spectra at the instrument grid, a convolution technique is applied. The instrument spectral response function (ISRF) is assumed as a Gaussian function with Full Width at Half Maximum (FWHM) of 3 nm with 201 sampling grid.

4. 04_SIM_RET
   The main codes are placed in MODULES dir, namely 02_main_l2m_sp.py for serial-processing or 02_main_l2m_mp.py for multi-processing. Using the multi-processing module is highly recommended to speed up the computational time. Before running the main code, the retrieval config file must be generated. The code 01_create_l2m_config.py can be used for this purpose. Although a standard config has been defined, users have the freedom to change as long as it is reasonable. Otherwise, deprecated retrieval performance may occur. The code can be executed from the terminal as follows,

   >> python3 01_create_l2m_config.py

   It will produce a config file put in INPUT dir, which summarizes the retrieval configuration. The a priori is set to 8 for *tau* and 10 μm for *reff* according to climatology data. The prior error is intentionally set to a relatively large number, twice of the a priori. This aims to minimize possible constrains in the retrieval. The maximum number of iteration is set to 15, although retrievals will most likely converge before 6 iterations. The noise flag 0 is for retrievals without noise, while 1 represents with noise applied.

The retrieval is performed using wavelengths between 630 and 670 nm in the VNIR window that is sensitive for *tau* and between 1500 and 1660 nm in the SWIR window which is sensitive for *reff*. This corresponds to 79 measurements, which implies an over-constrained problem (number of measurements > state vector). Additionally, it specifies the number of core employed in the retrieval and the *libRadtran* path. Eventually, the main code can be executed from terminal as follows,

>> python3 02_main_l2m_<sp/mp>.py

When running forward simulations, cloud files and *libRadtran* input-output files will be stored in INPUT dir. The retrieval module generates five main outputs, line-by-line spectra, convolved spectra, Jacobian matrix, Gain matrix, and retrieval statistic, which are stored in OUTPUT dir. The statistic summarizes number of iteration, cost function, optimum solution (retrieved state), and retrieval diagnostic. The diagnostic may indicate conv_id 1 = converged, 2 = not converged (exceed maximum number of iteration), and 3 = not converged (lower or upper boundary condition hit). Only retrievals with index 1 provide meaningful information. Jacobian matrix kmat_<pixel_id>_<iter_id>.dat contains information of wavelength, measured radiance, measurement error (variance), modeled radiance, Jacobian *tau*, and Jacobian *reff*. Gain matrix gain_<pixel_id>.dat has information on wavelength, Gain *tau*, and Gain *reff*. Those two matrices are meaningful to further analyze retrieval errors.

5. 05_ERROR_ANALYSIS
   The main code is located in MODULES dir, namely 02_main_error_analysis.py. For doing the linear error analysis, this module takes Jacobian matrix given by [4] as the input. Before running the main code, the configuration must be defined. For this purpose, 01_create_error_config.py can be used. Keep in mind, that the prior error (variance) must be consistent with that assumed in [4]. The code can be executed from the terminal as follows,

   >> python3 01_create_error_config.py

   This will generate the config file written in INPUT dir. Once it is done, the main code can be executed from the terminal as follows,

   >> python3 02_main_error_analysis.py

   Retrieval errors are stored in OUTPUT dir. Four different types of retrieval error are provided, prior_error = error estimate before measurement is made, noise_error = due to instrument noise, smoothing_error = due to the lack of prior information, and total_error = due to combined noise and smoothing error.

KNOWN LIMITATION

The code is dedicated for liquid water clouds. For cirrus, adjustments are required in the cloud file and *libRadtran* input e.g., cloud type, ice properties, etc. The computation of Jacobian matrix is considerably expensive. To calculate Jacobian *tau* and *reff*, the forward model need to be run three times. Moreover, the forward model uses an option of “tau550 set”. This makes the entire spectra including at 550 nm must calculated. Those factors makes one iteration needs

approximately 2.5 minutes, depending on the machine condition. Assuming one retrieval needs 5 iterations, ca. 12.5 minutes is required for one pixel retrieval. To compensate this high computational time, the use of multi-processing method is highly recommended. In the future, alternative methods to derive Jacobian matrix should be implemented.

SOURCE CODE

Codes are available on the version control https://github.com/trismono/CLOUD-RETRIEVAL.git. Should any questions arise regarding the modules, please do not hesitate to contact the author.

REFERENCE

Rodgers, C. D., 2000: Inverse Methods for Atmospheric Sounding: Theory and Practice. World Scientific Publishing Company, 240 pp.

APPENDIX

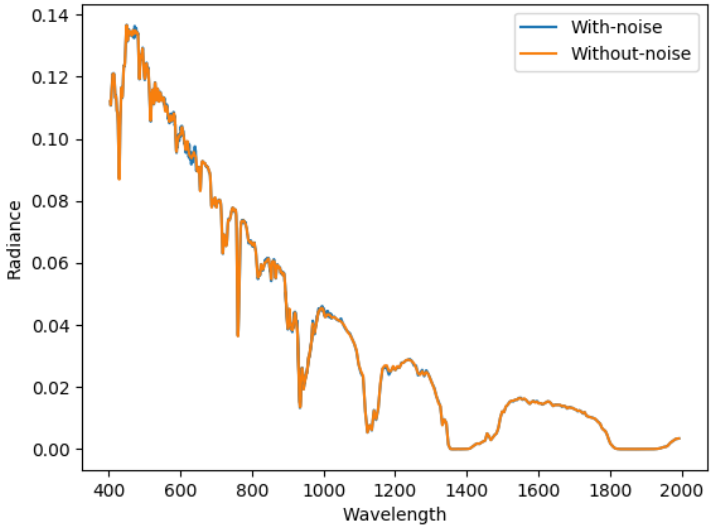

Figure 1. Spectral radiance with and without noise for cloudy condition with *tau* = 4 and *reff* = 8 µm, surface albedo = sea water, sza = 30°, and phi0 = 100°.

```
trismonok — ssh -YC trismonock@ssh.sron.nl — 120×46
[newton [MODULES]% ./02_main_l2m_sp.py
Info          | ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Info          | End-to-End Simulator Level-2 Retrieval Module
Info          | Author       : Trismono C. Krisna
Info          | Institution  : SRON Netherlands Institute for Space Research
Info          | Version      : v2.1
Info          | Date         : 27 October 2020
Info          | ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Info          |
Info          | Noise realisation : with noise
Info          | Initial guess of tau : 8.0
Info          | Initial guess of reff : 10.0 micron
Info          | Variance of tau : 16.0
Info          | Variance of reff : 20.0 micron
Info          | Number of maximum iteration : 15
Info          | Radiance noise : 3.0 %
Info          | LibRadtran dir : /deos/trismonock/libRadtran-2.0.2/
Info          |
Info          | Pixel ID 2 :: Iteration 1
Info          | Pixel ID 2 :: Perturbation coefficient p_tau = -0.32 and p_reff = -0.40 micron
Info          | Pixel ID 2 :: State tau = 3.718 and reff = 10.415 micron
Info          | Pixel ID 2 :: Cost function chi2 = 38305.492
Info          | Pixel ID 2 :: Iteration 2
Info          | Pixel ID 2 :: Perturbation coefficient p_tau = -0.15 and p_reff = 0.42 micron
Info          | Pixel ID 2 :: State tau = 4.061 and reff = 8.345 micron
Info          | Pixel ID 2 :: Cost function chi2 = 898.261
Info          | Pixel ID 2 :: Iteration 3
Info          | Pixel ID 2 :: Perturbation coefficient p_tau = 0.08 and p_reff = -0.17 micron
Info          | Pixel ID 2 :: State tau = 4.005 and reff = 8.379 micron
Info          | Pixel ID 2 :: Cost function chi2 = 25.870
Info          | Pixel ID 2 :: Iteration 4
Info          | Pixel ID 2 :: Perturbation coefficient p_tau = 0.04 and p_reff = -0.08 micron
Info          | Pixel ID 2 :: State tau = 3.992 and reff = 8.276 micron
Info          | Pixel ID 2 :: Cost function chi2 = 12.900
Info          | Pixel ID 2 :: Iteration 5
Info          | Pixel ID 2 :: Perturbation coefficient p_tau = -0.04 and p_reff = -0.08 micron
Info          | Pixel ID 2 :: State tau = 3.992 and reff = 8.274 micron
Info          | Pixel ID 2 :: Cost function chi2 = 12.682
Info          | Pixel ID 2 :: Convergence ID = 1 Retrieval succeed!
Info          | Pixel ID 2 :: Retrieval done in 705.43 sec
Info          | Pixel ID 2 :: Elapsed time per iteration = 141.09 sec
Info          |
Info          | ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Info          | End of Level-2 Retrieval Module
Info          | ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
newton [MODULES]%
```

Figure 2. Screen output of retrieval module. Converged after 5th iteration with *tau* = 3.98, *reff* = 8.21 µm, chi2 = 13.46 for n_measurement = 79.

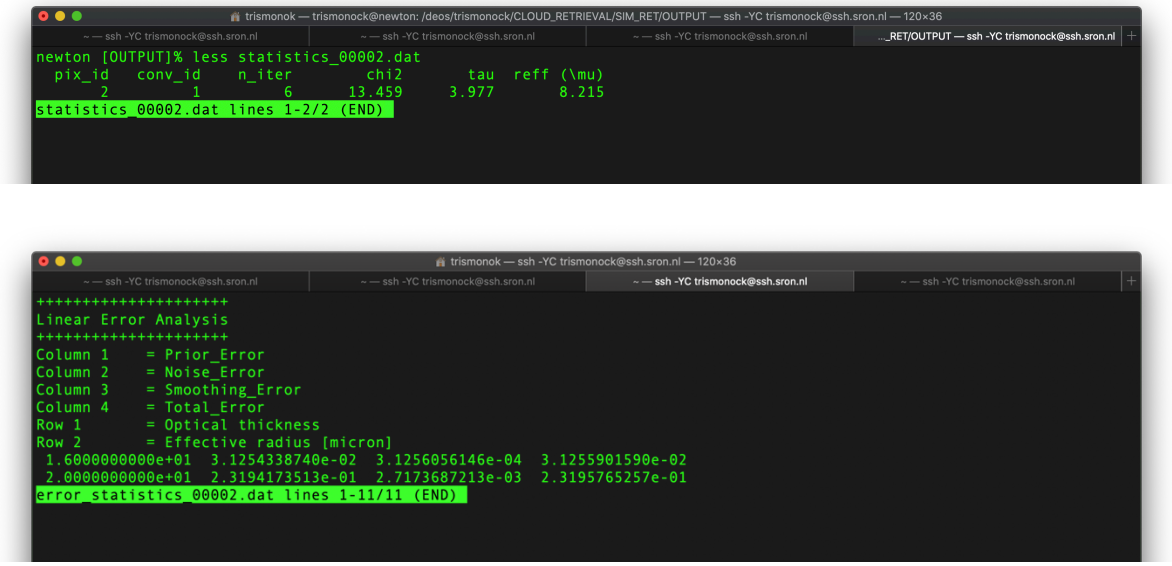

Figure 3. Retrieval statistic (top) and error (bottom). The order of retrieval error (from top): *tau* and *reff*.

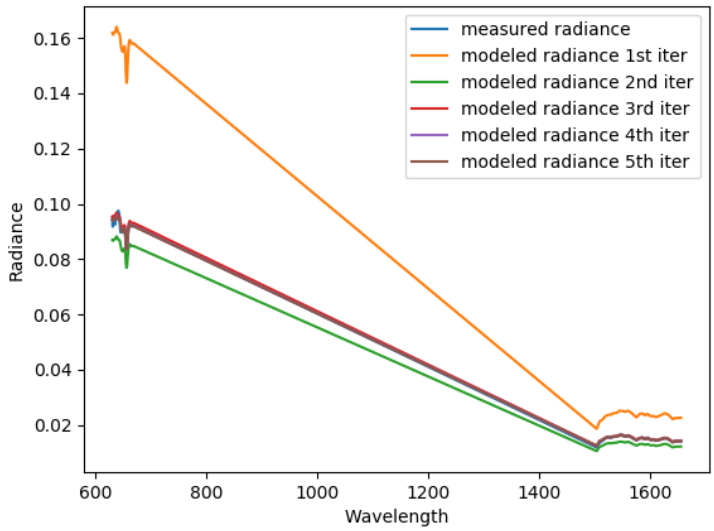

Figure 4. Comparison of measured and modeled radiance for each iteration. An optimum solution is reached at 5th iteration.

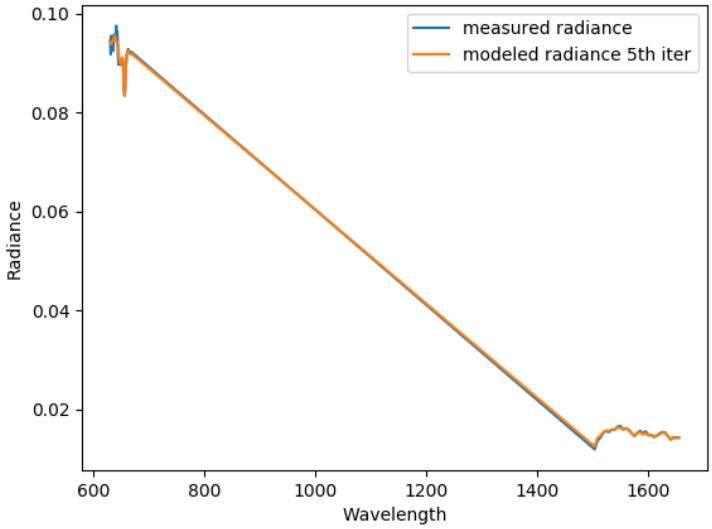

Figure 5. Comparison of measured and modeled (optimum solution – 5th iteration) radiance.