Bistatic first order scattering model Documentation

Release 0.1

Raphael Quast, Alexander Loew

Sep 20, 2023

CONTENTS

Contents:

CHAPTER

ONE

INTRODUCTION

The *RT1* package implements a bistatic first order scattering radiative transfer model that might be used for generic purposes. Unique features of this model are

- analytical solution of the radiative transfer equation up to first order with flexible BRDF and volume phase function specifications.
- extensive nonlinear least-squares fit module for parameter estimation
- thoroughly tested model code

1.1 Credits and References

RT1 is provided as open-source software, hoping that it will help you in your research.Please read the LICENSE agreement related to this software which gives you much flexibility to (re)use the code. Currently we use the APACHE-2.0 license.

The developers would very much appreciate to receive feedback how the model is used. Also contributions and suggestions for further improvements are highly welcome.

When you are using *RT1* as part of your publications, please give the developers credit by giving reference to the GitHub repository and to the following papers:

- R.Quast and W.Wagner, *Analytical solution for first-order scattering in bistatic radiative transfer interaction problems of layered media*, Appl.Opt.55, 5379-5386 (2016)
- R.Quast, C.Albergel, J.C.Calvet, W.Wagner, *A Generic First-Order Radiative Transfer Modelling Approach for the Inversion of Soil and Vegetation Parameters from Scatterometer Observations*, doi:10.3390/rs11030285

CHAPTER

TWO

INSTALLATION

There are currently different methods to install *rt1*:

2.1 Using conda

TBD conda installation is currently under development . . . be patient!

The installation using conda is as easy as:

```
conda install [-n YOURENV] -c conda-forge rt1
```

2.2 Using pip

The *rt1* package is provided on pip. Install is as easy as:

```
pip install rt1
```

2.3 The standard python way

You can also download the source code package from the project website or from pip. Unpack the file you obtained into some directory (it can be a temporary directory) and then run:

```
python setup.py install
```

If might be that you might need administrator rights for this step, as the program tries to install into system library pathes. To install into a user specific directory please refer to the *pip* documentation.

CHAPTER

THREE

THEORY

The RT1 module provides a method for calculating the scattered radiation from a uniformly illuminated rough surface covered by a homogeneous layer of tenuous media. The following sections are intended to give a general overview of the underlying theory of the RT1 module. A more general discussion on the derivation of the used results can be found in [QuWa16]. Details on how to define the scattering properties of the covering layer and the ground surface within the RT1-module are given in *Model Specification*.

The utilized theoretical framework is based on applying the Radiative Transfer Equation (RTE) (3.1) to the geometry shown in Fig. 3.1.

$$\cos(\theta)\frac{\partial I_f(r,\theta,\phi)}{\partial r} = -\kappa_{ex} I_f(r,\theta,\phi) + \kappa_s \int_0^{2\pi} \int_0^{\pi/2} I_f(r,\theta',\phi')\hat{p}(\theta,\phi,\theta',\phi')\sin(\theta')d\theta' d\phi' \tag{3.1}$$

The individual variables are hereby defined as follows:

- θ denotes the polar angle in a spherical coordinate system
- ϕ denotes the azimuth angle in a spherical coordinate system
- r denotes the distance within the covering layer
- $I_f(r,\theta,\phi)$ denotes the specific intensity at a distance r within the covering layer propagating in direction (θ,ϕ).
- κ_{ex} denotes the extinction-coefficient (i.e. extinction cross section per unit volume)
- $\hat{p}(\theta,\phi,\theta',\phi')$ denotes the scattering phase-function of the constituents of the covering layer

Note: To remain consistent with [QuWa16], the arguments of the functions $\hat{p}(\theta,\phi,\theta',\phi')$ and $BRDF(\theta,\phi,\theta',\phi')$ are defined as angles with respect to a spherical coordinate system in the following discussion. Within the RT1-module however, the functions are defined with respect to the associated zenith-angles!

A relationship between the module-functions and the functions within the subsequent discussion is therefore given by:

`SRF.brdf(t_0,t_ex,p_0,p_ex)` $\hat{=} BRDF(\pi-\theta_0,\phi_0,\theta_{ex},\phi_{ex})$ and `V.p(t_0,t_ex,p_0,p_ex)` $\hat{=} \hat{p}(\pi-\theta_0,\phi_0,\theta_{ex},\phi_{ex})$

3.1 Problem Geometry and Boundary Conditions

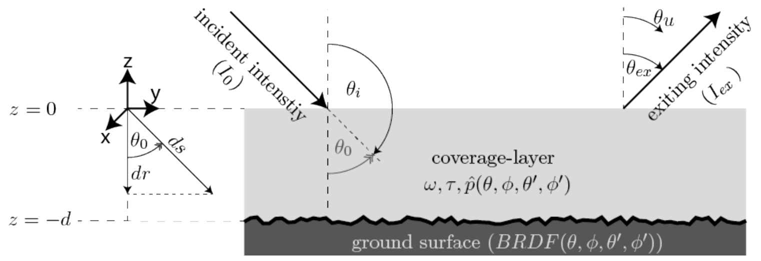

Fig. 3.1: Illustration of the chosen geometry within the RT1-module (adapted from [QuWa16])

As shown in Fig. 3.1, the considered problem geometry is defined as a rough surface covered by a homogeneous layer of a scattering and absorbing medium.

In order to be able to solve the RTE (3.1), the boundary-conditions are specified as follows:

- The top of the covering layer is uniformly illuminated at a single incidence-direction:

$$I_0(z = 0, \theta, \phi) = \frac{I_0}{\sin(\theta)} \delta(\theta - \theta_i) \delta(\phi - \phi_i) \tag{3.2}$$

- Radiation impinging at the ground surface is reflected upwards according to its associated Bidirectional Reflectance Distribution Function (BRDF)

$$I^+(z = -d, \theta, \phi) = \int_0^{2\pi} \int_0^{\pi} I^-(z = -d, \theta, \phi) BRDF(\theta, \phi, \theta', \phi') \sin(\theta') d\theta' d\phi' \tag{3.3}$$

The superscripts $I^{\pm}$ hereby indicate a separation between upwelling $(+)$ and downwelling $(-)$ intensity.

The additional specifications of the covering layer and the ground surface are summarized as follows:

3.1.1 Parameters used to describe the scattering properties of the covering layer

Scattering Phase Function: (i.e. *normalized differential scattering cross section*)

$$\hat{p}(\theta, \phi, \theta', \phi') \qquad \text{with} \qquad \int_0^{2\pi} \int_0^{\pi} \hat{p}(\theta, \phi, \theta', \phi') \sin(\theta') d\theta' d\phi' = 1 \tag{3.4}$$

Optical Depth:

$$\tau = \kappa_{ex}\, d = (\kappa_s + \kappa_a)\, d \tag{3.5}$$

where κ_{ex} is the *extinction coefficient* (i.e. extinction cross section per unit volume) , κ_s is the *scattering coefficient* (i.e. scattering cross section per unit volume) , κ_a is the *absorption coefficient* (i.e. absorption cross section per unit volume) and d is the *total height of the covering layer*.

Single Scattering Albedo:

$$\omega = \frac{\kappa_s}{\kappa_{ex}} = \frac{\kappa_s}{\kappa_s + \kappa_a} \leq 1 \tag{3.6}$$

3.1.2 Parameters used to describe the scattering properties of the ground surface

Bidirectional Reflectance Distribution Function:

$$BRDF(\theta,\phi,\theta',\phi') \qquad \text{with} \qquad \int_0^{2\pi}\int_0^{\pi/2} BRDF(\theta,\phi,\theta',\phi')\cos(\theta')\sin(\theta')d\theta' d\phi' = R(\theta,\phi) \leq 1 \tag{3.7}$$

where $R(\theta,\phi)$ denotes the **Directional-Hemispherical Reflectance** of the ground surface.

TBD: perhaps describe also normalization conditions for p and BRDF

3.2 First-order solution to the RTE

Incorporating the above specifications, a solution to the RTE is obtained by assuming that the scattering coefficient κ_s of the covering layer is small (i.e. $\kappa_s \ll 1$). Using this assumption, the RTE is expanded into a series with respect to powers of κ_s, given by:

$$I^+ = I_{\text{surface}} + I_{\text{volume}} + I_{\text{interaction}} + (I_{svs}) + \mathcal{O}(\kappa_s^2) \tag{3.8}$$

where the individual terms (representing the contributions to the scattered intensity at the top of the covering layer) can be interpreted as follows:

- I_{surface}: radiation scattered once by the ground surface
- I_{volume}: radiation scattered once within the covering layer
- $I_{\text{interaction}}$: radiation scattered once by the ground surface and once within the covering layer
- I_{svs}**: radiation scattered twice by the ground surface and once within the covering layer**
  (This contribution is assumed to be negligible due to the occurrence of second order surface-scattering)

After some algebraic manipulations the individual contributions are found to be given by (details can be found in [QuWa16]):

$$I_{\text{surface}}(\theta_0,\phi_0,\theta_{ex},\phi_{ex}) = I_0 e^{-\frac{\tau}{\cos(\theta_0)}}\ e^{-\frac{\tau}{\cos(\theta_{ex})}}\cos(\theta_0) BRDF(\pi-\theta_0,\phi_0,\theta_{ex},\phi_{ex}) \tag{3.9}$$

$$I_{\text{volume}}(\theta_0,\phi_0,\theta_{ex},\phi_{ex}) = I_0\ \omega\ \frac{\cos(\theta_0)}{\cos(\theta_0)+\cos(\theta_{ex})}\left(1-e^{-\frac{\tau}{\cos(\theta_0)}}\ e^{-\frac{\tau}{\cos(\theta_{ex})}}\right)\hat{p}(\pi-\theta_0,\phi_0,\theta_{ex},\phi_{ex}) \tag{3.10}$$

$$I_{\text{interaction}}(\theta_0,\phi_0,\theta_{ex},\phi_{ex}) = I_0\ \cos(\theta_0)\ \omega\left\{e^{-\frac{\tau}{\cos(\theta_{ex})}} F_{int}(\theta_0,\theta_{ex}) + e^{-\frac{\tau}{\cos(\theta_{ex})}} F_{int}(\theta_{ex},\theta_0)\right\} \tag{3.11}$$

$$\text{with} \qquad F_{int}(\theta_0,\phi_0,\theta_{ex},\phi_{ex}) = \int_0^{2\pi}\int_0^{\pi}\frac{\cos(\theta)}{\cos(\theta_0)-\cos(\theta)}\left(e^{-\frac{\tau}{\cos(\theta_0)}} - e^{-\frac{\tau}{\cos(\theta)}}\right)\ \hat{p}(\theta_0,\phi_0,\theta,\phi) BRDF(\pi-\theta,\phi,\theta_{ex},\phi_{ex})\sin(\theta \tag{3.12}$$

3.3 Evaluation of the interaction-contribution

In order to analytically evaluate the remaining integral appearing in the interaction-term, the BRDF and the scattering phase-function of the covering layer are approximated via a Legendre-series in a (possibly generalized) scattering angle of the form:

$$BRDF(\theta,\phi,\theta_s,\phi_s) = \sum_{n=0}^{N_b} b_n P_n(\cos(\Theta_{a_b})) \tag{3.13}$$

$$\hat{p}(\theta,\phi,\theta_s,\phi_s) = \sum_{n=0}^{N_p} p_n P_n(\cos(\Theta_{a_p})) \tag{3.14}$$

where $P_n(x)$ denotes the n$^{\text{th}}$ Legendre-polynomial and the generalized scattering angle Θ_a is defined via:

$$\cos(\Theta_a) = a_0\cos(\theta)\cos(\theta_s) + \sin(\theta)\sin(\theta_s)\left[a_1\cos(\phi)\cos(\phi_s) + a_2\sin(\phi)\sin(\phi_s)\right] \tag{3.15}$$

where θ, ϕ are the polar- and azimuth-angles of the incident radiation, θ_s, ϕ_s are the polar- and azimuth-angles of the scattered radiation and a_1, a_2 and a_3 are constants that allow consideration of off-specular and anisotropic effects within the approximations.

Once the b_n and p_n coefficients are known, the method developed in [QuWa16] is used to analytically solve $I_{\text{interaction}}$.

This is done in two steps:

First, the so-called fn-coefficients are evaluated which are defined via:

$$\int_0^{2\pi} \hat{p}(\theta_0,\phi_0,\theta,\phi)BRDF(\pi-\theta,\phi,\theta_{ex},\phi_{ex})d\phi = \sum_{n=0}^{N_b+N_p} f_n(\theta_0,\phi_0,\theta_{ex},\phi_{ex})\cos(\theta)^n \tag{3.16}$$

Second, $I_{\text{interaction}}$ is evaluated using the analytic solution to the remaining θ-integral for a given set of fn-coefficients as presented in [QuWa16].

Example

In the following, a simple example on how to evaluate the fn-coefficients is given. The ground is hereby defined as a Lambertian-surface and the covering layer is assumed to consist of Rayleigh-particles. Thus, we have: (R_0 hereby denotes the diffuse albedo of the surface)

- $BRDF(\theta,\phi,\theta_{ex},\phi_{ex}) = \frac{R_0}{\pi}$
- $p(\theta,\phi,\theta_{ex},\phi_{ex}) = \frac{3}{16\pi}(1+\cos(\Theta)^2)$ with $\cos(\Theta) = \cos(\theta)\cos(\theta_{ex}) + \sin(\theta)\sin(\theta_{ex})\cos(\phi-\phi_{ex})$

$$\begin{aligned} INT &= \int_0^{2\pi} p(\theta_0,\phi_0,\theta,\phi) * BRDF(\pi-\theta,\phi,\theta_{ex},\phi_{ex})d\phi \\ &= \frac{3R_0}{16\pi^2}\int_0^{2\pi}(1+[\cos(\theta_0)\cos(\theta)+\sin(\theta_0)\sin(\theta)\cos(\phi_0-\phi)]^2)d\phi \\ &= \frac{3R_0}{16\pi^2}\int_0^{2\pi}(1+\mu_0^2\mu^2+2\mu_0\mu\sin(\theta_0)\sin(\theta)\cos(\phi_0-\phi)+(1-\mu_0)^2(1-\mu)^2\cos(\phi_0-\phi)^2d\phi \end{aligned} \tag{3.17}$$

where the shorthand-notation $\mu_x = \cos(\theta_x)$ has been introduced.

The above integral can now easily be solved by noticing:

$$\int_0^{2\pi}\cos(\phi_0-\phi)^n d\phi = \begin{cases} 2\pi & \text{for } n=0 \\ 0 & \text{for } n=1 \\ \pi & \text{for } n=2 \end{cases} \tag{3.18}$$

Using some algebraic manipulations we therefore find:

$$INT = \frac{3R_0}{16\pi}\left[(3-\mu_0^2)+(3\mu_0-1)\mu^2\right] = \sum_{n=0}^{2} f_n\,\mu^n \tag{3.19}$$

$$\Rightarrow \quad f_0 = \frac{3R_0}{16\pi}(3-\mu_0^2) \qquad f_1 = 0 \qquad f_2 = \frac{3R_0}{16\pi}(3\mu_0-1) \qquad f_n = 0\,\forall\, n>2$$

An IPython-notebook that uses the RT1-module to evaluate the above fn-coefficients can be found HERE

References

CHAPTER

FOUR

MODEL SPECIFICATION

4.1 Evaluation Geometries

From the general definition of the fn-coefficients (3.16) it is apparent that they are in principle dependent on $\theta_0, \phi_0, \theta_{ex}$ and ϕ_{ex}. If the series-expansions ((3.13) and (3.14)) contain a high number of Legendre-polynomials, the resulting fn-coefficients turn out to be rather lengthy and moreover their evaluation might consume a lot of time. Since usually one is only interested in an evaluation with respect to a specific (a-priori known) geometry of the measurement-setup, the rt1-module incorporates a parameter that allows specifying the geometry at which the results are being evaluated. This generally results in a considerable speedup for the fn-coefficient generation.

The measurement-geometry is defined by the value of the `geometry`-parameter of the RT1-class object:

```
R = RT1(... , geometry = '????')
```

The `geometry`-parameter must be a **4-character string** that can take one of the following possibilities:

- `'mono'` for *Monostatic Evaluation*
- any combination of `'f'` and `'v'` for *Bistatic Evaluation*

To clarify the definitions, the used angles are illustrated in Fig. 4.1.

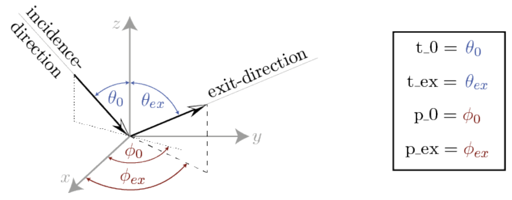

Fig. 4.1: Illustration of the used incidence- and exit-angles

4.1.1 Monostatic Evaluation

Monostatic evaluation refers to measurements where both the transmitter and the receiver are at the same location.

In terms of spherical-coordinate description, this is equal to (see Fig. 4.1):

$$\begin{aligned}\theta_{ex} &= \theta_0 \\ \phi_{ex} &= \phi_0 + \pi\end{aligned} \tag{4.1}$$

Since a monostatic setup drastically simplifies the evaluation of the fn-coefficients, setting the module exclusively to monostatic evaluation results in a considerable speedup.

The module is set to be evaluated at monostatic geometry by setting:

```
R = RT1(... , geometry = 'mono')
```

Note:

- If `geometry` is set to `'mono'`, the values of `t_ex` and `p_ex` have no effect on the calculated results since they are automatically set to `t_ex = t_0` and `p_ex = p_0 + Pi`
- For azimuthally symmetric phase-functions[1], the value of `p_0` has no effect on the calculated result and the best performance will be achieved by setting `p_0 = 0.`

4.1.2 Bistatic Evaluation

Any possible bistatic measurement geometry can be chosen by manually selecting the angles that shall be treated symbolically (i.e. variable), and those that are treated as numerical constants (i.e. fixed).

The individual characters of the `geometry`-string hereby represent the properties of the incidence- and exit angles (see Fig. 4.1) in the order of appearance within the RT1-class element, i.e.:

```
geometry[0] ... t_0
geometry[1] ... t_ex
geometry[2] ... p_0
geometry[3] ... p_ex
```

- **The character `'f'` indicates a fixed angle**
  - The given numerical value of the angle will be used rather than it's symbolic representation to speed up evaluation.
  - The resulting fn-coefficients are only valid for the chosen specific value of the angle.
- **The character `'v'` indicates a variable angle**
  - The angle will be treated symbolically when evaluating the fn-coefficients in order to provide an analytic representation of the interaction-term where the considered angle is treated as a variable.
  - The resulting fn-coefficients can be used for any value of the angle.

As an example, the choice `geometry = 'fvfv'` represents a measurement setup where the surface is illuminated at constant (polar- and azimuth) incidence-angles and the location of the receiver is variable both in azimuth- and polar direction.

Note:

- Whenever a single angle is set *fixed*, the calculated fn-coefficients are only valid for this specific choice!
- If the chosen scattering-distributions reqire an approximation with a high degree of Legendre-polynomials, evaluating the interaction-contribution with `geometry = 'vvvv'` might take considerable time since the resulting fn-coefficients are very long symbolic expressions.

[1] This referrs to any phase-function whose generalized scattering angle parameters satisfy `a[0] = ?, a[1] == a[2] = ?`. The reason for this simplification stems from the fact that the azimuthal dependency of a generalized scattering angle with `a[1] == a[2]` can be expressed in terms of $\cos(\phi_0 - \phi_{ex})^n$. For the monostatic geometry this reduces to $\cos(\pi)^n = 1$ independent of the choice of ϕ_0.

4.2 Linear combination of scattering distributions

Aside of directly specifying the scattering distributions by choosing one of the implemented functions, the RT1-module has a method to define linear-combinations of scattering distributions to allow consideration of more complex scattering characteristics.

> An IPython-notebook that shows the basic usage of linear-combinations within the RT1-module is provided HERE .

4.2.1 Combination of volume-scattering phase-functions

Linear-combination of volume-scattering phase-functions is used to generate a combined volume-class element of the form:

$$\hat{p}_{combined}(\theta_0, \phi_0, \theta_{ex}, \phi_{ex}) = \sum_{n=0}^{N} w_n * \hat{p}_n(\cos(\Theta_{a_n})) = \sum_{n=0}^{N} w_n * \sum_{k=0}^{K_n} \hat{P}_k(\cos(\Theta_{a_n})) * p_k^{(n)} \tag{4.2}$$

where $\hat{p}_n(\cos(\Theta_{a_n}))$ denotes the scattering phase-functions to be combined, $\cos(\Theta_{a_n})$ denotes the individual scattering angles (3.15) used to define the scattering phase-functions w_n denotes the associated weighting-factors, $p_k^{(n)}$ denotes the k$^{\text{th}}$ Legendre-expansion-coefficient (3.14) of the n$^{\text{th}}$ phase-function and $\hat{P}_k(x)$ denotes the k$^{\text{th}}$ Legendre-polynomial.

Note: Since a volume-scattering phase-function must obey the normalization condition:

$$\int_0^{2\pi} \int_0^{\pi} \hat{p}(\theta, \phi, \theta', \phi') \sin(\theta') d\theta' d\phi' = 1 \tag{4.3}$$

and each individual phase-function that is combined already satisfies this condition, the weighting-factors w_n must equate to 1, i.e.:

$$\sum_{n=0}^{N} w_n = 1 \tag{4.4}$$

Within the RT1-module, linear-combination of volume-scattering phase-functions is performed by the `LinCombV(omega, tau, Vchoices)` function:

```
from rt1.volume import LinCombV
```

In order to generate a combined phase-function, one must provide the optical depth `tau`, the single-scattering albedo `omega` and a list of volume-class elements along with a set of weighting-factors (`Vchoices`) of the form:

```
Vchoices = [ [weighting-factor , function] , [weighting-factor , function] , ...  ]
```

Once the functions and weighting-factors have been defined, the combined phase-function is generated via:

```
V = LinCombV(tau, omega, Vchoices)
```

The resulting volume-class element can now be used completely similar to the pre-defined scattering phase-functions.

Note: Since one can combine functions with different choices for the generalized scattering angle (i.e. the `a`-parameter), and different numbers of expansion-coefficients (the `ncoefs`-parameter) `LinCombV()` will automatically combine the associated Legendre-expansions based on the choices for `a` and `ncoefs`.

The parameters `V.a`, `V.scat_angle()` and `V.ncoefs` of the resulting volume-class element are therefore **NOT** representative for the generated combined phase-function!

4.2.2 Combination of BRDF's

Linear-combination of BRDF's is used to generate a combined surface-class element of the form:

$$BRDF_{combined}(\theta_0, \phi_0, \theta_{ex}, \phi_{ex}) = \sum_{n=0}^{N} w_n * BRDF_n(\cos(\Theta_{a_n})) = \sum_{n=0}^{N} w_n * \sum_{k=0}^{K_n} \hat{P}_k(\cos(\Theta_{a_n})) * b_k^{(n)} \tag{4.5}$$

where $BRDF_n(\cos(\Theta_{a_n}))$ denotes the BRDF's to be combined, $\cos(\Theta_{a_n})$ denotes the individual scattering angles (3.15) used to define the BRDF's w_n denotes the associated weighting-factors, $b_k^{(n)}$ denotes the k$^{\text{th}}$ Legendre-expansion-coefficient (3.13) of the n$^{\text{th}}$ BRDF and $\hat{P}_k(x)$ denotes the k$^{\text{th}}$ Legendre-polynomial.

Note: Since a BRDF must obey the following normalization condition:

$$\int_0^{2\pi} \int_0^{\pi/2} BRDF(\theta, \phi, \theta', \phi') \cos(\theta') \sin(\theta') d\theta' d\phi' = R(\theta_0, \phi_0) \leq 1 \tag{4.6}$$

there is in principle no restriction on the weighting-factors for combination of BRDF's!

It is however important to notice that the associated hemispherical reflectance $R(\theta_0, \phi_0)$ must always be lower or equal to 1. In order to provide a simple tool that allows validating the above condition, the function `RT1.Plots().hemreflect()` numerically evaluates the hemispherical reflectance using a simple Simpson-rule integration-scheme and generates a plot that displays $R(\theta_0, \phi_0)$.

Within the RT1-module, linear-combination of BRDF's is performed by the `LinCombSRF(omega, tau, SRFchoices)` function:

```
from rt1.surface import LinCombSRF
```

In order to generate a combined phase-function, one must provide a list of surface-class elements along with a set of weighting-factors (`SRFchoices`) of the form:

```
SRFchoices = [ [weighting-factor , function] , [weighting-factor , function] , ...  ]
```

Once the functions and weighting-factors have been defined, the combined BRDF is generated via:

```
SRF = LinCombSRF(SRFchoices)
```

The resulting surface-class element can now be used completely similar to the pre-defined BRDF's.

Note: Since one can combine functions with different choices for the generalized scattering angle (i.e. the `a`-parameter), and different numbers of expansion-coefficients (the `ncoefs`-parameter) `LinCombSRF()` will automatically combine the associated Legendre-expansions based on the choices for `a` and `ncoefs`.

The parameters `SRF.a`, `SRF.scat_angle()` and `SRF.ncoefs` of the resulting surface-class element are therefore **NOT** representative for the generated combined BRDF!

4.3 Current limitations

- The list of pre-defined BRDF's and volume-scattering phase-functions currently contains only few example of possible choices and is intended to be extended. For example the possibility of using delta-peaked scattering-distributions is contemplated for future versions of the code.
- If the expansion-coefficients of the BRDF and volume-scattering phase-function exceed a certain number (approx. ncoefs > 20) one might run into numerical precision errors.

CHAPTER FIVE

EXAMPLES

In the following, we provide reference to a couple of examples that reproduce the figures in the manuscript of Quast & Wagner (2016).

A script that produces all examples is provided in the doc/examples directory. The following links provide however links to IPython notebooks in the hope that this will facilitate the user to better understand the general concepts of the code.

- Example 1 from Quast and Wagner (2016)
- Example 2 from Quast and Wagner (2016)
- Example 3: usage of a linear combination for the phase function and BRDF
- Example 4: basic fitting example

CHAPTER

SIX

TECHNICAL DOCUMENTATION

6.1 RT1-module

Core module for implementation of 1st order scattering model using arbitrary BRDF and phase functions

References

Quast & Wagner (2016): doi:10.1364/AO.55.005379

class `rt1.rt1.`**`RT1`**(*I0*, *t_0*, *t_ex*, *p_0*, *p_ex*, *V=None*, *SRF=None*, *fn_input=None*, *_fnevals_input=None*, *geometry='mono'*, *bsf=0.0*, *param_dict={}*, *lambda_backend='symengine'*, *int_Q=True*, *verbosity=1*)

Bases: `object`

Main class to perform RT-simulations

Parameters

- **I0** (*scalar(float)*) – incidence intensity
- **t_0** (*array_like(float)*) – array of incident zenith-angles in radians
- **p_0** (*array_like(float)*) – array of incident azimuth-angles in radians
- **t_ex** (*array_like(float)*) – array of exit zenith-angles in radians (if geometry is 'mono', theta_ex is automatically set to t_0)
- **p_ex** (*array_like(float)*) – array of exit azimuth-angles in radians (if geometry is 'mono', phi_ex is automatically set to p_0 + np.pi)
- **V** (*rt1.volume*) – random object from rt1.volume class
- **SRF** (*surface*) – random object from rt1.surface class
- **fn_input** (*array_like(sympy expression), optional (default = None)*) – optional input of pre-calculated array of sympy-expressions to speedup calculations where the same fn-coefficients can be used. if None, the coefficients will be calculated automatically at the initialization of the RT1-object
- **_fnevals_input** (*callable, optional (default = None)*) – optional input of pre-compiled function to numerically evaluate the fn_coefficients. if None, the function will be compiled using the fn-coefficients provided. Note that once the _fnevals function is provided, the fn-coefficients are no longer needed and have no effect on the calculated results!
- **geometry** (*str (default = 'vvvv')*) – 4 character string specifying which components of the angles should be fixed or variable. This is done to significantly speed up the evaluation-process of the fn-coefficient generation

The 4 characters represent in order the properties of:
t_0, t_ex, p_0, p_ex

- ’f’ indicates that the angle is treated ‘fixed’ (i.e. as a numerical constant)
- ’v’ indicates that the angle is treated ‘variable’ (i.e. as a sympy-variable)
- Passing geometry = ‘mono’ indicates a monstatic geometry (i.e.: t_ex = t_0, p_ex = p_0 + pi) If monostatic geometry is used, the input-values of t_ex and p_ex have no effect on the calculations!

For detailed information on the specification of the geometry-parameter, please have a look at the “Evaluation Geometries” section of the documentation: (http://rt1.readthedocs.io/en/latest/model_specification.html#evaluation-geometries)

- **bsf** (*float (default = 0.)*) – fraction of bare-soil contribution (no attenuation due to vegetation)
- **param_dict** (*dict (default = {})*) – a dictionary to assign numerical values to sympy.Symbols appearing in the definitions of V and SRF.
- **lambda_backend** (*str (default = 'symengine' if possible, else 'sympy')*) – indicator to select the module that shall be used to compile a function for numerical evaluation of the fn-coefficients.

  possible values are:

  - ’sympy’ : sympy.lambdify is used to compile the _fnevals function
  - ’symengine’ : symengine.LambdifyCSE is used to compile the _fnevals function. This results in considerable speedup for long fn-coefficients
- **int_Q** (*bool (default = True)*) – indicator whether the interaction-term should be calculated or not
- **verbosity** (*int*) –

  select the verbosity level of the module to get status-reports

  - 0 : print nothing
  - 1 : print some infos during runtime
  - 2 : print more
  - >=3 : print all

`calc()`

Perform actual calculation of bistatic scattering at top of the random volume (z=0) for the specified geometry. For details please have a look at the documentation: (http://rt1.readthedocs.io/en/latest/theory.html#first-order-solution-to-the-rte)

Returns

- **Itot** (*array_like(float)*) – Total scattered intensity (Itot = Isurf + Ivol + Iint)
- **Isurf** (*array_like(float)*) – Surface contribution
- **Ivol** (*array_like(float)*) – Volume contribution
- **Iint** (*array_like(float)*) – Interaction contribution

`property fn`

interaction()

Numerical evaluation of the interaction-contribution (http://rt1.readthedocs.io/en/latest/theory.html#interaction_contribution)

Returns
- – Numerical value of the interaction-contribution for the given set of parameters

Return type
array_like(float)

jacobian(*dB=False*, *sig0=False*, *param_list=['omega', 'tau', 'NormBRDF']*)

Returns the jacobian of the total backscatter with respect to the parameters provided in param_list. (default: param_list = ['omega', 'tau', 'NormBRDF'])

The jacobian can be evaluated for measurements in linear or dB units, and for either intensity- or sigma_0 values.

Note: The contribution of the interaction-term is currently not considered in the calculation of the jacobian!

Parameters

- **dB** (*boolean (default = False)*) – Indicator whether linear or dB units are used. The applied relation is given by:

  dI_dB(x)/dx = 10 / [log(10) * I_linear(x)] * dI_linear(x)/dx

- **sig0** (*boolean (default = False)*) – Indicator wheather intensity- or sigma_0-values are used The applied relation is given by:

  sig_0 = 4 * pi * cos(inc) * I

  where inc denotes the incident zenith-angle and I is the corresponding intensity

- **param_list** (*list*) – a list of strings that correspond to the parameters for which the jacobian should be evaluated.

  possible values are: 'omega', 'tau' 'NormBRDF' and any string corresponding to a sympy.Symbol used in the definition of V or SRF

Returns
jac – The jacobian of the total backscatter with respect to omega, tau and NormBRDF

Return type
array-like(float)

property p_0

property p_ex

prv(*v*, *msg*)

function to set print output based on verbosity level v. possible values for v:

- 0 : print nothing
- 1 : print some infos during runtime
- 2 : print more
- >=3 : print all

Parameters

- **v** (*int*) – the verbosity.
- **msg** (*str*) – the message to be printed.

`surface()`

Numerical evaluation of the surface-contribution (http://rt1.readthedocs.io/en/latest/theory.html#surface_contribution)

Returns
- – Numerical value of the surface-contribution for the given set of parameters

Return type
array_like(float)

`surface_curv`(*dB=False*, *sig0=False*)

Numerical evaluation of the curvature (d^2I_s/dt_0^2) of the (!monostatic!) surface-contribution

Parameters

- **dB** (*bool (default = False)*) – indicator if the derivative is calculated for the dB values or for the linear values
- **sig0** (*bool (default = False)*) – indicator if the derivative is calculated for the intensity (False) or for sigma_0 = 4 * pi * cos(t_0) * intensity (True)

Returns
- – Numerical value of the monostatic curvature of the surface-contribution

Return type
array_like(float)

`surface_slope`(*dB=False*, *sig0=False*)

Numerical evaluation of the slope (dI_s/dt_0) of the (!monostatic!) surface-contribution

Parameters

- **dB** (*bool (default = False)*) – indicator if the derivative is calculated for the dB values or for the linear values
- **sig0** (*bool (default = False)*) – indicator if the derivative is calculated for the intensity (False) or for sigma_0 = 4 * pi * cos(t_0) * intensity (True)

Returns
- – Numerical value of the monostatic slope of the surface-contribution

Return type
array_like(float)

`property t_0`

`property t_ex`

`tot_curv`(*sig0=False*, *dB=False*)

numerical value of the (!monostatic!) curvature of total contribution (surface + volume)

Parameters

- **dB** (*bool (default = False)*) – indicator if the derivative is calculated for the dB values or for the linear values
- **sig0** (*bool (default = False)*) – indicator if the derivative is calculated for the intensity (False) or for sigma_0 = 4 * pi * cos(t_0) * intensity (True)

Returns
- – Numerical value of the monostatic curvature of the total-contribution

Return type
array_like(float)

`tot_slope`(*sig0=False*, *dB=False*)

numerical value of the (!monostatic!) slope of total contribution (surface + volume)

Parameters

- **dB** (*bool (default = False)*) – indicator if the derivative is calculated for the dB values or for the linear values
- **sig0** (*bool (default = False)*) – indicator if the derivative is calculated for the intensity (False) or for sigma_0 = 4 * pi * cos(t_0) * intensity (True)

Returns
- – Numerical value of the monostatic slope of the total-contribution

Return type
array_like(float)

`volume`()

Numerical evaluation of the volume-contribution (http://rt1.readthedocs.io/en/latest/theory.html#volume_contribution)

Returns
- – Numerical value of the volume-contribution for the given set of parameters

Return type
array_like(float)

`volume_curv`(*dB=False*, *sig0=False*)

Numerical evaluation of the curvature (d^2I_s/dt_0^2) of the (!monostatic!) volume-contribution

Parameters

- **dB** (*bool (default = False)*) – indicator if the derivative is calculated for the dB values or for the linear values
- **sig0** (*bool (default = False)*) – indicator if the derivative is calculated for the intensity (False) or for sigma_0 = 4 * pi * cos(t_0) * intensity (True)

Returns
- – Numerical value of the monostatic curvature of the volume-contribution

Return type
array_like(float)

`volume_slope`(*dB=False*, *sig0=False*)

Numerical evaluation of the slope (dI_v/dt_0) of the (!monostatic!) volume-contribution

Parameters

- **dB** (*bool (default = False)*) – indicator if the derivative is calculated for the dB values or for the linear values
- **sig0** (*bool (default = False)*) – indicator if the derivative is calculated for the intensity (False) or for sigma_0 = 4 * pi * cos(t_0) * intensity (True)

Returns
- – Numerical value of the monostatic slope of the volume-contribution

Return type
array_like(float)

6.2 rtplots-module

6.3 rtfits-module

6.4 rtparse-module

6.5 Scatter-module

Define general object for scattering calculations

`class rt1.scatter.Scatter`

Bases: `object`

The basis object for any Surface and Volume objects

`scat_angle`(*t_0*, *t_ex*, *p_0*, *p_ex*, *a*)

Function to return the generalized scattering angle with respect to the given zenith-angles (http://rt1.readthedocs.io/en/latest/theory.html#equation-general_scat_angle)

Standard-choices assigned in the volume- and surface class:

- Surface: a=[1,1,1] … pi-shifted scattering angle cos[t_0]*cos[t_ex] + sin[t_0]*sin[t_ex]*cos[p_0 - p_ex]
- Volume: a=[-1,1,1] … ordinary scattering angle -cos[t_0]*cos[t_ex] + sin[t_0]*sin[t_ex]*cos[p_0 - p_ex]

Note: The scattering angle is defined with respect to the incident zenith-angle t_0, and not with respect to the incidence-angle in a spherical coordinate system (t_i)! The two definitions are related via t_i = pi - t_0

Parameters

- **t_0** (*array_like(float)*) – incident zenith-angle
- **p_0** (*array_like(float)*) – incident azimuth-angle
- **t_ex** (*array_like(float)*) – exit zenith-angle
- **p_ex** (*array_like(float)*) – exit azimuth-angle
- **a** (*[float , float , float]*) – generalized scattering angle parameters

Returns
the generalized scattering angle

Return type
float

6.6 Surface-module

6.7 Volume-module

6.8 processing_config

CHAPTER

SEVEN

INDICES AND TABLES

- genindex
- modindex
- search

BIBLIOGRAPHY

[QuWa16] R.Quast and W.Wagner, “Analytical solution for first-order scattering in bistatic radiative transfer interaction problems of layered media,” Appl.Opt.55, 5379-5386 (2016)

PYTHON MODULE INDEX

INDEX