MANUAL GUIDELINE
CLOUD RETRIEVAL – MODIFIED OPTIMAL ESTIMATION

Author : Trismono Candra Krisna
Version : V2.0
Release date : 29 May 2020
Email : t.c.krisna@sron.nl

INTRODUCTION

This document is written as a brief guideline to the end-to-end (E2E) cloud retrieval software. 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. The software is written in Python 3.7 with its standard libraries. For the forward model, the library of radiative transfer routines and programs (*libRadtran*) version 2.0.2 is used. Please refer to *libRadtran* website http://www.libradtran.org/doc/libRadtran.pdf for more details.

CONTENT

Not only the retrieval module, the software is delivered as a full package consisting of several modules and each is represented by a folder as follows,

1. CREATE CASE
   To generate the configuration of measurement condition.

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

3. L1B
   To generate synthetic measurement given the output of [2] as the input. This module generates spectra at instrument wavelengths by a convolution with noise applied on top of it.

4. SIM_RET
   To retrieve bulk cloud properties *tau* and *reff*. The measurement input is taken from the output of [3]. A modified OE including Levenberg-Marquardt (LM) approach is implemented in the retrieval. Forward model and spectra convolution module are also carried out within the retrieval.

5. ERROR_ANALYSIS
   To calculate error using outputs of [4] as the input. The analysis is carried out by taking Jacobian matrix and error covariance from the last iteration (linear approach).

Additionally, DATA dir is aimed to store auxiliary data such as the spectral albedo (sea water) and the wavelength grid. Note, those folders must **be place in parallel** as a relative path is commonly used by each module to locate other directories. Currently, the modules work consecutively (end-to-end concept), that makes convenient for doing a sensitivity study.

However, it is possible to pull out an individual module, such as [4] and [5] if the user wants to work on the retrieval of real measurement data and so on.

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 an important matter for running retrievals with multiprocessing module. Before running the module, *libRadtran* should have been installed. The software is built and tested using *libRadtran* version 2.0.2. An older or newer version might be also applicable. As mentioned, the codes are written in Python3.7. Therefore, Python 3 compiler is necessary. All python 3 compilers should be relevant. The codes should also work on Python 2 with some adjustments. However, the use of Python 2 is **not recommended** considering its remaining lifetime.

HOW TO RUN MODULES

Brief steps to run each module is explained here. For more detailed information, please see tin he codes. Descriptions have been made to explain each task, so that the user can follow the workflow easily. Each module has corresponding MODULES dir, which is essential as it stores the routines. If users want to make changes, please make a copy dir so it won't risk the master.

1. CREATE_CASE
   Please go to subdir MODULES. The main code is called main_create.py. For this standard example, only *tau* and *reff* are varied. The other parameters remain constant but it's possible to be adjusted depending on the purpose. Normally, the code can be executed from the terminal,

   ```
   >> python3 main_create.py
   ```

   Make sure that **python3** is the alias of Python 3 compiler in your machine. Once the code has been executed, the outputs are stored in the subdir OUTPUT. Please make sure this folder exists (recommended), although the code is capable to create the dir. If the dir exists and (or) has outputs from previous executions, it will not be overwritten.

2. SIM_FWD
   The main code is located in the MODULES dir, namely main_fwd_sp.py (for serial processing) or main_fwd_mp.py (for multi-processing). **Important:** Change the path of *libRadtran* in fwd_wrapper.py based on the condition in your local machine and use an absolute path. Eventually, the code can be executed from the terminal,

   ```
   >> python3 main_fwd_<sp/mp>.py
   ```

   The use of multi-processing module is recommended. The number of process employed, namely n_proc, must be adjusted in main_fwd_mp.py. The standard code uses n_proc = 3. Users are free to extend this number as long as it does not exceed the number of core (CPU). Otherwise, the maximum performance won't be gained

   After execution, the outputs will be stored in the subdir INOUT. The subdir CLOUD is dedicated to store the cloud profiles. The main code may generate those dirs, but it is recommended to create them before execution.

3. L1B
   The main code is in the MODULES dir, namely main_l1b.py. The noise realization is adapted from the Moderate Resolution Imaging Spectroradiometer (MODIS) Signal-to-Noise Ratio (SNR). This is a simple assumption, thus it is possible to be modified by the user. Please read the description given in the code for more detail. The code can be executed from the terminal,

   >> python3 main_l1b.py

   The outputs are stored in the subdir OUTPUT. For each measurement, the spectra information is split into the visible to near-infrared range (VNIR) and the short wavelength infrared (SWIR). To define the spectra at a particular wavelength, a convolution is applied. The instrument spectral response function (ISRF) is assumed as a Gaussian function with FWHM of 3 nm and 201 sampling grid.

4. SIM_RET
   The main code is placed in the MODULES dir and it is called main_l2m_sp.py for serial-processing or main_l2m_sp.py for multi-processing. Using multi-processing module is recommended. The number of process employed, namely n_proc, must be adjusted. The standard code uses $n\_proc = 3$. Users are free to extend this number as long as it does not exceed the number of core (CPU). Otherwise, the maximum performance won't be gained.

   Important: To locate the *libRadtran* path to the right one, open l2m_wrapper.py in MODULES and adjust variable lib_dir. Please use an absolute path. Before running retrievals, please take a look in the subdir INPUT (do not change or remove this dir). A python code retrieval_setting.py is available to create the retrieval setting. Albeit, a standard setting has been defined, users have the freedom to change the setting. However, it must be reasonable.

   The a priori state vector is set to 8 for *tau* and 10 μm for *reff*, following climatology data. The weighting or prior error is set to a relatively large number to minimize possible constrains in the retrieval. The maximum number of iteration is currently set to 15. However, retrievals will most likely converge before 10 iterations (4-8 in average). The noise flag = 1 is for retrieval with noise and 0 for without noise applied on the measurement. Currently, the retrieval is performed using wavelengths between 630 and 670 nm in the visible range (sensitive for *tau*), and between 1500 and 1660 nm in the near-infrared range (sensitive for *reff*). By this definition, the number of measurements used in the retrieval would be 79 (over-constrained problem). Finally, the retrieval code can be executed from terminal as follows,

   >> python3 main_l2m_<sp/mp>.py

   Outputs are stored in the subdir OUTPUT, while the subdir DUMMY is for holding input-output of forward simulation, such as cloud profile, line-by-line, and convolved spectra. Those two dirs can be generated automatically by the code, but it is recommended to create them before execution.

The retrieval module generates three main outputs: the Jacobian matrix, the Gain matrix, the statistic file which summarize retrieval output and retrieval diagnostic. conv_id 1 = converged, 2 = not converged (exceed maximum number of iteration) , and 3 = not converged (boundary condition hit). Only retrievals with index 1 provide meaningful information. The Jacobian kmat_<pixel_id>_<iter_id>.dat is comprised of six columns: wavelength, measured radiance, measurement covariance, modeled radiance (forward model), Jacobian *tau*, and Jacobian *reff*. The Gain gain_<pixel_id>.dat has three columns: wavelength, Gain *tau*, and Gain *reff*. Those two matrices are useful to investigate retrieval errors.

5. ERROR_ANALYSIS
   The main code is located in the MODULES dir and it is called main_l2m_error.py. Variable weighting (prior error) is defined in the code and this must be consistent with those defined in the retrieval setting [4]. Jacobian matrix from the last iteration [4] is used in this analysis. The code can be executed from the terminal as follows,

   >> python3 main_l1m_error.py

   Error properties are stored in the subdir OUTPUT. Four different errors are provided: prior_error = error estimate before measurement is made, noise_error = error due to measurement noise, smoothing_error = error due to lack of prior information, and total_error = error due to noise and smoothing error.

KNOWN LIMITATION

The computation of Jacobian matrix is considerably expensive. To calculate Jacobian *tau* and *reff*, the forward model need to be run three times. Since the forward model opts for tau550 set, the entire spectra must be calculated. Those factors makes one iteration needs 3 minutes more or less, depending on the machine. Assuming one retrieval needs maximum 8 iteration, around 24 minutes are required in total for one pixel retrieval. Therefore, a multiprocessing module is applied to tackle this issue. In the future, alternative methods to derive Jacobian matrix should be implemented to reduce the high computational time.

MESSAGE FROM AUTHOR

Feel free to use, modify, and shares the codes. If you found issues or questions, please reach me at t.c.krisna@sron.nl

REFERENCE

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

APPENDIX

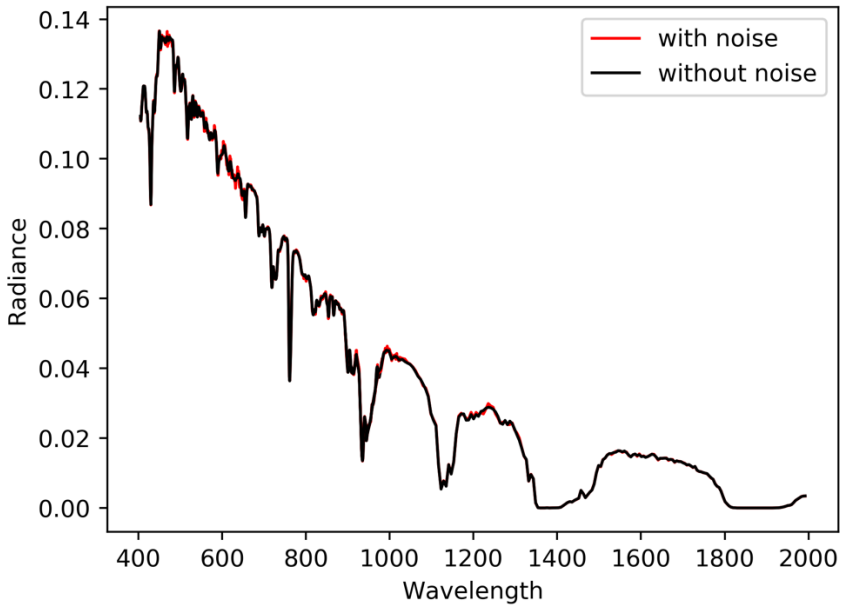

Figure 1. Spectra radiance with and without noise for cloudy condition. tau = 4 and reff = 8 μm, surface albedo = sea water, sza = 30° and phi0 = 100°.

```
Info     | Pixel ID 1 :: Iteration 3
Info     | Perturbation coefficient p_tau = 0.21 and p_reff = 0.42
Info     | tau = 4.720 and reff = 8.137
Info     | cost function chi2 = 2245.825
Info     | Pixel ID 1 :: Iteration 4
Info     | Perturbation coefficient p_tau = 0.13 and p_reff = 0.34
Info     | tau = 3.220 and reff = 8.450
Info     | cost function chi2 = 2033.958
Info     | Pixel ID 1 :: Iteration 5
Info     | Perturbation coefficient p_tau = 0.19 and p_reff = -0.33
Info     | tau = 3.975 and reff = 8.234
Info     | cost function chi2 = 2238.492
Info     | Pixel ID 1 :: Iteration 6
Info     | Perturbation coefficient p_tau = 0.19 and p_reff = -0.51
Info     | tau = 3.966 and reff = 8.177
Info     | cost function chi2 = 13.384
Info     | Pixel ID 1 :: Iteration 7
Info     | Perturbation coefficient p_tau = 0.16 and p_reff = 0.33
Info     | tau = 3.974 and reff = 8.210
Info     | cost function chi2 = 13.378
Info     | Convergence ID = 1 :: Retrieval succeed!
```

Figure 2. Screen output of retrieval module. Converged after 7 iterations with tau = 3.9, reff = 8.2 μm, chi2 = 13.38 for n_measurement = 79.

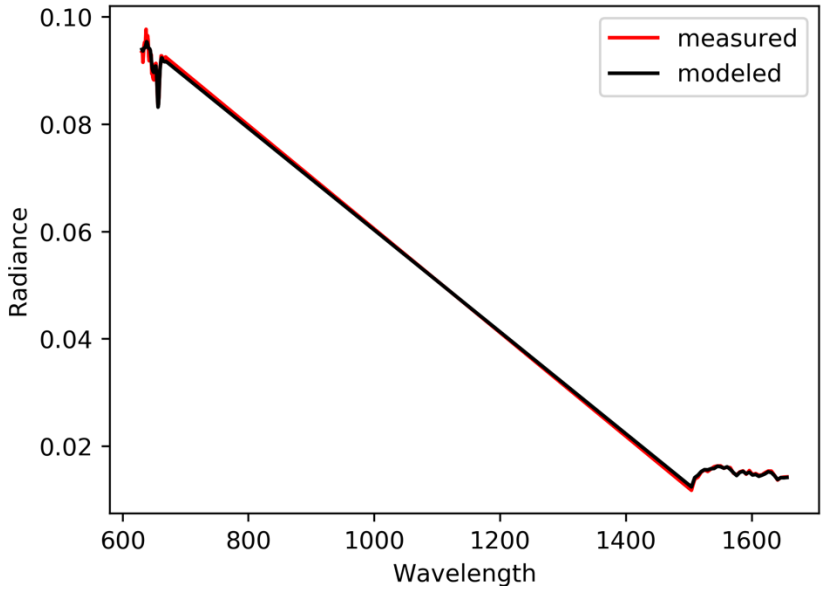

Figure 3. Comparison of spectral radiance between measured and retrieved. The retrieval is performed using wavelengths from 630-670 nm and 1500-1660 nm.