

This work was written as part of one of the author's official duties as an Employee of the United States Government and is therefore a work of the United States Government. In accordance with 17 U.S.C. 105, no copyright protection is available for such works under U.S. Law.

Public Domain Mark 1.0

<https://creativecommons.org/publicdomain/mark/1.0/>

Access to this work was provided by the University of Maryland, Baltimore County (UMBC) ScholarWorks@UMBC digital repository on the Maryland Shared Open Access (MD-SOAR) platform.

Please provide feedback

Please support the ScholarWorks@UMBC repository by emailing scholarworks-group@umbc.edu and telling us what having access to this work means to you and why it's important to you. Thank you.



Radiative interaction of atmosphere and surface: Write-up with elements of code

Sergey Korkin^{a,b,*}, Alexei Lyapustin^b

^a University of Maryland Baltimore County, Baltimore, MD, USA

^b NASA Goddard Space Flight Center, Mail Code: 613, Greenbelt, MD 20771 USA



ARTICLE INFO

Article history:

Received 18 January 2023

Revised 19 April 2023

Accepted 11 May 2023

Available online 15 May 2023

Keywords:

Radiative transfer

Polarization

Atmospheric correction

The Green's function technique

Scientific software

ABSTRACT

In passive satellite remote sensing of the Earth, separation of the path radiance (atmosphere-only contribution) from the surface reflection remains a “significant challenge”. Recent literature names it among the gaps in radiative transfer (RT) topics that “require continued research in the near future”. The challenge comes from multiple reflections (bouncing) between the atmosphere and surface – radiative interaction. In this paper we use a known RT technique, the matrix-operator method (MOM), and a new modification of the monochromatic vector RT (vRT) code IPOL (Intensity and POLarization) to simulate the interaction of a plane-parallel atmosphere and a few widely used surface reflection models.

Following the idea of the Green's function method, IPOL no longer takes the surface model parameters on input. Instead, it provides the path radiance, and the atmospheric reflection and transmission matrices as output. Despite many RT codes use the MOM formalism, this output does not seem common. The surface reflection matrix is computed externally. Therefore, this paper extends the Green's function atmospheric correction technique to the case of polarized light.

Aiming clarity rather than performance, we explain in Python the structure of the surface matrices for the isotropic (Lambertian), directional unpolarized, and polarized ocean reflection models. We then combine these surface matrices and the precomputed IPOL output to get numerically accurate signal at the top of atmosphere (TOA) and test it vs. published benchmarks. Then, for each benchmark scenario we show how to get the surface from the TOA signal, i.e. perform the RT-based atmospheric correction.

© 2023 Elsevier Ltd. All rights reserved.

1. Introduction

In passive satellite remote sensing of the Earth, decoupling (separation) of the surface and atmospheric contributions on its top (TOA) is a key step in atmospheric correction algorithms. There are two reasons to remove the effect of the atmosphere from images of land or ocean. First is to explore the planetary surface without scattering (blurring) and/or absorption (darkening). Second is to use the surface as a constraint in atmospheric studies. For the latter, one retrieves the surface reflectance under known atmospheric conditions, such as clean atmosphere (no clouds, negligible aerosol loading). The known surface is used later or retrospectively, within a reasonably short period of time, to study the atmosphere once the meteorological conditions have changed. In any case, inaccuracy from the decoupling propagates to the final product.

Multiple scattering of the solar light makes the atmospheric effects in the ultraviolet (UV), visible (VIS), and the near infrared (NIR) bands “significant and complex, dramatically altering the spectral nature of the radiation reaching the remote sensor”¹. A special set of analytical and numerical techniques, commonly called Radiative Transfer (RT) [1–3], is used for simulation of light scattering in the atmosphere. Adding surface reflectance makes it even worse. The radiative interaction (coupling) of the atmosphere and the surface remains one of the “gaps in RT topics that require continued research in the near future” [4: pp. 41–42]. And vice versa, at TOA the decoupling (separation) of light scattered in the atmosphere and the one reflected from the surface remains a “significant challenge” [5: p.8] because of multiple bouncing of light at the bottom of atmosphere (BOA). In general, the effect of multiple bouncing makes straightforward (without approximations) separation impossible [6: Abstract].

One of the practically important approximations is an assumption of the isotropically reflecting (Lambertian) surface. Hereinafter,

* Corresponding author at: University of Maryland Baltimore County, Baltimore, MD, USA.

E-mail address: sergey.v.korkin@nasa.gov (S. Korkin).

¹ https://en.wikipedia.org/wiki/Atmospheric_correction

“isotropic” means the surface reflects light evenly at all directions regardless of the direction of irradiation. This term should not be confused with “homogeneous” – meaning same spatial (across the surface) reflection properties. In what follows we restrict ourselves to the spatially homogeneous, non-emitting, surfaces. We use the term “scattering” for the light changing direction in the atmosphere, “reflection” for the same effect at the surface, and “bouncing” when the atmosphere and surface interact. “Multiple” scattering, including the first one, generates “diffuse light”. Lastly, we use the words “light”, “radiation”, “radiance” (but not “irradiance”), “intensity”, and “signal” interchangeably in this paper.

Although shadows from trees, spatially oriented city buildings, and uneven landscape violate the assumption of the isotropic Lambertian reflection, this surface model yields an equation for the TOA signal [1, p. 273, 7: Sec. III]:

$$I(\mu, \mu_0, \varphi) = J(\mu, \mu_0, \varphi) + \frac{\rho}{\pi} \frac{T(\mu)E(\mu_0)}{1 - \rho C_0}. \quad (1)$$

Eq. (1) assumes the atmosphere is plane-parallel; μ , μ_0 , and φ are cosines of the view (VZA) and solar (SZA) zenith angle, and the relative azimuth angle (RAA), respectively. I is the upwelling total intensity at TOA, J is the upwelling diffuse path radiance (scattered radiation that never hits the surface), ρ is the surface albedo, E is the surface irradiance from the downwelling diffuse path radiance and the direct solar beam, T is the upward atmospheric transmittance (direct and diffuse – see Fig. 1), C_0 is the atmospheric spherical albedo at BOA (see Section 2.3.1).

I , J , and E/π are in the units of radiance; ρ , T , and C_0 are unitless. Scaling Eq. (1) by $\pi/(\mu_0 F_0)$, where F_0 is the solar irradiance at TOA, makes all elements in Eq. (1) unitless. In particular, the product of $T(\mu)E(\mu_0)$ becomes unitless bidirectional transmittance – a parameter that defines transparency of the atmosphere. The path radiance J is a solution to the RT equation over a black surface. It is the only parameter in Eq. (1) contributing to the dependence of the TOA signal I on azimuth. The product of T and E , and C_0 , are often precomputed as look-up tables (LUTs) using three forward RT simulations: for a black surface and two user-defined values of the surface albedo ρ . Thus, Eq. (1) yields ρ from the TOA measurement $I(\mu, \mu_0, \varphi)$, and the precomputed $J(\mu, \mu_0, \varphi)$, $T(\mu)E(\mu_0)$ and C_0 . For an ideal Lambertian surface, all ρ -s would be the same. But, because of the shadows, one gets an array of noticeably different values $\rho(\mu, \mu_0, \varphi)$ – the Lambertian equivalent reflectance (LER):

$$I(\mu, \mu_0, \varphi) \approx J(\mu, \mu_0, \varphi) + \frac{\rho(\mu, \mu_0, \varphi)}{\pi} \frac{T(\mu)E(\mu_0)}{1 - \rho(\mu, \mu_0, \varphi)C_0}. \quad (2)$$

Strictly speaking, Eqs. (1) and (2) ignore the diffuse transfer of the azimuthally dependent – not to mention polarized – surface reflectance from BOA to TOA. But they account for bouncing of light at BOA as the sum of an infinite geometric series: $\rho/(1 - \rho C_0)$. The analytical simplicity of Eqs. (1) and (2) determines their broad application. For example, in the standard Moderate Resolution Imaging Spectroradiometer (MODIS) surface reflectance product [8: Sec. 2.1], and in retrieval of atmospheric parameters in the Dark-Target [9: p.4085], Deep Blue [10: p.559], Visible Infrared Imaging Radiometer Suite (VIIRS) [11: Sec. 4], Ozone Monitoring Instrument (OMI) [12: p.4], and other algorithms. The Multi-Angle Implementation of Atmospheric Correction (MAIAC) algorithm uses Eq. (2) to characterize the spectral dependence of the surface reflection and in aerosol retrievals – but not for the atmospheric correction [13: Sec. 3].

Eq. (2) gives independent LER values for the given solar-view geometries. In order to get LER at a different solar-view geometry, e.g. for aerosol retrievals at days different from the clear ones, the LER is interpolated over angles. The Deep Blue algorithm uses a second-order polynomial fit over the scattering angle (the one between the solar beam direction and the direction of observation)

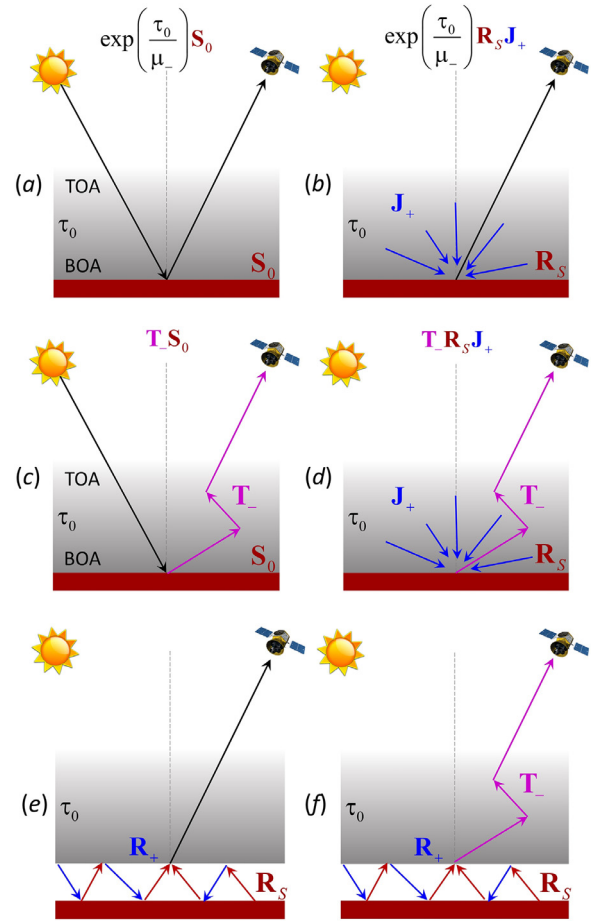


Fig. 1. The TOA contribution from different physical effects at BOA: (a) direct transmittance of the reflected direct solar beam; (b) direct transmittance of the reflected diffuse path radiance; (c) diffuse transmittance of the reflected direct solar beam; (d) diffuse transmittance of the reflected path radiance; (e) direct transmittance of multiple bouncing at BOA; (f) diffuse transmittance of multiple bouncing at BOA. See Eq. (38) and Figs. 4 and 5 below for structure of the transmittance matrix \mathbf{T}^m .

[14: p. 10]; OMI NO_2 fits the Ross-Thick Lee-Sparse (RTLS – see Section 6.2) model [15: p.459]. However, it was shown that the Lambertian assumption “smooths out” the realistic angular dependence of the surface reflectance [16: Sec. 6].

Depending on application, one may use other forms of Eq. (2), e.g.

$$I(\mu, \mu_0, \varphi) \approx J(\mu, \mu_0, \varphi) + \frac{\rho(\mu_0, \mu, \varphi)}{\pi} \frac{T(\mu)E(\mu_0)}{1 - q(\mu_0)C_0} \quad (3)$$

in the snow grain size retrievals [17: p.1979]. In Eq. (3), the surface albedo $q(\mu_0)$ depends on actual angular distribution of the downwelling path radiance – hence the dependence on μ_0 in the multiple bouncing term (see Eq. (6.46) in [18]). In case of multi-angle observations, the new Multi-angle Imaging Spectroradiometer (MISR) research aerosol retrieval algorithm estimates the surface albedo A from measurements [19: p.7 and 10], not from numerical simulations:

$$I(\mu, \mu_0, \varphi) \approx J(\mu, \mu_0, \varphi) + \frac{\rho(\mu_0, \mu, \varphi)}{\pi} \frac{T(\mu)E(\mu_0)}{1 - AC_0}. \quad (4)$$

We deliberately use different notation for the surface albedos in the denominators of Eqs. (2)–(4) because they are obtained differently. In all these equations, the path radiance and the bidirectional atmospheric transmittance come from the RT simulations; the azimuthal dependence of the latter is ignored.

In ocean studies, the water leaving radiance contributes at most ~10% to the TOA signal (in the blue) [20: Section 2.B]. Despite the signal is relatively small, it contains important information, e.g., about phytoplankton – the basis of the marine food chain. Conventionally, the water leaving radiance is retrieved from Eqs. (1) and (2) using single bouncing of light from the isotropic surface: $\rho/(1 - \rho C_0) \approx \rho$ [13: Sec. 3; 21: Sec. 2; 22: p.2089]. This step converts Eq. (2) to the one that accounts for the azimuthal dependence of the path radiance and the surface reflection, but again ignores azimuth in the diffuse bidirectional transmittance [21: Sec. 2; 23: p.555; 24: p.7]

$$I(\mu, \mu_0, \varphi) \approx J(\mu, \mu_0, \varphi) + \rho(\mu, \mu_0, \varphi)T(\mu)E(\mu_0)/\pi. \quad (5)$$

This simplification introduces an error which is unacceptable for certain applications. The ocean chlorophyll algorithms improves accuracy by correcting the diffuse atmospheric transmittance for the azimuthal dependence [25]. However, the correction factor is not always helpful [23: p.555] and it ignores polarization.

Previous studies showed “a very rapid variation of the polarization with azimuthal angle which is observed just beneath the ocean surface for directions of observation near the zenith and nadir and when the sun is not too close to the zenith” [26: Section 8]. Figs. 10 and 11 in [27] show strong azimuthal dependence of the bidirectional properties of the water leaving total and polarized radiances, and their spectral sensitivity to inherent optical properties (chlorophyll concentration, hydrosol type) of the ocean. Like for land, observation of the total and polarized water leaving radiance under a clear-sky condition is desirable because scattering by Rayleigh atmosphere is known. However, it is essential to account for polarization and azimuthal dependence of the bidirectional atmospheric transmittance, especially for the Rayleigh scattering in the blue band.

Contrary to Eqs. (1)–(4), for atmospheric correction MAIAC accurately accounts for the azimuthal dependence of the atmospheric bidirectional transmittance, including multiple bouncing of light at BOA, and thus retrieves shape of the surface reflectance. For that, MAIAC relates the TOA signal and the surface bidirectional reflectance distribution function (BRDF) using the Green's function formalism [18: Sec. 6.4; 28]. The Green's functions describe how atmosphere transfers a beam of light from a given direction (point source) to any direction on the same (reflection) or opposite (transmission) side of the atmosphere. The Green's functions at BOA are solutions to the RT problem adjoint to the one for the atmospheric path radiance (no surface, sequence of the optical layers is flipped upside-down). The plane-parallel scalar spherical harmonics RT code SHARM [18] does this job in MAIAC.

A reflecting surface is a set (infinite) of unidirectional point sources. Each beam contributes to all directions of interest. Their combined effect is a superposition (integral over angles) of the surface BRDF and the Green's functions. By using this approach one (a) simulates atmospheric and surface properties independently; and (b) gets analytical solution for the TOA signal and an arbitrary BRDF. The Green's functions are precomputed as LUTs at nodes of sufficiently dense Gaussian quadrature for numerical integration over the zenith angles; the Fourier expansion is used for integration over the azimuth.

For a kernel driven RTLS surface model (see Section 3.2), MAIAC represents the TOA reflectance as

$$I(\mu, \mu_0, \varphi) = J(\mu, \mu_0, \varphi) + k_L F_L(\mu, \mu_0) + k_V F_V(\mu, \mu_0, \varphi) + k_G F_G(\mu, \mu_0, \varphi) + R_{nl}(\mu, \mu_0). \quad (6)$$

In Eq. (6), $\mathbf{k} = [k_L \ k_V \ k_G]$ are linear parameters (kernel weights). The corresponding F -functions depend on the direct and diffuse transmission of the respective kernel functions of the RTLS model from BOA to TOA using the Green's functions. They also

weakly depend on the surface parameters \mathbf{k} in addition to scaling by the same. Thus, the overall dependence of all $kF(\cdot)$ -products on the kernel weights is weakly nonlinear.

R_{nl} is a residual term in which nonlinear dependence on the surface parameters is relatively strong, but magnitude is small. The nonlinear dependence in F and R_{nl} on \mathbf{k} comes from the multiple bouncing of light at BOA. Analytical formulae for all elements in Eq. (6) are given in [18: Section 6.3.3]. Here we note only that F_L does not depend on the azimuth. The same dependence in R_{nl} is assumed negligible for the total intensity. The nonlinear residual is calculated iteratively. The mentioned approximate formula for the residual helps iterations to converge fast. One needs 2 iterations to achieve ~0.5% error in the simulated TOA signal for a relatively thick atmosphere, $\tau \approx 0.5$, and a bright surface, $\rho \approx 0.5$. We use similar iterative approach in this paper and refer our reader to Section 3.2 for details.

MAIAC retrieves the RTLS parameters using the linear least squares technique

$$\chi^2 = \sum (R_{nl}^{(n)} - k_L^{(n)} F_L^{(n)} - k_V^{(n)} F_V^{(n)} - k_G^{(n)} F_G^{(n)})^2 \rightarrow \min(\mathbf{k}), \quad (7)$$

where summation runs over all measurement geometries (e.g., different days in case of MODIS), the non-linear residual at the n -th iteration is $R_{nl}^{(n)} = I - J - R_{nl}^{(n-1)}$, I is the TOA signal, and J is the path radiance. At the first iteration, the method assumes single bouncing of the direct solar beam and the diffuse downwelling path radiance from the surface. For that, the weak nonlinearity of the F -function is neglected, and the residual R_{nl} is assumed zero.

Eq. (6) removes systematic biases of the Lambertian approximation, which ignores the bowl-shaped surface reflectance. The biases grow as atmospheric scattering, hence optical path, grows from the red to blue bands, and/or with the increase of the aerosol loading, and/or with the increase of the solar and view zenith angles [29: p.10]. For the MODIS surface retrievals over several Aerosol Robotic Network (AERONET) sites, the slope of regression of the Lambertian surface reflectance vs. AERONET-based Surface Reflectance Validation Network (ASVRN) data [30] was reported to be 0.85 in the red and 0.6 in the green bands (1 means perfect agreement). In absolute units, the error of the Lambertian approximation results in the albedo reduction by 0.004, 0.005 and 0.008 in the NIR, red, and green MODIS bands, respectively [31].

In this paper, we wish to take advantage of the Green's function formalism and include polarization in Eq. (6). This will support the polarimetric satellite observations of the Earth atmosphere over ocean [32] and land [33]. As mentioned in [28: Sec. 1], “the Green's function technique is closely related to the adding-doubling method”, which in its turn is identical to the invariant embedding and the matrix-operator methods (MOM) [2]. There are several reasons to choose the MOM-based formalism over the extension of the existing scalar Green's function technique to the case of polarization. A MOM-based RT code, no matter scalar or vector, uses the diffuse reflectance and transmittance atmospheric Green's functions, already in discretized form (matrices), as core elements to solve the RT equation. There is no need to calculate these matrices separately. The matrix Green's functions are calculated for TOA and BOA, which makes the solution to the adjoint problem unnecessary. For a vRT code, the MOM elements already account for polarization. Noteworthy, the MOM formalism is the same in scalar and vector case, except for the matrices are “bigger” for polarization. One only needs to get the path radiance and necessary matrices as output parameters from the MOM-based RT code. As we show below, this type of output is not a common practice.

Therefore, our goals in this paper are: a) to report a new modification of the MOM-based vRT code IPOL (Intensity and POLarization) which now provides the MOM matrices as standard output in addition to the path radiance (Stokes vector); and b) using this

output, to show how to “attach” different surface models to get the TOA signal. Since theory for the topic has been well developed, we complement the known equations with elements of Python code explaining both step-by-step. We believe, this approach helps better grasp relevant theory and simultaneously supplies the reader with a ready-to-use tool [34]. Despite its advantages, the “equations + code” bundle still seems to be underutilized in papers on atmospheric optics. Our third goal c) is to provide numerical examples to illustrate the inverse process: how to “detach” the surface from the known atmosphere and thus retrieve the surface model parameters from the TOA signal (decouple atmosphere and surface).

With these goals in mind, we organize our paper as follows. In the next Section 2 “Methodology” we briefly remind the structure and meaning of elements of MOM. The section reviews relevant literature focusing on publicly available MOM-based RT codes (Section 2.1). Then, it explains computation of the TOA signal from the MOM-elements, their physical meaning, how and why the solution can be simplified (Section 2.2), and relevance to the existing approaches (Section 2.3). Section 3 “Code snippets to combine atmosphere and surface” presents Python scripts for coupling of atmosphere with the isotropic (Section 3.1), scalar bidirectional (Section 3.2), and polarized waved ocean (Section 3.3) surface models. In Section 4 “Validation of the Python functions”, we test our scripts vs. published benchmarks for a homogeneous molecular (Rayleigh) atmosphere at two different wavelengths, and an atmosphere with 2 layers: molecular layer over a layer with a molecular-aerosol mix. Section 5 “Notes on the package implementation” offers a few ideas for the code speed up (Section 5.1), lists all files in the package, describes purpose of each, and provides the call graphs for our Python scripts (Section 5.2). In Section 6 we demonstrate on numerical examples how to remove atmosphere, i.e., perform atmospheric correction. This is done for atmospheres from Section 4 and all the three types of the surface models. We do not provide any code for that because Section 6 relies on scripts from Section 3. We summarize the paper in Section 7 and conclude it with 2 appendices. Appendix A provides an extract of information about polarization and expansion of the surface models in the Fourier series. Appendix B lists the total (photometric) and polarimetric measurement uncertainties for some passive polarimeters as reported in literature. We also refer the reader to [35] for definition of the reflectance quantities.

The Python package provided with the paper is complete in terms of quick and independent reproducibility of the results reported in Section 4. The files are available from the journal website and from our GitHub repository.² We now begin with theoretical background for the package.

2. Methodology

This section contains three parts. Section 2.1 recalls the basic idea of MOM and refers the reader to a few MOM-based RT codes. We argue in this section that a useful feature of providing the matrix Green's functions as standard output, alongside with radiance, is underutilized in most of the existing MOM-based codes. Section 2.2 goes in detail of the MOM elements. It shows how the TOA signal is expressed via the path radiance vector, the atmospheric matrices, and the surface matrix \mathbf{R}_s . In Section 2.3 we express the traditional approaches from the Introduction via MOM. The presented information explains theoretical background for subsequent discussion of the Python codes.

2.1. General background of MOM

Suppose we know the diffuse path radiance J_{TOA} (scattered light that never hits the surface) at TOA from a solution to the RT equation – scalar or vector. Except for the Monte Carlo method, the RT equation is solved numerically using some discretization [2,34,36]. To be specific, we assume the discretization over the cosine of the zenith angle, like in the method of discrete ordinates. The same formalism works for the method of spherical harmonics. Suppose further we know how the atmosphere transfers light from all discrete directions to all discrete directions. This is described by a matrix \mathbf{A} (not necessarily square) which depends on the atmospheric optical (scattering and absorption) properties. Finally suppose, we know how a surface reflects light from all downwelling directions to all upwelling ones, \mathbf{R}_s . This matrix depends only on the surface model parameters. For the TOA reflected signal, I_{TOA} , one can write the following equation

$$I_{\text{TOA}} = J_{\text{TOA}} + \mathbf{A} \begin{bmatrix} \mathbf{0} \\ \mathbf{R}_s I_{\text{BOA}} \end{bmatrix}. \quad (8)$$

In Eq. (8), the vector of zeros, $\mathbf{0}$, means there is no downwelling diffuse light at TOA. I_{BOA} is the downwelling radiance at BOA. It has two parts: the direct (not yet scattered) solar beam and the diffuse downwelling radiation. Unlike the other elements in Eq. (8), which we have assumed known, the diffuse downwelling light is not known. I_{BOA} depends on the surface properties through multiple bouncing of light, in addition to explicit scaling by \mathbf{R}_s in Eq. (8). But the important thing is both matrices, \mathbf{A} and \mathbf{R}_s , act as linear operators – hence the name of the matrix-operator method (MOM).

The idea of the method was described by G.G. Stokes as early as in 1860 [37] (see also [38]). Since then, numerous papers, several books and book chapters about the method have been published. We refer our reader to Sec. 2 “Matrix operator theory and invariant imbedding” in [39] for some milestone references and a brief yet sufficient mathematical overview of the method. Adding and adding-doubling methods also use the MOM formalism [2]. In addition to the atmosphere-surface interaction, MOM accounts for variation of the optical properties with height – vertical profile.

Whereas literature describes many MOM-based RT codes, in this paper we refer only to those publicly available. The adding-doubling RT3 and RT4 [40] are two solvers of the PolRadTran model,³ also available from the libRadtran package⁴ [41: Section 3.5]. The Fortran program pbrf.f⁵ computes the Stokes reflection matrix for a semi-infinite plane-parallel layer using the Ambartsumian's invariant imbedding method [42,43]. The System for Transfer of Atmospheric Radiation for polarized radiance⁶ (PSTAR) [44,45] uses MOM to calculate the Stokes vector at discrete Gaussian ordinates, and integration of the source function technique to solve the RT equation at user-defined directions. The advanced doubling-adding model ADA [46] and the advanced matrix-operator method AMOM [47,48] are used in the Community Radiative Transfer Model⁷ (CRTM). The XRTM⁸ package also provides different RT solvers based on MOM.

Among these codes, only Mishchenko's pbrf.f computes a matrix for simulation of the diffuse polarized reflection as a standard output. The other codes usually compute intensity or the Stokes vector (optionally, derivatives) on output. However, pbrf.f is limited to a vertically homogeneous infinitely thick slab. For this

³ <https://nit.coloradolinex.com/polrad.html>

⁴ <http://www.libradtran.org/doku.php>

⁵ <https://www.giss.nasa.gov/staff/mnmishchenko/brf/>

⁶ <http://157.82.240.167/~clastr/>

⁷ <https://www.jcsda.org/jcsda-project-community-radiative-transfer-model>

⁸ <https://reef.atmos.colostate.edu/~gregm/xrtm/>

² https://github.com/korkins/jqsrt2022_mom_ipol

work we refactored our vRT code IPOL. Now it provides the reflection and transmission matrices (elements of \mathbf{A} in Eq. (8)), and the path radiance as standard output. Simulation of the surface reflection, matrix \mathbf{R}_s , is removed from the IPOL solver.

IPOL is a C/C++ vRT code [49] for numerical simulation of multiple scattering of the monochromatic solar light in a plane-parallel atmosphere. The Fortran 90/95 version of IPOL was thoroughly tested during Phase A of the International Polarized Radiative Transfer (IPRT) code intercomparison [50]. Among other scenarios, the IPRT project includes isotropic (Lambertian) and polarized ocean surfaces. At present, IPOL is used to generate LUTs for MAIAC [13], snow [51] and cloud [52] studies, as a reference code for our successive orders RT solver SORD [53,54], and the Gauss-Seidel iterations code GSIT [34]. For this study we used IPOL in Fortran as a reference for the C version, in addition to the published benchmarks [53: p.306].

In IPOL, the discrete-ordinates method [2] replaces the integro-differential RT equation with a system of differential equations for the Stokes vector at discrete view zenith angles (ordinates) that match with the nodes of the Gaussian quadrature. Solution to the system is expressed via exponential matrix. The exponential matrix is computed using the eigen-value decomposition [55,56]. Since the solution to the eigen-problem is time consuming, we use Intel® Math Kernel Library (MKL), now freely available from oneAPI Toolkit. The scaling transformation [57] removes positive eigen-values from the solution for numerical stability. The eigen-values and eigen-vectors for the user-defined directions are excluded from the system of equations and computed with the “augmented” matrix [58: Sec. 8.1; 59: Sec. 2.3]. Unlike in the Fortran version of IPOL, we now skip the Stokes-V component (ellipticity). Thus we avoid the complex numbers in the eigen-decomposition [60,61] and accelerate the solution without loss of accuracy in linear polarization [53,62,63]. Radiation scattered once is calculated exactly. IPOL uses MOM to account for vertical optical profile in the atmosphere and (de-)couple the atmosphere and surface.

The idea of using MOM for decoupling is not new. The matrix Green’s function formalism is discussed in [64]. In [65], the adding method was used to decouple an atmosphere from a horizontally inhomogeneous isotropic surface. In [3] the bidirectional reflection $R(\mu, \mu_0, \varphi)$ and transmission $T(\mu, \mu_0, \varphi)$ functions are used to remove the atmospheric effects from an image of a small Lambertian target located on a uniform Lambertian background. Computational advantage of the MOM-based adding-doubling approach for a thick homogeneous (cloud) layer in combination with the Gauss-Seidel iterations for a thinner (cloud-free) atmosphere with vertical profile is used in [66]. The Markov-Chain-Adding-Doubling code [67] uses the Markov Chain method for a vertically inhomogeneous atmosphere, and the Adding-Doubling for a homogeneous ocean. In support of development of the polarized atmospheric correction algorithms, IPOL now provides the path radiance and the MOM elements on output.

2.2. TOA signal as function of MOM elements

Except for the single scattering, one solves the plane-parallel vRT equation in the Fourier domain:

$$\begin{aligned} I(\tau, \mu, \varphi) &= \sum_{m=0}^M (2 - \delta_{0,m}) I_m(\tau, \mu) \cos(m\varphi), \\ Q(\tau, \mu, \varphi) &= \sum_{m=0}^M (2 - \delta_{0,m}) Q_m(\tau, \mu) \cos(m\varphi), \\ U(\tau, \mu, \varphi) &= \sum_{m=0}^M (2 - \delta_{0,m}) U_m(\tau, \mu) \sin(m\varphi). \end{aligned} \quad (9)$$

In Eq. (9), I is the total intensity, and Q and U are components of the Stokes vector responsible for linear polarization; I_m, Q_m, U_m are their m -th Fourier coefficients; $\delta_{0,m}$ is the Kronecker delta. For the VZA cosine μ , we use the following sign convention: positive and negative cosines correspond to downwelling (towards the surface) and upwelling directions, respectively: $\mu_+ > 0, \mu_- < 0$. In the vector RT community, it is common to define Q and U components in the meridian plane.

To account for the atmospheric optical profile, IPOL splits the vertically inhomogeneous atmosphere into homogeneous layers with given optical properties. MOM accounts for contribution from each optical layer, and from multiple bouncing of light between the layers, into the total signal. The idea of MOM is that each layer can be characterized by the transmittance T and reflectance R functions that depend on two directions before and after (multiple) scattering. In the Fourier space, Eq. (9), T and R do not depend on the relative azimuth, hence the directions are described solely by the zenith angles. In case of diffuse light, there are many possible directions for the light to enter and escape the layer. Hence, one can combine all T and R functions of two variables, representing upward and downward directions, into corresponding matrices (not necessarily square). Therefore, MOM uses four matrices for each optical layer. From these matrices, one derives the MOM expression for the whole atmosphere – the matrix \mathbf{A} in Eq. (8). For a homogeneous layer containing isotropic and mirror-symmetric particles, \mathbf{A} is symmetric. Otherwise, the atmosphere affects upwelling and downwelling light differently. However, the form of the matrix equation is the same for one optical layer and for many layers combined in one:

$$\begin{bmatrix} \mathbf{I}_-^m \\ \mathbf{I}_+^m \end{bmatrix} = \begin{bmatrix} \mathbf{J}_-^m \\ \mathbf{J}_+^m \end{bmatrix} + \begin{bmatrix} \mathbf{R}_-^m & \mathbf{T}_-^m \\ \mathbf{T}_+^m & \mathbf{R}_+^m \end{bmatrix} \begin{bmatrix} \mathbf{0} \\ \mathbf{I}_s^m \end{bmatrix}. \quad (10)$$

In Eq. (10), “-” and “+” denote the upwelling and downwelling radiation, respectively; \mathbf{I}_-^m and \mathbf{J}_-^m are the Fourier components for the total and path radiances (the latter never hits the surface) propagating at user-defined directions (hence, each is a vector). Elements in the top and bottom rows of Eq. (10) correspond to TOA and BOA, respectively. For example, \mathbf{I}_-^m is the upward radiance at the TOA, \mathbf{I}_+^m is the downward radiance at the BOA.

\mathbf{R}_-^m models how the light that irradiates the atmosphere from the top is reflected to the space (upward - hence the sign); \mathbf{T}_-^m is a matrix which describes how the light that goes upward at the surface level, \mathbf{I}_s^m , is transmitted directly and diffusely (by multiple scattering) through the atmosphere to its top. Likewise, \mathbf{T}_+^m describes the transfer of the top boundary condition through the atmosphere to its bottom; and \mathbf{R}_+^m describes how the surface boundary contribution \mathbf{I}_s^m is reflected back towards the surface. Eq. (10) looks the same with or without polarization.

Later in the paper, we define the size of all elements in Eq. (10) necessary to compute the TOA signal. Here we note that the surface contribution, \mathbf{I}_s^m , depends on the reflected downwelling radiation at BOA, \mathbf{I}_+^m , and the reflected direct solar beam (which is the only source of light considered in this paper)

$$I_d(\tau, \mu, \mu_0, \varphi) = F_0 \exp\left(-\frac{\tau}{\mu_0}\right) \delta(\mu - \mu_0) \delta(\varphi). \quad (11)$$

The surface reflects the solar beam at many directions. In the Fourier m -space, they make up a vector \mathbf{S}_0^m . In Eq. (11), the angular distribution of intensity of the direct solar beam is represented by the Dirac delta-function $\delta(x)$. F_0 is the extraterrestrial TOA irradiance (in units of flux per unit square). Like the single scatter path radiance, \mathbf{S}_0^m is known analytically because the delta-functions remove integrals over angles.

Let \mathbf{R}_S^m be a Fourier moment for a matrix that “reflects” the downwelling diffuse light at BOA \mathbf{I}_+^m . Eqs. (10) and (11) give

$$\begin{bmatrix} \mathbf{I}_+^m \\ \mathbf{I}_-^m \end{bmatrix} = \begin{bmatrix} \mathbf{J}_+^m \\ \mathbf{J}_-^m \end{bmatrix} + \begin{bmatrix} \mathbf{R}_+^m & \mathbf{T}_+^m \\ \mathbf{T}_-^m & \mathbf{R}_-^m \end{bmatrix} \begin{bmatrix} \mathbf{0} \\ \mathbf{R}_S^m \mathbf{I}_+^m + \mathbf{S}_0^m \end{bmatrix}. \quad (12)$$

In Eq. (12) only \mathbf{R}_S^m and \mathbf{S}_0^m depend on the surface parameters. By eliminating \mathbf{I}_+^m , one gets the MOM expression for the TOA signal in the Fourier space

$$\mathbf{I}_-^m = \mathbf{J}_-^m + \mathbf{T}_-^m \mathbf{S}_0^m + \mathbf{T}_-^m \mathbf{R}_S^m (\mathbf{1} - \mathbf{R}_+^m \mathbf{R}_S^m)^{-1} (\mathbf{J}_+^m + \mathbf{R}_+^m \mathbf{S}_0^m). \quad (13)$$

Eq. (13) has clear physical meaning. The TOA signal \mathbf{I}_-^m consists of the path radiance \mathbf{J}_-^m , direct (exponential: $\exp(\tau_0/\mu_-)$) and diffuse (scattered) transmittance of the direct solar beam bounced once from the surface $\mathbf{T}_-^m \mathbf{S}_0^m$, the rest is multiple bouncing of light between the atmosphere and the surface. Fig. 1 illustrates the TOA contribution by single (a-d) and multiple (e-f) bouncing of light at BOA. Charts (a-b) and (e) correspond to the direct transfer from BOA to TOA; other charts show the transfer via scattering. We refer our reader to Eq. (38), Figs. 4 and 5 below for structure of the transmittance matrix \mathbf{T}^m .

Let's consider the elements of the multiple bouncing part in Eq. (13) from right to left. The atmosphere-surface bouncing arises from the direct solar beam reflection which, after bouncing back towards the surface as $\mathbf{R}_+^m \mathbf{S}_0^m$, adds to the downwelling path radiance \mathbf{J}_+^m . Their combination $(\mathbf{J}_+^m + \mathbf{R}_+^m \mathbf{S}_0^m)$ begins to bounce back and forth infinitely long at BOA. This infinite number of bouncing is described by the sum of the geometrical series $(\mathbf{1} - \mathbf{R}_+^m \mathbf{R}_S^m)^{-1}$ (see Eqs. (1) and (2) in this paper and Eqs. (4) and (5) in [37]). The downwelling radiation, enforced to some degree by the infinite bouncing, is reflected from the surface (i.e., scaled by \mathbf{R}_S^m) and transmitted directly, $T_{dir} = \exp(-\tau_0/\mu_-)$, and diffusely (via multiple scattering) through the atmosphere of the total optical thickness τ_0 (i.e., scaled by \mathbf{T}_-^m). Eq. (13) is general in a sense that it includes multiple bouncing at BOA, azimuthal dependence of the atmospheric transmittance and the surface reflectance, and polarization. The Fourier expansion is the only limitation in Eq. (13).

2.3. Conventional approaches via elements of MOM

In this section we express Eq. (1) for multiple and Eq. (5) for single bouncing of light from surface, and the Green's function approach, Eq. (6), via elements of MOM. For the Lambertian case, the MOM approach offers an advantage over Eq. (1): instead of running the RT code 3 times (see Introduction), one could get all necessary parameters by running the MOM-based code once. For Eq. (5), the MOM version accounts for the azimuthal and polarization dependences of the atmospheric bidirectional transmittance. The azimuthal dependence is included in MAIAC's Eq. (6), the MOM version of which accounts for polarization.

2.3.1. Lambertian reflection

In order to derive Eq. (1) from Eq. (13), we use the geometric series expansion

$$(\mathbf{1} - \mathbf{R}_+^m \mathbf{R}_S^m)^{-1} = \mathbf{1} + \mathbf{R}_+^m \mathbf{R}_S^m + [\mathbf{R}_+^m \mathbf{R}_S^m]^2 + \dots \quad (14)$$

for modification of the multiple bouncing factor

$$\mathbf{R}_S^m (\mathbf{1} - \mathbf{R}_+^m \mathbf{R}_S^m)^{-1} = (\mathbf{1} - \mathbf{R}_S^m \mathbf{R}_+^m)^{-1} \mathbf{R}_S^m. \quad (15)$$

Note, the order of matrices in the matrix-matrix multiplication in the brackets of Eqs. (14)-(15) has changed. Using Eq. (15), we focus on the \mathbf{S}_0^m -terms in Eq. (13) (the common matrix factor \mathbf{T}_-^m is dropped for the instance)

$$\mathbf{S}_0^m + \mathbf{R}_S^m (\mathbf{1} - \mathbf{R}_+^m \mathbf{R}_S^m)^{-1} \mathbf{R}_+^m \mathbf{S}_0^m = (\mathbf{1} - \mathbf{R}_S^m \mathbf{R}_+^m)^{-1} \mathbf{S}_0^m, \quad (16)$$

and convert Eq. (13) to

$$\mathbf{I}_-^m = \mathbf{J}_-^m + \mathbf{T}_-^m (\mathbf{1} - \mathbf{R}_S^m \mathbf{R}_+^m)^{-1} (\mathbf{R}_S^m \mathbf{J}_+^m + \mathbf{S}_0^m). \quad (17)$$

Eq. (13) simulates multiple bouncing at BOA starting from the downward radiation that is initially reflected from the surface up and then from the atmosphere down (hence, the order $\mathbf{R}_+^m \mathbf{R}_S^m$ - to be read from right to left). Contrary to that, Eq. (17) starts from the upward radiation which is at first reflected from the atmosphere down, and then from the surface up (hence, the order is reversed $\mathbf{R}_S^m \mathbf{R}_+^m$). Note that in general the two matrices are not commutable.

$$\mathbf{R}_+^m \mathbf{R}_S^m \neq \mathbf{R}_S^m \mathbf{R}_+^m. \quad (18)$$

From Eqs. (1) and (17), one notices the equivalence of the surface brightness $E\rho/\pi$ and $\mathbf{R}_S^m \mathbf{J}_+^m + \mathbf{S}_0^m$, and ρ and \mathbf{R}_S^m for the azimuthally symmetric component: $m = 0$.

The atmospheric spherical albedo C_0 is obtained from \mathbf{R}_+^m as follows. Suppose \mathbf{r}_+ is a subset of \mathbf{R}_+^m responsible for reflection of the azimuthally independent ($m = 0$) upward intensity I_0 that irradiates the atmosphere from the bottom. The corresponding irradiance is

$$E_- = \int_0^{2\pi} \int_0^{-1} I_0 \mu d\mu d\varphi = \pi I_0, \quad (19)$$

and reflection of I_0 from the atmosphere is

$$\mathbf{r}_+ \mathbf{I}_-^0 = [\mathbf{r}_+]_{N \times N} \begin{bmatrix} I_0 \\ I_0 \\ \vdots \\ I_0 \end{bmatrix}_N = I_0 \begin{bmatrix} \sum_{j=1}^N \mathbf{r}_+^{1j} \\ \vdots \\ \sum_{j=1}^N \mathbf{r}_+^{ij} \\ \vdots \\ \sum_{j=1}^N \mathbf{r}_+^{Nj} \end{bmatrix}_N = \mathbf{I}_+^0. \quad (20)$$

In Eq. (20), i and j are the row and column indices, respectively, both running from 1 to N . Numerical integration of this downward intensity over angles using the Gaussian quadrature yields downward flux

$$\begin{aligned} E_+ &= \int_0^{2\pi} \int_0^1 I_+^0 \mu d\mu d\varphi \\ &= 2\pi \int_0^1 I_+^0 \mu d\mu \approx 2\pi I_0 \sum_{i=1}^N w_i \mu_i \sum_{j=1}^N \mathbf{r}_+^{ij}. \end{aligned} \quad (21)$$

Ratio of the fluxes, Eqs. (21) and (19), is the atmospheric spherical albedo at BOA calculated from elements of \mathbf{R}_+^m as (note the factor of 2) [3: p.6]

$$C_0 = 2 \sum_{i=1}^N w_i \mu_i \sum_{j=1}^N \mathbf{r}_+^{ij}. \quad (22)$$

It is worthwhile to check numerically Eq. (22) vs. C_0 computed directly from Eq. (1) or as the largest eigenvalue of the matrix-matrix multiplication (note the equality is true despite Eq. (18))

$$\rho C_0 = \max(\text{eig}(\mathbf{R}_S^0 \mathbf{R}_+^0)) = \max(\text{eig}(\mathbf{R}_+^0 \mathbf{R}_S^0)). \quad (23)$$

In Eq. (23) the superscript “0” is the Fourier moment $m = 0$ - azimuthally averaged component.

2.3.2. Single bouncing of light from surface

Contrary to Eq. (1), Eq. (13) accounts for the directional reflection and polarization. Therefore a MOM-based RT code is a convenient tool to study the polarization properties of land and ocean from orbit [68–72]. Moreover, using a simplification, one can reduce the non-linear (w.r.t. the surface parameters) problem, Eq. (13), to the linear one, if the surface model is also linear w.r.t. its parameters.

The simplification has a clear physical meaning. Due to absorption of light in the atmosphere and at the surface, the energy of each subsequent bouncing vanishes. By cutting the series expansion

$$(\mathbf{1} - \mathbf{R}_+^m \mathbf{R}_+^m)^{-1} = \mathbf{1} + \mathbf{R}_+^m \mathbf{R}_+^m + [\mathbf{R}_+^m \mathbf{R}_+^m]^2 + \dots \quad (24)$$

one simulates a finite number of the atmosphere-surface bouncing. It is known, e.g., from the method of successive orders of scattering that higher Fourier moments m require less terms in Eq. (24) due to natural angular smoothing of the intensity over angles after each reflection. Assuming $\mathbf{R}_+^m \mathbf{R}_+^m = \mathbf{0}$ in Eq. (24) one simplifies Eq. (13) to the form of Eq. (5) but with the full dependence on azimuth and polarization. The Fourier expansion (index m) accounts for the azimuthal dependence of the surface reflectance and the atmospheric transmittance:

$$\mathbf{I}_-^m = \mathbf{J}_-^m + \mathbf{T}_-^m \mathbf{S}_0^m + \mathbf{T}_-^m \mathbf{R}_+^m \mathbf{J}_+^m. \quad (25)$$

Looking at the last term in Eq. (25) from left to right, one notes that it describes the direct and diffuse BOA-to-TOA transmittance of the surface reflection of the downwelling path radiance. If the surface model is a linear function of its parameters, like RTLS, these parameters can be determined from Eq. (25) using the linear least squares technique. The two terms in Eq. (24) yield better accuracy in case of thick atmospheres and/or bright surface. We discuss a few relevant numerical examples later.

2.3.3. Matrix form of the Green's function method

Eqs. (13), (24), and (25) are the matrix form of the Green's function method [28,73] which MAIAC uses to compute LUTs for the RTLS surface model [74]. Using this method, MAIAC accurately corrects for the atmospheric effects in satellite measurements [75,76]. However, the Green's functions are not the standard output of the MAIAC's scalar spherical harmonics RT solver SHARM [77]. Leaving derivatives aside, a typical RT code provides intensity (radiance) or the Stokes vector [50,78,79] on output. However, for the atmospheric correction, for the decrease of the size of LUTs by eliminating of the surface properties, for simplicity of adding of a new surface to the existing RT solver (flexibility) it is practical to provide \mathbf{R}_+^m and \mathbf{T}_-^m matrices, in addition to the path radiance, as the standard output. The MOM-based RT codes make this option possible as well as account for polarization in the atmospheric Green's functions.

Eq. (25) accounts for the path radiance and single bouncing of light from surface. Using it, we rewrite Eq. (17) as

$$\mathbf{I}_- = \mathbf{J}_- + \sum_m (\mathbf{T}_-^m \mathbf{S}_0^m + \mathbf{T}_-^m \mathbf{R}_+^m \mathbf{J}_+^m) + \mathbf{R}_{2+}, \quad (26)$$

where \mathbf{I}_- , \mathbf{J}_- , and \mathbf{R}_{2+} are, respectively, the TOA signal, path radiance, and contribution (residual) from the second and higher bouncing at BOA. In general, these three elements possess azimuthal dependence. The m -summation in Eq. (26) converts the TOA contribution by the single bouncing of light at BOA from the Fourier to the azimuthal space.

For the RTLS model, one can convert Eq. (26) to the form of Eq. (5)

$$\mathbf{I}_- = \mathbf{J}_- + k_L \mathbf{F}_L + k_V \mathbf{F}_V + k_G \mathbf{F}_G + \mathbf{R}_{2+}, \quad (27)$$

where, e.g., for the volumetric term,

$$k_V \mathbf{F}_V = k_V \sum_m (\mathbf{T}_-^m \hat{\mathbf{S}}_0^m + \mathbf{T}_-^m \hat{\mathbf{R}}_+^m \mathbf{J}_+^m), \quad (28)$$

In Eq. (28), the vector for the direct solar beam reflection $\hat{\mathbf{S}}_0^m$ and the matrix for reflection of the diffuse downwelling path radiance $\hat{\mathbf{R}}_+^m$ are filled with values of the volumetric kernel function (note the cap symbol ^). The kernel weight k_V is a common scaling factor. Same happens to the Lambertian and the geometric terms,

except for the Lambertian one does not depend on the azimuth: $m = 0$. Therefore, the three kernel weights are obtained by fitting the simulated TOA signal to the measured one using the linear least squares minimization. The residual is calculated iteratively.

As in MAIAC, in order to get a "first guess" in the iterations, we neglect the multiple bouncing, $\mathbf{R}_{2+} = \mathbf{0}$, and retrieve the RTLS parameters using Eq. (25) as if the light has bounced from the surface once. We then subtract this single bouncing component and the path radiance from the total TOA signal calculated using Eq. (13) and the "first guess" parameters. The difference is the TOA contribution from the second and higher bouncing at BOA – the residual \mathbf{R}_{2+}^m for the next iteration. It is feasible to get an (approximate) analytical expression for \mathbf{R}_{2+}^m in the matrix form and accelerate convergence of the iterations. But because of polarization, it is not easy to follow the MAIAC logic and derive a simple analytical expression for the residual part in vector form. At present, we skip this analytical step and calculate \mathbf{R}_{2+}^m iteratively as summarized below.

First, we represent the TOA signal \mathbf{I}_{TOA} , Eq. (17), in the azimuth space as

$$\mathbf{I}_{TOA} = \mathbf{J} + \mathbf{I}_{TOA}^{SB}(\mathbf{k}) + \mathbf{R}_{2+}(\mathbf{k}), \quad (29)$$

where \mathbf{J} is the path radiance, $\mathbf{I}_{TOA}^{SB}(\mathbf{k})$ is the TOA contribution of the single bouncing (superscript "SB") of light from surface at BOA, Eq. (25). Let i be the index of iteration. The vector of the RTLS parameters at iteration i is $\mathbf{k}_i = [k_L^i \quad k_V^i \quad k_G^i]^T$. For $i = 0$, we neglect the multiple bouncing at BOA, $\mathbf{R}_{2+}(\mathbf{k}_0) = \mathbf{0}$, and get the first guess solution \mathbf{k}_0 from the linear least squares fit of $\mathbf{I}_{TOA}^{SB}(\mathbf{k})$ to the TOA signal. We then simulate the total TOA signal $\mathbf{I}_{TOA}(\mathbf{k}_0)$ and calculate the residual as

$$\mathbf{R}_{2+}(\mathbf{k}_0) = \mathbf{I}_{TOA}(\mathbf{k}_0) - \mathbf{J} - \mathbf{I}_{TOA}^{SB}(\mathbf{k}_0). \quad (30)$$

A minor difference from MAIAC is that Eq. (30) makes no assumptions regarding the residual and accounts for its azimuthal dependence and polarization. We obtain the next guess \mathbf{k}_i from the linear fit of the single bouncing contribution to the TOA signal net the calculated residual and the path radiance

$$\mathbf{I}_{TOA} - \mathbf{J} - \mathbf{R}_{2+}(\mathbf{k}_{i-1}) = \mathbf{I}_{TOA}^{SB}(\mathbf{k}_i). \quad (31)$$

Iterations continue until satisfactory convergence, or a limit on the number of iterations, is achieved. Since all MOM matrices are precomputed, we do not solve the forward RT problem to get the total signal $\mathbf{I}_{TOA}(\mathbf{k})$, but "attach" the RTLS surface to the "known" atmosphere. In the next section we show in Python how it is done for the RTLS and two other surface models.

3. Code snippets to combine atmosphere and surface

Three parts of this section show in Python how to define the surface MOM matrices, connect the matrices to the atmospheric ones, and get the TOA signal. The RT code IPOL computes the atmospheric MOM elements as LUTs. First we consider the Lambertian model (Section 3.1). This surface reflects light evenly over angles (isotropic). Hence its Fourier expansion contains only the azimuthally averaged moment, $m = 0$. The Lambertian model is completely depolarizing; Eq. (1) accurately couples it with atmosphere. Nevertheless, we treat it in the framework of the vector MOM, Eq. (17), filling with zeros those elements of the matrices responsible for polarization. By doing that we aim a methodological benefit for our next examples: unpolarized RTLS land reflection (Section 3.2) and polarized waved ocean surface reflection (Section 3.3).

The RTLS model depends on the relative azimuth ($m > 0$) and the zenith angles. Otherwise, the MOM formalism explained for the isotropic surface remains the same. The RTLS is also depolarizing, however we will show that it does contribute to polarization

on TOA. The last surface model, a polarizing waved ocean surface, is easy to explain based on the previous two. Compared to RTLS, only polarization is new. This explanation-by-analogy results in the longest description for the simplest isotropic model in Section 3.1.

For better explanation, we complement equations with relevant Python code. By using Python, we sacrifice performance but strive for clarity of our scripts. For the same reason, we prefer Numpy's `matmul()`⁹ over `dot()` or `@` to explicitly indicate multiplication of matrices. This choice, maybe controversial, prevents us from using Numba¹⁰ to compile our Python code. For the best performance, though, one should implement numerically intense functions in C or Fortran and, if needed, wrap them in Python.

The three Python scripts for adding the surfaces to atmosphere rely on the low-level dependences - the surface reflection models. Literature explains the surface models in detail; source codes are available from well-developed RT packages. For quick and easy reproducibility of our results, we translated all necessary low-level functions (e.g., Gaussian quadrature, bidirectional and Fresnel ocean reflection, rotation matrix, Fourier expansion, etc.) from our C and Fortran codes [49,53,77] into Python. Appendix A contains a bare minimum of the well-known information to explain our low-level functions that come with the paper.

3.1. Adding isotropic surface reflection

The surface albedo, ρ , is the only parameter for the Lambertian model. The intensity I of radiation reflected from BOA is

$$I_m(\tau_0, \mu_-, \varphi) = \frac{\rho}{\pi} \mu_0 F_0 \exp\left(-\frac{\tau_0}{\mu_0}\right) + \frac{\rho}{\pi} \int_0^{2\pi} \int_0^1 \mu I(\tau_0, \mu, \varphi) d\mu d\varphi, \quad (32)$$

while the Stokes-Q and -U components are absorbed by the surface (recall: we drop the V-component). The second term in the right-hand side of Eq. (32) is the doubled Fourier moment $m = 0$ (see Appendix A; note the scaling factor 2 in Eq. (33))

$$I_m(\tau_0, \mu_-) = \frac{\rho}{\pi} \mu_0 F_0 \exp\left(-\frac{\tau_0}{\mu_0}\right) + 2\rho \int_0^1 \mu I_m(\tau_0, \mu) d\mu. \quad (33)$$

Using the Gaussian quadrature, one integrates over the downwelling radiation numerically. Let N be the number of the Gaussian nodes $\mu_j > 0$ and weights w_j ($j = 1 \dots N$). Eq. (33) transforms to

$$I_m(\tau_0, \mu_-, \varphi) = \frac{\rho}{\pi} \mu_0 F_0 \exp\left(-\frac{\tau_0}{\mu_0}\right) + 2\rho \sum_{j=1}^N w_j \mu_j I_m(\tau_0, \mu_j). \quad (34)$$

Summation in Eq. (34) is the dot product of two vectors: \mathbf{r} and \mathbf{I} , where

$$\mathbf{I} = \begin{bmatrix} I_m(\tau_0, \mu_1) & Q_m(\tau_0, \mu_1) & U_m(\tau_0, \mu_1) \\ \dots & I_m(\tau_0, \mu_N) & Q_m(\tau_0, \mu_N) & U_m(\tau_0, \mu_N) \end{bmatrix}^T \quad (35)$$

is the Stokes vector at the Gaussian directions (T is the transpose operation) and a sparse (row-)vector

$$\mathbf{r} = 2\rho \begin{bmatrix} w_1 \mu_1 & 0 & 0 & w_2 \mu_2 & 0 & 0 & \dots & w_N \mu_N & 0 & 0 \end{bmatrix} \quad (36)$$

accounts for the fact that the Stokes-Q and -U components are not reflected from the Lambertian surface. To account for multiple bouncing of light from the Gaussian directions to the Gaus-

sian directions, we build the following $3N \times 3N$ sparse matrix for Eq. (13)

$$\mathbf{R}_S = \begin{bmatrix} \mathbf{r} \\ 0 \\ 0 \\ 0 \\ 0 \\ \dots \\ \mathbf{r} \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad [\mathbf{R}_S] = 3N \times 3N. \quad (37)$$

Figs. 2 and 3 show the Python function `add_lambert()` that executes the described steps.

Hereinafter, we use bold symbols for arrays; \mathbf{J}_m , \mathbf{J}_p , \mathbf{T}_m , and \mathbf{R}_p are the atmospheric MOM elements from IPOL. Matrices containing the Fourier moments, \mathbf{J}_p , \mathbf{T}_m , and \mathbf{R}_p correspond, respectively, to \mathbf{J}_+^m , \mathbf{T}_-^m , and \mathbf{R}_+^m in Eq. (13). Contrary to these, the TOA path radiance \mathbf{J}_m comes in the azimuth space and already corrected for the single scattering.

Now we turn our attention to the matrix \mathbf{T}^m (in the Fourier space). There are two types of transmittances: direct and diffuse. The direct transmittance accounts only for the exponential attenuation along the line of sight, $E_d = \exp(-\tau/|\mu|)$. The diffuse transmittance, \mathbf{T}^m , accounts for contribution of light scattered from all directions towards a given one. Assuming for example `nmu=3` user-defined directions, `ng1=4` Gaussian nodes per hemisphere, and no polarization, the 3-by-(3 + 4) transmittance matrix would be

$$[\mathbf{E}_d \quad \mathbf{T}^m] = \begin{bmatrix} \exp\left(-\frac{\tau}{|\mu_1|}\right) & 0 & 0 \\ 0 & \exp\left(-\frac{\tau}{|\mu_2|}\right) & 0 \\ 0 & 0 & \exp\left(-\frac{\tau}{|\mu_3|}\right) \end{bmatrix} [\mathbf{T}^m]_{3 \times 4}, \quad (38)$$

and the corresponding vector for the total intensity \mathbf{I} would contain 7 elements.

For a single line of sight, Eq. (38) becomes a row vector. However, as the number of the user-defined directions `nmu` grows and becomes comparable, or even exceeds, the Gaussian order `ng1` (e.g., for building a LUT for a Rayleigh atmosphere), the matrix Eq. (38) becomes sparse. Efficient memory allocation requires to either use special routines for the sparse matrices, or deal with the direct and diffuse transmittances separately.

We chose the second option and consider, for instance, the transmittance of the direct solar beam bounced once from the surface: $\mathbf{T}_0^m \mathbf{S}_0^m$ in Eq. (13). The vector of the direct solar beam reflected from BOA towards all directions comes from the first term in the right-hand side of Eq. (34). Unscattered but attenuated contribution of the surface reflection of the direct solar beam at TOA is

$$\mathbf{E}_d \mathbf{S}_0 = \frac{\rho}{\pi} \mu_0 F_0 \exp\left(-\frac{\tau_0}{\mu_0}\right) \begin{bmatrix} \exp\left(-\frac{\tau_0}{|\mu_1|}\right) \\ 0 \\ 0 \\ \exp\left(-\frac{\tau_0}{|\mu_2|}\right) \\ 0 \\ 0 \\ \dots \\ \exp\left(-\frac{\tau_0}{|\mu_N|}\right) \\ 0 \\ 0 \end{bmatrix}, \quad (39)$$

where the number of elements in \mathbf{S}_0 depends on the number of user-defined directions (e.g., 3 in Eq. (38)). The diffuse contribution

⁹ Hereinafter we use `Courier` for elements of code: variables, functions, file names, paths, executable commands, etc.

¹⁰ <https://numba.pydata.org/numba-doc/dev/reference/numbysupported.html>

```

1. import numpy as np
2. from scipy import linalg as la
3. def add_lambert(srfa, tau0, mu, mu0, raz, mug, wg, Jm, Jp, Tm, Rp):
4.     '''
5.     In:
6.         srfa      f          Lambertian surface albedo
7.         tau0      f          total atmosphere optical thickness
8.         mu        f[nmu]    cos of user-defined zenith angles; mu < 0
9.         mu0       f[nmu0]   cos of solar zenith angles; mu0 > 0
10.        raz       f[naz]    relative azimuths in radians
11.        mug       f[ng1]    nodes of Gaussian quadrature in hemisphere; mug > 0
12.        wg        f[ng1]    weights of Gaussian quadrature
13.        Jm        f[naz, nmu0, nmu3] TOA path radiance at user-defined zeniths
14.        Jp        f[nmu0, ng3] m=0 Fourier moment for BOA path radiance at Gaussian nodes
15.        Tm        f[nmu3, ng3] m=0 Fourier moment for transmittance matrix, Tminus
16.        Rp        f[ng3, ng3] m=0 Fourier moment for reflectance matrix, Rplus
17.     Out:
18.         Itoa      f[naz, nmu0, nmu3] TOA Stokes vector
19.     Note:
20.         ng3 = 3*ng1, nmu3 = 3*nmu - to account for the Stokes-IQU.
21.         A[nz, ny, nx] is a multidimensional array with 'nx' as lead dimension (C-order)
22.     '''

```

Fig. 2. Interface for the Python function that couples an atmosphere with the Lambertian surface. Bold symbols denote arrays. **Jm** comes in the azimuth space (line 13), while **Jp**, **Tm**, and **Rp** contain the Fourier components.

```

def add_lambert(srfa, tau0, mu, mu0, raz, mug, wg, Jm, Jp, Tm, Rp):
...
23.     Itoa = np.zeros_like(Jm) # initialize output array
24. #
25.     pi = np.pi
26.     nmu = len(mu)
27.     nmu0 = len(mu0)
28.     naz = len(raz)
29.     ng1 = len(mug)
30.     ng3 = ng1*3
31. #
32.     rs = np.zeros(ng3)
33.     rs[0:ng3:3] = 2.0*srfa*wg*mug
34.     Rs = np.zeros((ng3, ng3))
35.     for ig in range(ng1):
36.         Rs[3*ig, :] = rs
37.     I_RpRs = np.eye(ng3) - np.matmul(Rp, Rs)

```

Fig. 3. The surface reflection matrices for multiple bouncing: **Rs**, Eq. (37), and $(1 - \mathbf{R}_p^* \mathbf{R}_s)$, Eq. (24). See Eqs. (33) and (36) for the scaling factor 2 in line 33.

is

$$\mathbf{T}_m^m \mathbf{S}_0 = \frac{\rho}{\pi} \mu_0 F_0 \exp\left(-\frac{\tau_0}{\mu_0}\right) \mathbf{T}_m^m \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ \dots \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}. \quad (40)$$

Contrary to Eq. (39), the length of \mathbf{S}_0 in Eq. (40) depends on the number of the Gaussian nodes (e.g., 4 in Eq. (38); we keep the same notation \mathbf{S}_0 in Eqs. (39) and (40) for simplicity). IPOL provides the diffuse transmittance \mathbf{T}_m^m on output.

The most sophisticated term in Eq. (13) simulates multiple bouncing of light between atmosphere and surface. Initially, the surface is irradiated by the downwelling path radiance \mathbf{J}_+^m (m -th moment, including the one for single scattering) and the direct solar beam first reflected from surface (up), and then reflected from BOA back to the surface (down), $\mathbf{R}_+^m \mathbf{S}_0^m$. Geometrical progression, in matrix form, simulates multiple bouncing of these two compo-

nents

$$\mathbf{I}_B^+ = (\mathbf{I} - \mathbf{R}_+^m \mathbf{R}_S^m)^{-1} (\mathbf{J}_+^m + \mathbf{R}_+^m \mathbf{S}_0^m). \quad (41)$$

Since Eq. (41) uses downwelling diffuse radiation, the user-defined upwelling zeniths are not involved at this step – all matrices are square. Eq. (41) operates only with $ng1$ Gaussian nodes ($ng3=3*ng1$ to account for IQU). The elements in Eq. (41) have the following size: for a given solar zenith, \mathbf{J}_+^m and \mathbf{S}_0^m are vectors of $ng3$ elements. The latter is not to be confused with \mathbf{S}_0 of $nmu3$ elements in Eq. (39). \mathbf{R}_+^m , \mathbf{R}_S^m are $ng3$ -by- $ng3$ matrices.

The downwelling radiation, Eq. (41), is reflected from surface, $\mathbf{R}_S^m \mathbf{I}_B^+$, and transmitted from BOA to TOA: $\mathbf{T}_m^m \mathbf{R}_S^m \mathbf{I}_B^+$. Like with the transmittance of the reflected direct solar beam, Eqs. (39)–(41), it is efficient to consider the direct (exponential) and diffuse transmittances separately. Fig. 4 shows the Python code for these steps (note explicit loop over the solar zenith cosine and see Section 5 for relevant discussion).

Finally, all components are combined at TOA – Fig. 5.

Note again that implementation of the conventional approach, Eq. (1), is easier and computationally more efficient (e.g., one should skip the surface contribution to polarization – there is none). If MOM is used, one should use $ng1$ -by- $ng1$ matrices and $ng1$ -long vectors for the total intensity only. However, most of

```

def add_lambert(srfa, tau0, mu, mu0, raz, mug, wg, Jm, Jp, Tm, Rp):
...
38. S0 = np.zeros(ng3)
39. for imu0 in range(nmu0):
40.     cs0 = mu0[imu0]
41.     for ig in range(ng1):
42.         S0[3*ig] = cs0*srfa*np.exp(-tau0/cs0)/pi # bouncing of direct solar beam
43.         Im0 = np.matmul(Tm, S0) # diffuse transmittance of S0 from BOA to TOA
44.         Ip = Jp[imu0, :] + np.matmul(Rp, S0) # diffuse light falling on surface
45.         Ipb = la.solve(I_RpRs, Ip) # multiple bouncing of diffuse light
46.         Ib = np.matmul(Rs, Ipb) # bouncing light is reflected ...
47.         Im = np.matmul(Tm, Ib) # ... and transmitted diffusely from BOA to TOA

```

Fig. 4. Computation of the direct (exponential) and diffuse transmittances of light from BOA to TOA.

```

def add_lambert(srfa, tau0, mu, mu0, raz, mug, wg, Jm, Jp, Tm, Rp):
...
23. Itoa = np.zeros_like(Jm) # initialize output array
...
39. for imu0 in range(nmu0):
...
48.     for iaz in range(naz):
49.         for imu in range(nmu):
50.             Itoa[iaz, imu0, 3*imu] = Jm[iaz, imu0, 3*imu] + \
51.                 Im0[3*imu] + Im[3*imu] + \
52.                 2.0*srfa*np.dot(mug*wg, Ipb[0:ng3:3])*np.exp(tau0/mu[imu]) + \
53.                 cs0*srfa*np.exp(-tau0/cs0)*np.exp(tau0/mu[imu])/pi # mu < 0
54.             Itoa[iaz, imu0, 3*imu+1] = Jm[iaz, imu0, 3*imu+1]
55.             Itoa[iaz, imu0, 3*imu+2] = Jm[iaz, imu0, 3*imu+2]
return Itoa

```

Fig. 5. Summation of all components contributing to the TOA signal. Lines 52 and 53 contain direct transmittances of the BOA signal caused by the single reflection of the direct solar beam (line 53) and multiple bouncing of light at BOA (line 52). The Lambertian surface contributes nothing to the Stokes-Q (line 54) and -U (line 55) components – hence these TOA signals are essentially path radiances. See Eqs. (33)-(34) for the scaling factor 2 (line 52).

steps in `add_lambert()` for $m = 0$ are repeated for the directional reflectance that require $m > 0$.

3.2. Adding directional land reflection

The RTLS surface model [80] includes three components: the Lambertian (L), volumetric (V) [81], and geometric-optics (G) [82] reflection

$$\rho(\mu, \mu', \varphi - \varphi') = k_L + k_V f_V(\mu, \mu', \varphi - \varphi') + k_G f_G(\mu, \mu', \varphi - \varphi'). \quad (42)$$

In Eq. (42), k -s and $f()$ -s are the kernel weights and the kernel functions, respectively; μ, μ' are the cosines of the zenith angles of the reflected and incident beams, respectively; $\varphi - \varphi'$ is the relative azimuth (for the solar beam $\varphi' = \varphi_0 = 0^\circ$). We adhere to the abovementioned sign convention: $\mu < 0, \mu' > 0$, the surface hot-spot is at $\varphi - \varphi' = 180^\circ$ (“backscattering”). Literature [71: Sec. 2.1.1; 77: Sec. 2B; 81] gives explicit expressions for all elements in Eq. (42). The RTLS scripts are also available as part of other open-source RT codes; our Python implementation of the RTLS model comes with this paper (see Section 5.2).

The azimuthal dependence of the surface reflection changes Eq. (25) for the upwelling light at BOA

$$I(\tau_0, \mu_-, \varphi) = \frac{1}{\pi} \mu_0 F_0 \rho(\mu, \mu_0, \varphi) \exp\left(-\frac{\tau_0}{\mu_0}\right) + \frac{1}{\pi} \int_0^{2\pi} \int_0^1 \mu' \rho(\mu, \mu', \varphi - \varphi') I(\tau_0, \mu, \varphi') d\mu' d\varphi'. \quad (43)$$

Coupling of atmosphere and the azimuthally dependent RTLS requires expansion of the surface BRDF in the Fourier series

$$\rho(\mu, \mu', \varphi) = \sum_{m=0}^M (2 - \delta_{0m}) \rho_m(\mu, \mu') \cos(m\varphi). \quad (44)$$

```

1. import numpy as np
2. from scipy import linalg as la
3. from def_gauszw import gauszw
4. from def_rtls import rtls
5. from def_rtlsm import rtlsm

```

Fig. 6. Modules used by the Python function `add_rtls()`.

One also needs moments with $m > 0$ for the downwelling path radiance J_+^m , and the atmospheric reflection R_+^m and transmission T_+^m matrices. Eqs. (9) and (44), in which the Fourier index m runs independently, and the orthogonality of the cosine function, yields in Eq. (43)

$$I(\tau_0, \mu_-, \varphi) = \frac{1}{\pi} \mu_0 F_0 \rho(\mu, \mu_0) \exp\left(-\frac{\tau_0}{\mu_0}\right) + 2 \int_0^1 \mu' \rho_m(\mu, \mu') I_m(\tau_0, \mu') d\mu'. \quad (45)$$

Note the two scaling factors in the right side of Eq. (45): $1/\pi$ in reflection of the direct solar beam, Eqs. (11), and 2 in reflection of the diffuse downwelling radiation $I_m(\tau_0, \mu')$. Depending on definition of the extraterrestrial solar irradiance F_0 , the reader may find different factors in literature (e.g., see Eq. (6.1b) in [18, p. 206]). So, it is important to make sure that F_0 is the same for all terms in Eq. (45). Recall, in this paper we assume $F_0 = 1$.

Numerical calculation of the expansion moments, $\rho_m(\mu, \mu')$, using the Gaussian quadrature rule is known: see Appendix A and [34] for the full text of `gauszw()`. Fig. 6 shows modules for the `add_rtls()` function.

Input for adding the Lambertian and RTLS models is similar except for RTLS uses three kernel weights as parameters: k_L, k_V, k_G . It also needs the accuracy parameters for the Fourier expansion, Eq. (22): the number of the Fourier moments nm (not exceeding the one for the downwelling path radiance) and the or-

```

6. def add_rtls(nm, nga, kl, kv, kg, tau0, mu, mu0, raz, mug, Jm, Jp, Tm, Rp):
7.     '''
8.     In:
9.         nm          i          number of Fourier moments
10.        nga         i          order of Gaussian quadrature in Fourier expansion
11.        kl, kv, kg   f          RTLS kernel weights
12.        tau0         f          total atmosphere optical thickness
13.        mu           f[nmu]     cos of user-defined zenith angles; mu < 0
14.        mu0          f[nmu0]    cos of solar zenith angles; mu0 > 0
15.        raz          f[naz]     relative azimuths in radians
16.        mug          f[ngl]     nodes of Gaussian quadrature in hemisphere; mug > 0
17.        wg           f[ngl]     weights of Gaussian quadrature
18.        Jm           f[naz, nmu0, nmu3] TOA path radiance at user-defined zeniths
19.        Jp           f[nm, nmu0, ng3] Fourier moments for BOA path radiance at G-nodes
20.        Tm           f[nm, nmu3, ng3] Fourier moments for transmittance matrix, T_minus
21.        Rp           f[nm, ng3, ng3] Fourier moments for reflectance matrix, R_plus
22.     Out:
23.         Itoa        f[naz, nmu0, nmu3] Stokes vector at TOA
24.     Note:
25.         ng3=3*ngl, nmu3=3*nmu - to account for IQU components of the Stokes vector.
26.         A[nz, ny, nx] is a multidimensional array with 'nx' as a lead dimension
27.     '''

```

Fig. 7. Interface for the Python function `add_rtls()`.

der of the Gaussian quadrature for the numerical integration over azimuth, nga (number of the gaussian azimuths) – see Fig. 7. We now proceed to construction of the necessary surface matrices.

Like in the case of the Lambertian surface, the RTLS model is depolarizing – hence \mathbf{R}_S is sparse. Unlike the Lambertian case, the RTLS depends on m :

$$\mathbf{r}_m(\mu < 0) = 2 \begin{bmatrix} w_1 \mu'_1 \rho_m(\mu, \mu'_1) & 0 & 0 & w_2 \mu'_2 \rho_m(\mu, \mu'_2) \\ 0 & 0 & \dots & w_N \mu'_N \rho_m(\mu, \mu'_N) & 0 & 0 \end{bmatrix}. \quad (46)$$

In Eq. (46), all primed values are nodes of the Gaussian quadrature; refer to Eq. (45) for the factor of 2. The surface reflection matrix is responsible for multiple bouncing of light at BOA from the Gaussian nodes to the Gaussian nodes. As before, its size is $3N \times 3N$ ($ng3 \times ng3$ in terms of the code variables):

$$\mathbf{R}_S^m = \begin{bmatrix} \mathbf{r}_m(\mu'_1) \\ 0 \\ 0 \\ \mathbf{r}_m(\mu'_2) \\ 0 \\ 0 \\ \dots \\ \mathbf{r}_m(\mu'_N) \\ 0 \\ 0 \end{bmatrix}, \quad [\mathbf{R}_S^m] = 3N \times 3N. \quad (47)$$

Similar matrix that simulates reflection of light from the Gaussian nodes to K user-defined directions is

$$\mathbf{r}_S^m = \begin{bmatrix} \mathbf{r}_m(\mu_1) \\ 0 \\ 0 \\ \mathbf{r}_m(\mu_2) \\ 0 \\ 0 \\ \dots \\ \mathbf{r}_m(\mu_K) \\ 0 \\ 0 \end{bmatrix}, \quad [\mathbf{r}_S^m] = 3K \times 3N. \quad (48)$$

In terms of the code variables, the size of the matrix in Eq. (48) is $nmu3 \times ng3$. Fig. 8 shows the corresponding code, up to computation of the multiple bouncing matrix $(\mathbf{1} - \mathbf{R}_+^m \mathbf{R}_S^m)^{-1}$. Line 42 indicates that the code is executed for each Fourier index m independently (one can run the loop over m in parallel) and that the atmospheric MOM-matrices depend on m (lines 43–45).

Reflection of the direct solar beam from the RTLS surface, lines 58–59 in Fig. 9, is simulated precisely as in the isotropic case, except for one calls `rtls_m()` instead of using the isotropic surface albedo `srfa` as a single parameter. Simulation of the multiple bouncing of light at TOA and the diffuse transmittance of that from BOA to TOA is also identical in the Lambertian and RTLS models. We refer our reader to Fig. 3 for explanation of the corresponding element of code. Lines 66–74 in Fig. 8 execute the azimuthal summation of the Fourier components. Namely, the direct transmittance of the downwelling path radiance (line 71) contributes to the I -component only. However, the RTLS model contributes to the Stokes- Q and $-U$ via the diffusely transmitted direct solar beam reflectance (see line 60 for `Im0`) and the light I_b bounced multiple times at BOA (see line 64 for `Im`).

After the Fourier summation is complete, one combines the diffuse surface contribution at TOA with the path radiance. Finally, the direct contribution of the solar beam reflected from the surface just once – single bouncing component – is added. This single bouncing component is not expanded in the Fourier series: Fig. 10.

To conclude this subsection, we note again the peculiarity in adding the unpolarized but directional surface reflection. Contrary to the Lambertian surface, the RTLS surface does contribute to polarization on TOA. The surface reflected intensity possesses some angular dependence, which in its turn generates Q and U components while scattering in atmosphere, like the unpolarized solar beam generates polarization in the scattered radiation. In Section 4 we quantify the level of the contribution. In addition to that, `add_rtls()` is computationally more demanding as compared to `add_lambert()`. We refer to Section 5.1 for a brief discussion of “bad” ideas that we intentionally use in Figs. 8–10 for clarity. Section 5.1 also offers ways for improvement using as an example our last considered model – polarized ocean.

3.3. Adding polarized reflection from ocean surface

Reflection from the ocean surface is significantly polarized at the glint and maintains noticeable polarization beyond it. We refer to [83] for the basics and to [84] for detailed mathematical description of the polarized light scattering in planetary atmospheres. Here we bring only a few facts necessary for the subsequent discussion (recall, we skip the Stokes- V component):


```

def add_rtls(nm, nga, kl, kv, kg, tau0, mu, mu0, raz, mug, wg, Jm, Jp, Tm, Rp):
... lines 28:33 define: pi, ng1, ng3, nmu, nmu3, naz - see add_lambert()
34. Itoa = np.zeros_like(Jm)
35. Itoa_srf = np.zeros_like(Jm)
36. rs = np.zeros((nmu, ng1))
37. Rs = np.zeros((ng3, ng3))
38. S0 = np.zeros(ng3)
39. zaz, waz = gauszw(0.0, 2.0*pi, nga)
40. Tdir = np.exp(tau0/mu) # direct transmittance from BOA to TOA, mu < 0
41. scale_m12 = 1.0 # (2 - Kronecker_0m)
42. for im in range(nm):
43.     Jp_im = Jp[im, :, :]
44.     Rp_im = Rp[im, :, :]
45.     Tm_im = Tm[im, :, :]
46.     for imu in range(nmu):
47.         for jg in range(ng1):
48.             rs[imu, jg] = 2.0*wg[jg]*mug[jg]* \
49.                 rtls(im, zaz, waz, nga, kl, kv, kg, mug[jg], mu[imu])
50.         for ig in range(ng1):
51.             for jg in range(ng1):
52.                 Rs[3*ig, 3*jg] = 2.0*wg[jg]*mug[jg]* \
53.                     rtls(im, zaz, waz, nga, kl, kv, kg, mug[jg], -mug[ig])
54.     I_RpRs = np.eye(ng3) - np.matmul(Rp_im, Rs)

```

Fig. 8. Begin of the im -loop over the Fourier moments m and computation of the surface reflection matrices. See Eq. (46) for the scaling factor 2 in lines 48 and 52. The Python function `rtls()` computes the m -th Fourier moment for RTLS – see Fig. 16 for the call graph and Appendix A.

```

def add_rtls(nm, nga, kl, kv, kg, tau0, mu, mu0, raz, mug, wg, Jm, Jp, Tm, Rp):
41. scale_m12 = 1.0 # (2 - Kronecker_0m) = 1 for m = 0
42. for im in range(nm):
...
55.     for imu0 in range(nmu0):
56.         cs0 = mu0[imu0]
57.         for ig in range(ng1):
58.             S0[3*ig] = cs0*np.exp(-tau0/cs0)* \
59.                 rtls(im, zaz, waz, nga, kl, kv, kg, cs0, -mug[ig])/pi
60.         Im0 = np.matmul(Tm_im, S0) # diffuse transmittance of S0 from BOA to TOA
61.         Ip = Jp_im[imu0, :] + np.matmul(Rp_im, S0) # downwelling light at BOA
62.         Ipb = la.solve(I_RpRs, Ip) # multiple bouncing of light at BOA
63.         Ib = np.matmul(Rs, Ipb) # bouncing light is reflected ...
64.         Im = np.matmul(Tm_im, Ib) # ... and transmitted diffusely from BOA to TOA
65.         # Accumulate Fourier series for each azimuth
66.         for iaz in range(naz):
67.             cmaz = np.cos(im*raz[iaz])*scale_m12 # for I&Q-components
68.             smaz = np.sin(im*raz[iaz])*scale_m12 # for U-components
69.             for imu in range(nmu):
70.                 Itoa_srf[iaz, imu0, 3*imu] += (Im0[3*imu] + Im[3*imu] + \
71.                     np.dot(rs[imu, :], Ipb[0:ng3:3])*Tdir[imu])*cmaz
72.                 Itoa_srf[iaz, imu0, 3*imu+1] += (Im0[3*imu+1] + Im[3*imu+1])*cmaz
73.                 Itoa_srf[iaz, imu0, 3*imu+2] += (Im0[3*imu+2] + Im[3*imu+2])*smaz
74.         scale_m12 = 2.0 # (2 - Kronecker_0m) = 2 for m > 0; end of im-loop

```

Fig. 9. Summation of the Fourier series for each relative azimuth (lines 66–73) and computation of the TOA surface contribution, `Itoa_srf`. Direct transmittance of the downwelling path radiance (line 71) contributes to the I -component only. However, RTLS contributes to the Stokes- Q and $-U$ via diffusely transmitted direct solar beam reflectance (see line 60 for `Im0`) and the diffuse light bounced multiple times `Ib` (see line 64 for `Im`).

- (a) A 3-by-3 Mueller matrix \mathbf{M} describes reflection of the Stokes vector $\mathbf{S} = [I, Q, U]^T$ as $\mathbf{S} = \mathbf{M}\mathbf{S}_0$. Reflection of the direct unpolarized Solar beam, $\mathbf{S}_0 = [I, 0, 0]^T$ requires only the first column of the Mueller matrix \mathbf{M} .
- (b) The Stokes- Q and $-U$ components are responsible for the degree of linear polarization (DoLP). Their sign and magnitude depend on selection of the reference plane (system of coordinates). So does the Mueller matrix. The vRT equation uses the meridional plane defined by the local zenith and the line of sight as the reference one. However, the surface Muller matrix is defined w.r.t. a plane that contains normal to a reflecting facet. In case of a rough ocean surface, the reflecting

- facets are randomly oriented, hence rotation of the reference plane is required to simulate the reflection.
- (c) Rotation of the reference plane at an angle α is described by the rotation matrix, $\mathbf{L}(\alpha)$ (see Eq. (A2) in the Appendix). Reflection of the polarized beam requires two rotations: from the vRT equation system of coordinates to the surface one before, and backward after the reflection has occurred. The ocean surface is not flat, hence directions of irradiation and observation, in general, do not belong to the same plane. Likewise, the meridional planes before and after the reflection from the ruffled ocean surface do not coincide. Thus, one needs two rotation angles, α_1 and α_2 .

```

def add_rtls(nm, nga, kl, kv, kg, tau0, mu, mu0, raz, mug, wg, Jm, Jp, Tm, Rp):
...
42.     for im in range(nm):
...
74.         scale_m12 = 2.0 # (2 - Kronecker_m0)=2 for m > 0; end of im-loop
75.         # Combine TOA contribution (including direct surface)
76.         Itoa = Jm + Itoa_srf
77.         for iaz in range(naz):
78.             for imu0 in range(nmu0):
79.                 cs0 = mu0[imu0]
80.                 Tsol = np.exp(-tau0/cs0) # TOA->BOA solar beam direct transmittance
81.                 for imu in range(nmu):
82.                     Itoa[iaz, imu0, 3*imu] += cs0*Tsol*Tdir[imu]* \
83.                                             rtls(kl, kv, kg, cs0, mu[imu], raz[iaz])/pi
84.     return Itoa

```

Fig. 10. The path radiance J_m is combined with the diffuse surface contribution I_{toa_srf} (line 76) and the direct contribution from single bouncing of the solar beam (lines 82–83); see Eq. (45) for the factor of $1/\pi$ in line 83.

```

1. import numpy as np
2. from scipy import linalg as la
3. from def_gauszw import gauszw
4. from def_oceanm import oceanm
5. from def_oceanm0 import oceanm0
6. from def_wavysrf import wavysrf
7. from def_fresnel0 import fresnel0
8. def add_ocean(nm, nga, shad, sgm2, refre, tau0, \
                mu, mu0, raz, mug, wg, Jm, Jp, Tm, Rp):

```

Fig. 11. Dependences and interface of the Python function `add_ocean()`. Except for the surface parameters, the interface is like for `add_rtls()`.

Therefore, one utilizes the following sequence of the matrix multiplications to simulate reflection of the polarized light from a wind-waved ocean surface

$$\rho(\mu, \mu', \varphi - \varphi') = S(\mu, \mu', w)P(\mu, \mu', \varphi - \varphi', w)L(\pi - \alpha_2) \mathbf{F}(\mu, \mu', \varphi - \varphi', n)L(-\alpha_1). \quad (49)$$

In Eq. (49), $S()$ and $P()$ are scalar functions for mutual shadowing of waves, and probability of the wave slope distribution, respectively. Beside the solar/view geometry, they depend on the windspeed w (usually defined in meters per second, m/s, at 10 m above the ocean surface). In this paper, we consider numerical examples that ignore the wind direction. However, specifying the wind azimuth does not prevent the use of the MOM technique. \mathbf{F} is a 3-by-3 Fresnel reflection matrix that depends on the refractive index of water n . Like the RTLS model, many authors describe Eq. (49) in detail [77: Sec. 2C; 85,86]. Well documented, reliable source codes [87] are available as stand-alone programs `ocean.shadow.f` and `ocean.phase.f`¹¹, and as part of other publicly available RT source codes [41,53,60]. For independent reproducibility, we also provide our implementation of Eq. (49) in Python with this paper, but do not repeat others in discussion of relevant details.

The ocean surface reflects the Fourier components with $m > 0$ like the directional land model. Contrary to RTLS, all 3×3 blocks in the surface matrices are, in general, non-zero. In the scripts below, we only provide those lines of the `add_ocean()` function that differ from the corresponding lines in `add_rtls()`. We inherit the line numbers from `add_rtls()` where possible; unique `add_ocean()` lines are left unnumbered.

Fig. 11 shows dependences and interface for `add_ocean()`: `oceanm()` computes the m -th Fourier moment for the ocean Mueller matrix, `oceanm0()` does the same for the direct solar beam reflection (first column of the Mueller matrix), `wavysrf()`

simulates waves and, optionally, shadows, `fresnel0()` is the first column of the rotated Fresnel reflection matrix.

Compared to `add_rtls()`, the three new surface parameters are: `shad` to turn the mutual shadowing on or off (below we reproduce both published scenarios), `sgm2` is the squared standard deviation (variance) of the waves slope distribution (assumed Gaussian), `refre` is the refractive index of water. The slope distribution depends on the wind speed. However, we prefer `sgm2` as input because different authors use slightly different dependence of `sgm2` on the wind speed.

All elements of the surface reflection matrices are now filled with numbers – see Fig. 12. Note, we fill up `rs` and `Rs` nonconsecutively. This is easy to read and understand, but inefficient. Moreover, `oceanm()` in lines 50 and 54 is the most time-consuming part of `add_ocean()`. Same for `ocnm0()` for simulation of the direct solar beam reflection at BOA in the Fourier space (Fig. 13), if the number of the solar angles `nmu0` is large. We refer our reader to Section 5.1 for possible solution.

Accumulation of the Fourier series, Fig. 14, and adding an exact (without expansion in series) contribution of the direct solar beam reflection, Fig. 15, are also like those in `add_rtls()`.

The ocean surface model is the most general case in a sense that it is directional and polarized. However, we do not even discuss the MOM part in this subsection. Once the MOM matrices are created appropriately, the MOM formalism works alike for any surface model. In the next section, we test `add_lambert()`, `add_rtls()`, and `add_ocean()` against published benchmarks.

4. Validation of the Python functions

In this section, we solve the forward problem: surface is attached to atmosphere. Using IPOL, we precomputed MOM elements for the atmosphere. A few simple scripts read the IPOL output, compute the TOA signal using functions from Section 3, and test the result vs. published benchmarks. We report runtime for

¹¹ <https://www.giss.nasa.gov/staff/mmishchenko/brf/>

```

def add_ocean(nm, nga, shad, sgm2, refre, tau0, \
              mu, mu0, raz, mug, wg, Jm, Jp, Tm, Rp):
...
38. rs = np.zeros((nmu3, ng3))
...
44. for im in range(nm):
...
48.     for imu in range(nmu):
49.         for jg in range(ng1):
50.             ocnm = oceanm(im, shad, sgm2, refre, zaz, waz, mug[jg], mu[imu])
51.             rs[3*imu:3*imu+3, 3*jg:3*jg+3] = 2.0*wg[jg]*mug[jg]*ocnm
52.         for ig in range(ng1):
53.             for jg in range(ng1):
54.                 ocnm = oceanm(im, shad, sgm2, refre, zaz, waz, mug[jg], -mug[ig])
55.                 Rs[3*ig:3*ig+3, 3*jg:3*jg+3] = 2.0*wg[jg]*mug[jg]*ocnm

```

Fig. 12. The surface matrices for reflection of light from the Gaussian directions to the user-defined ones (line 51) and for recurrence bouncing from the Gaussian directions to the same (line 55).

```

def add_ocean(nm, nga, shad, sgm2, refre, tau0, \
              mu, mu0, raz, mug, wg, Jm, Jp, Tm, Rp):
...
44. for im in range(nm):
...
57.     for imu0 in range(nmu0):
58.         cs0 = mu0[imu0]
59.         for ig in range(ng1):
60.             ocnm0 = oceanm0(im, shad, sgm2, refre, zaz, waz, cs0, -mug[jg])
61.             S0[3*ig:3*ig+3] = cs0*np.exp(-tau0/cs0)*ocnm0/pi

```

Fig. 13. Reflection of the direct Solar beam at BOA; *ocnm0* is a vector of 3 elements (Fourier components for the *IQU*-triplet).

```

def add_ocean(nm, nga, shad, sgm2, refre, tau0, \
              mu, mu0, raz, mug, wg, Jm, Jp, Tm, Rp):
...
44. for im in range(nm):
...
57.     for imu0 in range(nmu0):
...
        Ibl = np.matmul(rs, Ipb) # bouncing from Gauss to user nodes
68.     for iaz in range(naz):
69.         cmaz = np.cos(im*raz[iaz])*scale_m12
70.         smaz = np.sin(im*raz[iaz])*scale_m12
71.         for imu in range(nmu):
72.             Itoa_srf[iaz, imu0, 3*imu] += (Im0[3*imu] + Im[3*imu] + \
73.                                             Ibl[3*imu]*Tdir[imu])*cmaz
74.             Itoa_srf[iaz, imu0, 3*imu+1] += (Im0[3*imu+1] + Im[3*imu+1] + \
75.                                             Ibl[3*imu+1]*Tdir[imu])*cmaz
75.             Itoa_srf[iaz, imu0, 3*imu+2] += (Im0[3*imu+2] + Im[3*imu+2] + \
76.                                             Ibl[3*imu+2]*Tdir[imu])*smaz

```

Fig. 14. Accumulation of the Fourier series for the Stokes-*I*, -*Q*, and -*U* components. The *IQU*-triplet is located in *Itoa_srf* sequentially.

```

def add_ocean(nm, nga, shad, sgm2, refre, tau0, \
              mu, mu0, raz, mug, wg, Jm, Jp, Tm, Rp):
...
77. # Combine TOA contribution (including direct surface)
78. Itoa = Jm + Itoa_srf
79. for iaz in range(naz):
80.     for imu0 in range(nmu0):
81.         cs0 = mu0[imu0]
82.         Tsol = np.exp(-tau0/cs0)
83.         for imu in range(nmu):
84.             wsrfsrf(shad, sgm2, cs0, mu[imu], raz[iaz])
85.             ocn0 = wsrfsrf*fresnel0(refre, cs0, mu[imu], raz[iaz])
86.             Itoa[iaz, imu0, 3*imu:3*imu+3] += cs0*Tsol*Tdir[imu]*ocn0/pi
87. return Itoa

```

Fig. 15. Adding the path radiance, the diffuse (line 78) and direct (line 86) surface contributions at TOA.

Table 1

Validation of the surface-adding functions: input, maximum (max) and average (avr) deviations from benchmarks, and runtime¹⁵ in seconds (sec.) for the Python and C versions of the functions. For reference, the bottom line shows runtime for the vRT code IPOL (in C).

Parameter	Atmosphere Case 1			Case 2	Case 3
The case short description:	Homogeneous molecular (Rayleigh) layer with total optical thickness $\tau_0 = 0.1$ and atmospheric spherical albedo $C_0 \approx 0.084$			Rayleigh, $\tau_0 = 0.6301$, $C_0 \approx 0.343$	2 layers with aerosol, $\tau_0 = 0.6$, $C_0 \approx 0.201$
Path radiance accuracy:	ng1 = 8, nm = 3			ng1 = 24, nm = 16	
Surface:	Lambertian $\rho = 0.3$	RTLS: 0.33 0.053, 0.066	Ocean 2 m/s with shadows	Ocean 7 m/s no shadows	Lambertian $\rho = 0.1$
$dI, \%$ max:	0.02	0.06	0.03	0.00	0.00
$dI, \%$ avr:	0.01	0.04	0.02	0.00	0.00
$dP, \%$ max:	0.04	0.03	0.02	0.00	0.12 ¹⁶
$dP, \%$ avr:	0.00	0.00	0.01	0.00	0.01
runtime (sec.) for surface in Py	0.003	0.03	5	3	0.002
runtime (sec.) for surface in C	0.000	0.01	0.04	0.03	0.000
runtime (sec.) for IPOL in C			0.006		0.5

¹⁵ DELL Latitude E6520 laptop on Intel i7-2720QM CPU, 2.2 GHz, 8GB DD3 SDRAM (1333 MHz), Windows 10 64 bit; Py-runtime: Python 3.7.6 integrated with Anaconda 3/Spyder 4 IDE; C-runtime: Microsoft Visual Studio with C/C++ compiler and Intel MKL library.

¹⁶ de Haan et al. [89] use the fixed format which results in lower relative accuracy at points with small DoLP.

the code written in Python and C/C++, and for the vRT solver for reference.

We consider 5 numerical examples for calculation of the TOA intensity I and the degree of linear polarization P for 3 different atmospheric cases and 3 different types of the surface reflectance. In Case 1, we keep the same atmosphere and the solar/view geometry and change only the surface model: Lambertian (no angular dependence), RTLS (with angular dependence, no polarization), and Ocean (directional, polarized). The atmospheric Case 1 alone and the three surfaces are sufficient to illustrate how to use output from our vRT code IPOL, and the MOM formalism in general, in order to attach the surface to a given atmosphere. However, we add two more atmospheric cases – one with a thicker atmosphere (Rayleigh at a shorter wavelength), and another one with 2 optically different layers.

Case 1 defines the atmosphere and the solar/view geometry following the IPRT¹² Cases A2 (for the Lambertian surface) and A6 (for the ocean surface) [50]. Namely, we consider a pure Rayleigh scattering atmosphere of the total optical thickness $\tau_0 = 0.1$ ($\lambda \approx 550$ nm [88]) and the depolarization factor 0.03 (dry air). The atmospheric spherical albedo is $C_0 \approx 0.084$. We combine SZA = 50° from the IPRT Case A2 and SZA = 45° from the IPRT Case A6 in one atmospheric scenario. Instead of the IPRT's relative azimuth grid from 0° to 180° with step 5° (37 values total) we consider only 5 values: from 0° to 180° step 45°. This keeps our benchmark tables concise and easier to analyze visually. We also limit ourselves to the signal at TOA with the VZA ranging from 0° (nadir) to 80° (approaches horizon), 5° step as in IPRT. The different number of SZAs (2), VZAs (17), and azimuths (5) help illustrate the shape of arrays and avoid confusion with dimensions at coding.

For the Lambertian and the ocean surfaces, accurate numbers are published as the IPRT Case A2 (SZA=50° only) and IPRT Case A6 (SZA=45° only). In addition to those, we couple the same atmosphere and the RTLS surface with the following parameters: $k_l=0.33$, $k_v=0.053$, $k_g=0.066$ [53, p. 309]. Numbers for the RTLS case are not readily available from literature. We refer our reader to a text file provided with the paper: `benchmark_case1b.txt` (see Section 5.2 for the package description).

Our Case 2 atmosphere coincides with the Atmosphere-Ocean System model I (AOS-I) [79]. The AOS-I also considers a molecu-

lar (Rayleigh) atmosphere, but at different wavelength $\lambda \approx 350$ nm ($\tau_0 = 0.6301$ – see Table 5a in their paper), the atmospheric spherical albedo is $C_0 \approx 0.343$, different windspeed (7 m/s vs. 2 m/s in the IPRT Case A6), and slightly different refractive index of water (1.34 vs. 1.33 in the IPRT Case A6). Mutual shadowing of the waves is ignored in the AOS-I [79], but considered in the IPRT Case A6 [50]; foam is ignored in both cases, the ocean body is assumed completely absorbing.

Finally, our atmospheric Case 3 deals with a 2-layer atmosphere over the Lambertian surface. Accurate numbers for this scenario are available in [89]. In this case, a molecular scattering (Rayleigh) layer with $\tau_R = 0.1$ rests on top of the Deirmendjan's Water-Haze L [90] aerosol layer with $\tau_A = 0.4$ at $\lambda \approx 700$ nm (see [91: p.243] for the phase matrix expansion coefficients) mixed with the Rayleigh's $\tau_R = 0.1$. The total atmosphere thickness is $\tau_0 = 0.6$ and the atmospheric spherical albedo is $C_0 \approx 0.201$. Again, we limit ourselves to the TOA values but added azimuths 180° and 150° to 0° and 30° reported in [89]. This case is intended to emphasize that the atmosphere-surface coupling formalism works in the same way for any type of the atmospheric composition.

Table 1 summarizes the input parameters, output errors, and the runtime. In the table, ng1 and nm are the number of the Gaussian nodes per hemisphere and the Fourier moments, respectively, that IPOL used to precompute the path radiance and the MOM matrices. The same ng1 are used for numerical integration over the zenith in Eqs. (25) and (43); the Lambertian surface requires nm=1 ($m = 0$); for the RTLS and the ocean we used the same nm as for the path radiance. The higher Fourier moments, generated by the surface, contribute to the TOA signal without scattering. The maximum (max) and averaged (avr) over all solar/view geometries relative deviation for the total intensity and DoLP, dI and dP respectively, are shown in%. dP is computed as absolute difference of the DoLPs scaled by 100%. We used accurate benchmarks, simple scenarios (compared e.g. to cirrus clouds), and sufficient ng1 and nm – not surprising that all errors are small.

A few quick comments should be made here regarding the presented errors. For the RTLS model, the importance of bouncing of the path radiance from the surface at least once has been discussed in literature (see Section 2). Neglect of polarization generated by re-scattering of the surface reflected intensity (Fig. 9: nullify lines 72 and 73) gives up to 1.3% (0.3% on average) error in the DoLP at TOA (in our scenario). This error is at the level of the

¹² <https://www.meteo.physik.uni-muenchen.de/~iprt/>

Table 2

Maximum (max) and average (avr) relative errors (in %) resulted from truncation, Eq. (24), for two ocean scenarios; $n = 1$ means single bouncing of the direct and diffuse light from surface.

Number of terms in Eq. (24), n :	Ocean (2 m/s, with shadows) under the Rayleigh atmosphere with $\tau_0 = 0.1$				Ocean (7 m/s, no shadows) under the Rayleigh atmosphere with $\tau_0 = 0.6301$			
	dI , %		dP , %		dI , %		dP , %	
	max.	avr.	max.	avr.	max.	avr.	max.	avr.
1	0.34	0.09	0.16	0.04	0.20	0.13	0.10	0.04
2	0.03	0.02	0.02	0.01	0.01	0.01	0.00	0.00
3	0.03	0.02	0.02	0.01	0.00	0.00	0.00	0.00

desired measurement precision of the state-of-the art polarimeters (see Appendix B).

For both ocean scenarios, Table 2 below shows errors caused by truncation of the number of bouncing in Eq. (24). This table can be reproduced by modifying the code in Figs. 3 and 4, which is the same for the isotropic, directional and ocean surfaces. Namely, one replaces inversion of the multiple bouncing matrix (Fig. 3 line 37, Fig. 4 line 45) with a truncated series (we keep 3 terms as an example):

```
RpRs = np.matmul(Rp_im, Rs)
RpRs_n = np.eye(ng3) + RpRs + np.matmul(RpRs, RpRs)
```

The multiple bouncing of the downwelling light at BOA (Fig. 4 line 45) becomes

```
lpb = np.matmul(RpRs_n, lp)
```

Table 2 shows that Eq. (24) converges rapidly for the black ocean: even $n = 1$ term is enough to meet the criterion formulated in [92: p.3343]: “the errors (in addition to the shot noise) on the intensity need to be $\leq 1\%$, and on the relative Stokes parameter q [Q/I] and u [U/I] need to be ≤ 0.002 ”, where $0.002 = 0.2\%$, and the combined as square root q - and u -errors give $\sim 0.3\%$ in DoLP.

What does not fit in the mentioned criterion and the measurement uncertainty (see Appendix B) is the error caused by misunderstanding of the system of coordinates between the vRT code and the surface adding functions. From Eq. (9), the sign of the relative azimuth is important for the Stokes- U component. This sign is a matter of definition: the relative azimuth could be assumed positive clockwise or counter-clockwise looking in or into the positive Z-direction (zenith) of the right-handed Cartesian system of coordinates. By artificially changing the sign of the relative azimuth inside the add_ocean() subroutine, for the DoLP we get up to 20% error (3% on average) in our Case 1 and up to 4% (1% on average) in Case 2 (recall, the atmosphere is 6 times thicker in Case 2 – multiple scattering tends to depolarize). The change of the azimuth does not affect the IPOL's upwelling path radiance (Stokes vector) since it comes in the azimuth space, but affects the computation of the TOA surface contribution. The total intensity remains unaffected in both cases.

Another conclusion, that follows from Table 1, is the necessity to speed-up the numerical simulation of the surface reflectance, especially for the ocean (see bold symbols for runtime in Table 1). We consider this software development problem in the next section.

5. Notes on the package implementation

Given a huge amount of satellite remote sensing data, numerical performance of the data processing software is always a concern. In Section 5.1 we offer a few ideas on how to speed up our Python code. Section 5.2 describes purpose and content of 16

Python sources, 12 binary files, and 11 txt-files, 39 files total, supplementing the paper.

5.1. Speeding-up the code

From Table 1 one sees that add_ocean() is unacceptably slow. The first candidate for acceleration is the Fourier expansion, oceanm(). This function is called for all Gaussian (Fig. 12: line 50) and user-defined (line 54) directions. These calls take 90% of the runtime in our case. Aside from the direct runtime measurement, the corresponding source code shows that add_lambert() and add_ocean() use the same MOM operations, while the runtime for add_lambert() is very small. The only difference between the two is the need for numerical expansion of the surface reflection in the Fourier series over the azimuth for each pair of zeniths of incidence/reflection. In addition to that, polarization inevitably slows down the vector surface, as compared to the directional scalar surface, by an order of magnitude.

An obvious step to accelerate the ocean function is to move the computation of the surface reflection matrix, for all directions, from the m -loop (see Fig. 16, right). Computation of the surface reflection matrix itself can be accelerated by optimizing its dependences: the Fresnel 3×3 reflection matrix, the rotation matrix, and the wave/shadow function. All the three depend on the $[\mu, \mu', \varphi - \varphi']$ -triplet. Refactoring these functions in a way that they take μ and μ' as input variables and the relative azimuth $\varphi - \varphi'$ as an array (because of Eq. (A5)) complicates the code insignificantly in any language, but offers the opportunity to precompute the φ -independent parameters in each function before the φ -loop (explicit or vectorized). The overall goal of this step is to reduce the number of the floating-point operations in general, and the number of computationally expensive calls to the trigonometric and square root functions in particular. We follow this strategy in the Fortran version of IPOL.

Storage of the surface reflection matrix for all directions in memory may run on an obstacle for cases, like the IPRT Case B4: “Standard atmosphere with cloud layer and underlying ocean surface” [50]. In the case like that the precomputed surface matrix becomes large, however reciprocal symmetry of its elements and the use of single precision instead of double helps relax the memory requirement.

Next, lines 51 and 55 in Fig. 12 access the elements of the corresponding arrays nonconsecutively when large matrices are filled in with 3×3 blocks. The 3×3 Mueller matrix is a natural and convenient output for the oceanm() function. The consecutive access is organized with buffer arrays. Using reflection from/to the Gaussian nodes only, and assuming, $r11 \dots r33$ are elements of the 3×3 Mueller matrix, one first creates lines of the big $3N \times 3N$ matrix. The lines **L1**, **L2**, **L3** are then combined in a big array **L**.

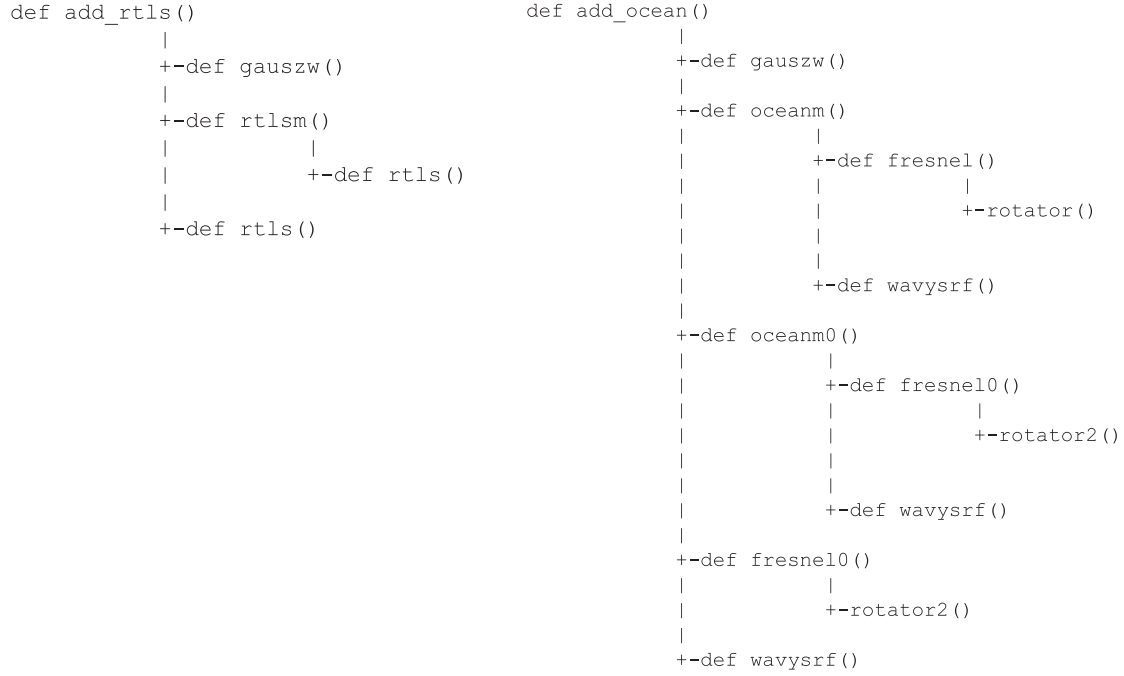


Fig. 16. Call graph for `add_rtls()` (left) and `add_ocean()` (right); `add_lambert()` has no dependences. See Table 3 for brief description of the functions.

```

L1 = np.zeros(3*ng1)
L2 = np.zeros_like(L1)
L3 = np.zeros_like(L1)
L = np.zeros((3*ng1, 3*ng1))
for ig in range(ng1):
    for jg in range(ng1):
        compute r11 ... r33 for [ig, jg]
        L1[3*jg:3*jg+3] = [r11, r12, r13]
        L2[3*jg:3*jg+3] = [r21, r22, r23]
        L3[3*jg:3*jg+3] = [r31, r32, r33]
    L[3*ig, :] = L1
    L[3*ig+1, :] = L2
    L[3*ig+2, :] = L3

```

Solution to the system of equations, `la.lsolve()` in Fig. 4, line 45 (same for `add_rtls()` and `add_ocean()`) takes place inside the solar loop over `imu0`. This Python function is based on the LU-decomposition. It is efficient either to perform the decomposition before the loop (LAPACK's `_getrf`) and then apply it in the loop (LAPACK's `_getrs`) or combine all solar zeniths in one matrix along a given row (column-wise). To avoid a potential confusion caused by transposing, in this paper we use all arrays as they come from IPOL. In this case the view zenith angles occupy columns in each given row (lead dimension).

Yet one more step is to use a compiled language. As Table 1 indicates, the C-implementation of the surface subroutines is ~ 2 orders of magnitudes faster. Like in the Python functions, we did not precompute the surface to optimize the C code. For clarity of our Python scripts, we opted for `numpy.matmul()`, instead of `numpy.dot()`, and for some other features, not supported by the Python compiler Numba. Running the Fourier im-loop in parallel for the directional reflection and using only the scalar modification of MOM ($N \times N$ matrices instead of $3N \times 3N$) for the isotropic one further decreases the runtime. Recall, even in the Lambertian case the MOM-based approach has an advantage over Eq. (1): one does not need to solve the RT problem three times with a black (for the path radiance) and two reflecting surfaces (for the bidirectional transmittance and atmospheric spherical albedo) in order to generate LUTs for the atmospheric components.

For reference, Table 1 also indicates the runtime for our vRT solver. We note that some acceleration is still possible in IPOL. For

example, we have not yet implemented the half-eigenproblem solution for $m > 0$, the matrix-operator method inside IPOL does not treat the upwelling and downwelling direct transmittances through optical layers (e.g., 2 in Case 3) separately from the diffuse ones, Eq. (38). Hence these matrices are sparse if $\text{nmu} \gg \text{ng1}$. Lastly, IPOL does not truncate the phase matrix.

All the Python sources, the IPOL input, and the benchmark data come with the paper. We suppose, it will take literally minutes of the user's time to download and reproduce the Python results from Table 1. An installed Python package is the only prerequisite. Digging through the code, of course, will take more time and require some effort. In order to help the reader, the next section describes content and purpose of all files and shows call graphs for the used Python functions.

5.2. Files description

This paper supports independent reproducibility of our results, and also serves as a documentation. Fig. 16 shows all Python dependences for `add_rtls()` and `add_ocean()`; `add_lambert()` needs none. Table 3 briefly describes what each function does. These mid-level and low-level functions are located in `./src` directory of the package. We created a simple unit-testing for each low-level function. The unit testing is executed when one runs the function as a separate script (`if __name__ == "__main__"`).

In the same root directory, there are three high-level caseX.py testing scripts ($X = 1, 2, 3$), and two folders: `./src/LUTs/` and `./src/benchmarkrks/`. The LUTs folder contains the binary files with the MOM matrices and text files with metadata. Names of the binary files match the content: `Jm_`, `Jp_`, `Rp_`, and `Tm_` contain, respectively, the TOA path radiance at user-defined directions, downwelling Fourier moments for the BOA path radiance at the Gaussian nodes, and the Fourier moments for the reflection and transmittance matrices. This data is stored in the double precision format because IPOL uses it for computations. In applications, it is of course sufficient to use single precision and thus decrease size of the arrays. The `metadata_.txt` files contain the total atmosphere optical thick-

Table 3

List of the Python functions, with short description, starting from the lowest level (no dependences). See Fig. 16 for the call graphs.

Name	Brief description
<code>gauszw()</code>	Computes the Gaussian nodes (zeros of the Legendre polynomial) and weights
<code>rotator()</code>	Computes elements for both rotation matrices
<code>rotator2()</code>	Same as <code>rotator()</code> except for the second rotation matrix only
<code>rtls()</code>	Computes the RTLS surface reflection model
<code>wavysrf()</code>	Computes the scaling factor that simulates waves, optionally with shadowing
<code>fresnel()</code>	Computes the rotated Fresnel reflection matrix
<code>fresnel0()</code>	Same as <code>fresnel()</code> except for the 1st column only
<code>oceanm()</code>	Computes the m -th Fourier moment (matrix) for the ocean reflection model
<code>oceanm0()</code>	Same as <code>oceanm()</code> , except for the 1st column only
<code>rtlsm()</code>	Computes the m -th Fourier moment for the RTLS surface reflection model
<code>add_lambert()</code>	Couples an atmosphere with the isotropic (Lambertian) surface
<code>add_ocean()</code>	Couples an atmosphere with the waved ocean surface (polarized)
<code>add_rtls()</code>	Couples an atmosphere with the directional land (RTLS) model

ness, the solar flux on TOA, the solar/view geometry, and the accuracy parameters (number of the Gaussian nodes and the Fourier moments). In order to avoid a potential confusion with the single vs. double Gaussian quadrature [93] and with the order in which the nodes are sorted, the metadata files contain the Gaussian nodes and weights (within the positive hemisphere, $\mu_+ > 0$). The benchmark data in the ASCII format is in the `./src/benchmarks/` folder.

Let's consider the workflow of `case1.py` as an example. It reads the input from `_case1.bin` files and `metadata_case1.txt`. After that, the script couples the

three different surfaces, (a) Lambertian; (b) RTLS; (c) ocean, with the same atmosphere, and computes the TOA signal. It then reads the benchmark data from `_case1a.txt`, `_case1b.txt`, and `_case1c.txt`, tests the MOM subroutines vs. corresponding benchmarks, and prints out on the screen the maximum and average deviation, in %, for the intensity and DoLP (see Table 1). The other two scripts, `case2.py` and `case3.py`, do the same, but only for one atmosphere and one surface each. A few other text files in the `./src/benchmarks/` folder are used for unit testing of the `fresnel()`, `fresnel0()`, `rotator()`, `rotator0()`, and `wavysrf()` functions.

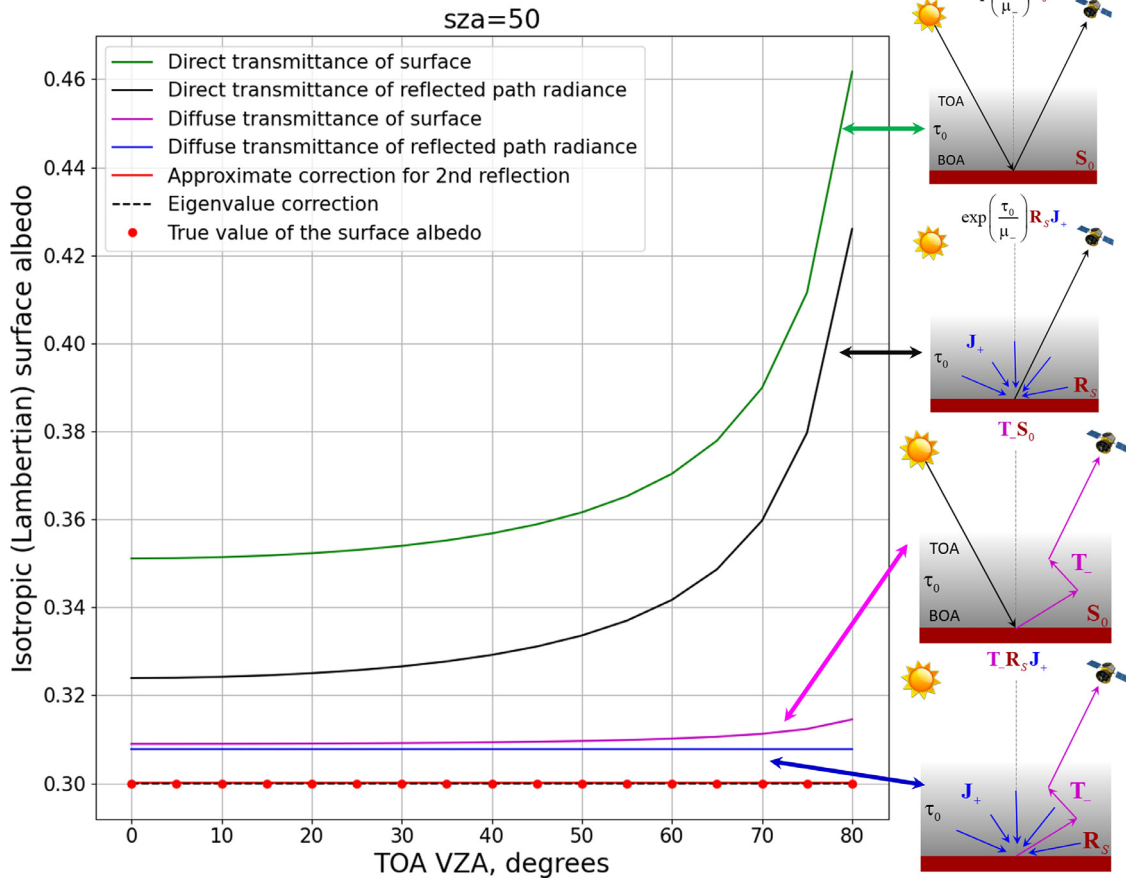


Fig. 17a. The Lambertian surface albedo (Y-axis) retrieved independently at different VZAs (X-axis). Scenario: conservative Rayleigh atmosphere, total optical thickness $\tau_0 = 0.1$, atmospheric spherical albedo $C_0 = 0.084$, the true value of the surface albedo is $\rho = 0.3$. Different lines account for different types of the atmosphere-surface interaction – see plots on the right. The path radiance is always compensated. Each effect, starting from the second top, includes the effects mentioned above it.

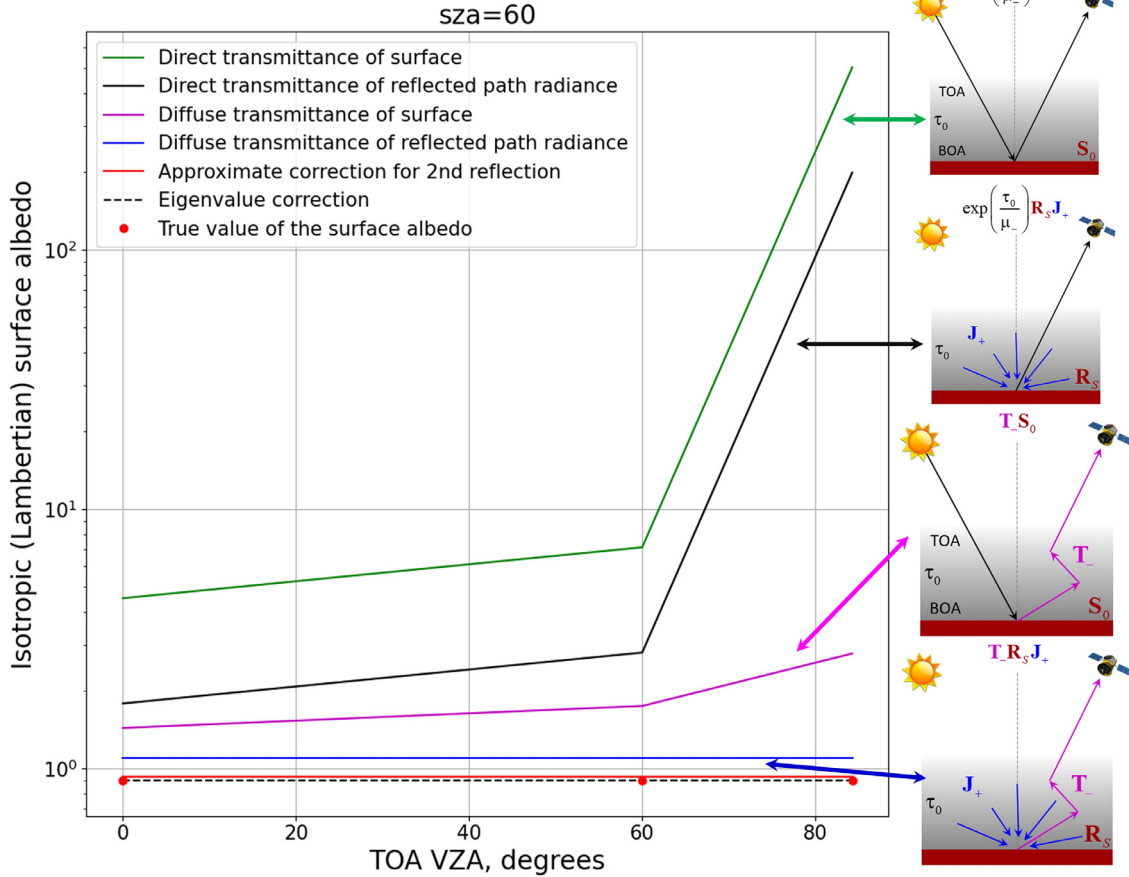


Fig. 17b. Same as in Fig. 17(a), except for atmosphere from Case 3: 2 layers, Rayleigh on top of the aerosol & Rayleigh mixture, $\tau_0 = 0.6$, $C_0 \approx 0.201$. A very bright Lambertian surface $\rho = 0.9$ replaces the default albedo $\rho = 0.1$ in order to increase the effect of multiple bouncing and its direct and diffuse contribution to TOA. The shown VZA dependences are not smooth because of 3 angles used in the Case 3 scenario (17 VZAs in Case 1).

6. Atmosphere-surface decoupling

In this part we remove the atmospheric effects from the TOA signal and retrieve the surface model parameters. We show numerical results for scenarios described in Section 4 using codes from Section 3. For that reason, the three subsections below discuss algorithm and data, but not codes. A subsection for the Lambertian model is followed by those for RTLS, and ocean.

6.1. Lambertian surface

We start with the atmospheric Case 1: pure Rayleigh scattering atmosphere, total optical thickness $\tau_0 = 0.1$ and the atmospheric albedo $C_0 \approx 0.084$ at BOA. From the simulated measurements, we want to retrieve the surface albedo, $\rho = 0.3$. Fig. 17(a) shows the value of the retrieved albedo (on Y-axis) plotted vs. the VZA at TOA (on X-axis). While the path radiance is always compensated, the different lines account for different types of the atmosphere-surface interaction (see Fig. 1).

- Only direct transmittances of the solar beam from TOA to the surface and back are considered – Fig. 1(a). In this case, the retrieved ρ preserves significant angular dependence even for this relatively thin atmosphere.
- In addition to (a), we include single reflection of the downward diffuse path radiance from surface – Fig. 1(b). After the reflection, this component is assumed to reach TOA directly without scattering. This step noticeably reduces error in the retrieved albedo, but significant angular dependence remains.

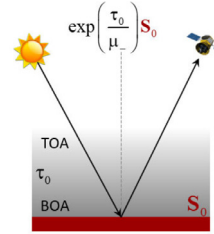
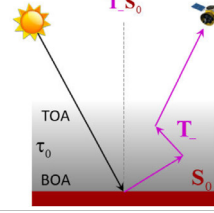
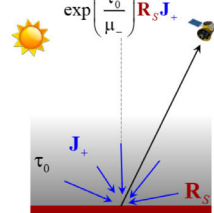
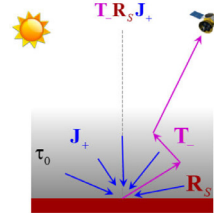
- In addition to (b), this option accounts for the diffuse BOA→TOA transmittance of the direct solar beam reflected once from the surface – Fig. 1(c). Only weak angular dependence remains at the horizon, VZA $\sim 80^\circ$, but the retrieved surface albedo $\rho \sim 0.31$ differs noticeably from the true one: $\rho = 0.3$;
- In addition to (c), simulation of the diffuse BOA→TOA transmittance of the downwelling path radiance, reflected once from the surface, is included – Fig. 1(d). This removes the angular dependence entirely (in this scenario), but the retrieved surface albedo is still not perfect.

The four options (a)-(d) account for the single reflection of the direct solar beam and the downwelling path radiance from the surface, and transmittance of that light directly and diffusely from BOA to TOA. Thus, (a)-(d) linearly depend on the surface albedo ρ . Combined, these components result in $\rho^* \approx 0.308$ or +2.6% w.r.t. the true value of 0.3.

- The error can be further reduced by considering double bouncing of light from the surface. For that, Eq. (24) must use two terms. The second reflection from the surface is directly proportional to $(\rho^* + \Delta\rho)^2 \approx \rho^*\rho^* + 2\rho^*\Delta\rho$, where $\rho^* \approx 0.308$ is known from the previous step (d), and $\Delta\rho^2$ is ignored. This yields linear equation for the perturbation parameter $\Delta\rho$ and a good agreement with truth in this scenario.
- Finally, Eq. (13) yields an exact solution for multiple bouncing.

Table 4

Retrieval of the RTLS parameters assuming single bouncing of light from surface. The left column indicates a component of the atmosphere-surface interaction (see Fig.1); all components mentioned above it are assumed included. The "Variable" column indicates the kernel weights k , the white-sky albedo A , the maximum and average relative error in the TOA signal for k retrieved using the indicated component(s). The rest columns correspond to three atmospheric cases; the total optical thickness τ_0 and the atmospheric spherical albedo C_0 at BOA are indicated for reference. The four bottom rows show the true values; "-" means unrealistic result.

Component	Variable	Case 1: $\tau_0 = 0.1$, $C_0 \approx 0.084$	Case 2: $\tau_0 = 0.6302$, $C_0 \approx 0.343$	Case 3: $\tau_0 = 0.6$, $C_0 \approx 0.201$
	k_L	0.3545	-	-
	k_V	0.1549	-	-
	k_G	0.0548	-	-
	A	0.308	-	-
	$\Delta I_{MAX}, \%$	39	-	-
	$\Delta I_{AVR}, \%$	17	-	-
	k_L	0.3491	-	0.7839
	k_V	0.0757	-	0.1586
	k_G	0.0636	-	0.1121
	A	0.276	-	0.660
	$\Delta I_{MAX}, \%$	13	-	110
	$\Delta I_{AVR}, \%$	8.2	-	47
	k_L	0.3337	0.7497	0.4042
	k_V	0.0633	0.4979	0.3252
	k_G	0.0646	0.0580	0.0567
	A	0.257	0.764	0.388
	$\Delta I_{MAX}, \%$	4.7	41	29
	$\Delta I_{AVR}, \%$	2.3	34	15
	k_L	0.3336	0.7462	0.3398
	k_V	0.0576	0.1444	0.0609
	k_G	0.0654	0.1244	0.0646
	A	0.254	0.602	0.262
	$\Delta I_{MAX}, \%$	2.7	16	3.3
	$\Delta I_{AVR}, \%$	1.6	13	1.4
RTLS true parameters	k_L	0.3300	0.6600	0.3300
	k_V	0.0530	0.1060	0.0530
	k_G	0.0660	0.132000	0.0660
	A	0.249	0.498	0.249

For the atmospheric Case 3, $C_0 \approx 0.201$ and $\rho = 0.1$. The effect of multiple bouncing at BOA, $C_0 \rho \approx 0.02$, is close to that in Case 1, hence not very interesting. To increase the effect of multiple bouncing at BOA, we redefine the surface albedo to $\rho = 0.9$, and simulate a new TOA signal for the same atmospheric MOM matrices. Fig. 17(b) shows the result. Recall, Case 3 uses 3 VZAs – so the lines look broken. Noteworthy, the combined effect of the single bouncing of light (see (a)–(d) above), yields an “efficient” $\rho \approx 1.099$, the approximate second bouncing results in $\rho \approx 0.930$ (see (e) above), and inclusion of the multiple bouncing gives the true value $\rho \approx 0.900$ (for both SZA = 60° and 84°).

6.2. RTLS

This section illustrates the use of the Green’s function formalism, Eq. (6), and its MOM version Eq. (27). Table 4 shows the re-

sult of the minimization depending on the type of the atmosphere-surface interaction, Fig. 1. The table also indicates the surface white-sky albedo¹³ A , and the maximum and average deviation of the TOA signal from the one obtained for the true surface.

Table 4 indicates that neglecting of multiple bouncing causes ΔI -errors that exceed the measurement ones (see Appendix B) even for a thin atmosphere. Section 2.2.3 above explains how to account for multiple bouncing iteratively. In Table 5, we show convergence of the iterations for the considered atmospheres. Each iterative step uses precomputed atmospheric MOM parameters. Thus, the atmospheric RT calculation is avoided. One only needs to

¹³ Ratio of the reflected-over-incident fluxes for an ideal isotropic illumination. We calculated the fluxes numerically using $nga = 90$ and $ng1 = 16$ Gaussian nodes for integration over the relative azimuth and zenith angles, respectively; accuracy was confirmed by doubling the number of the nodes.

Table 5

Convergence of iterations to account for multiple bouncing of light at BOA. The iteration number is shown in the left column, the rest content coincides with that in Table 4. Iterations start from the “best guess” – last set of parameters in Table 4.

Iteration #	Variable	Case 1: $\tau_0 = 0.1$, $C_0 \approx 0.084$	Case 2: $\tau_0 = 0.6302$, $C_0 \approx 0.343$	Case 3: $\tau_0 = 0.6$, $C_0 \approx 0.201$
0 (from Table 4)	k_L	0.3336	0.7462	0.3399
	k_V	0.0576	0.1444	0.0609
	k_G	0.0654	0.1244	0.0646
	A	0.254	0.602	0.262
	$\Delta I_{MAX}, \%$	2.7	16	3.3
	$\Delta I_{AVR}, \%$	1.6	13	1.4
1	k_L	0.3298	0.6140	0.3286
	k_V	0.0528	0.0840	0.0522
	k_G	0.0660	0.1366	0.0661
	A	0.249	0.442	0.248
	$\Delta I_{MAX}, \%$	0.1	8.0	0.4
	$\Delta I_{AVR}, \%$	0.1	6.6	0.2
2	k_L	0.3300	0.6801	0.3302
	k_V	0.0530	0.1158	0.0531
	k_G	0.0660	0.1300	0.0660
	A	0.249	0.523	0.249
	$\Delta I_{MAX}, \%$	0.0	3.6	0.0
	$\Delta I_{AVR}, \%$	0.0	3.0	0.0
3	k_L		0.6502	0.3300
	k_V		0.1011	0.0530
	k_G		0.1330	0.0660
	A		0.486	0.249
	$\Delta I_{MAX}, \%$		1.8	0.0
	$\Delta I_{AVR}, \%$		1.5	0.0
4			...	
5	k_L		0.6578	
	k_V		0.1049	
	k_G		0.1322	
	A		0.496	
	$\Delta I_{MAX}, \%$		0.4	
	$\Delta I_{AVR}, \%$		0.3	
...				
11	k_L		0.6600	
	k_V		0.1060	
	k_G		0.1320	
	A		0.498	
	$\Delta I_{MAX}, \%$		0.0	
	$\Delta I_{AVR}, \%$		0.0	
RTLS true parameters	k_L	0.3300	0.6600	0.3300
	k_V	0.0530	0.1060	0.0530
	k_G	0.0660	0.1320	0.0660
	A	0.249	0.498	0.249

“attach” the surface with the updated parameters k to the known atmosphere using Eq. (13) to get a new TOA signal.

It takes 2, 11, and 3 iterations in Case 1, 2, and 3, respectively, to get 4 digits in the kernel weights precisely. This numerical exercise, although impractical due to measurement errors, confirms numerical stability of the technique. On practice, we wish to converge within some measurement error, which is typically a fraction of percent. The Cases 1 and 3 achieve this accuracy using 1 iteration, Case 2 needs 5. We ran 15 iterations for all the cases to make sure they do not diverge.

6.3. Ocean

In this section, we extract a non-linear parameter of the ocean surface reflectance model. The ocean surface model depends on the refractive index of water, n , and the windspeed, w (m/s), at 10 meters (m) above the surface. For our numerical example we pick the latter one and express it in terms of the wave slope distribution, $\sigma^2 = 0.00512 w + 0.003$. Following Table 3, we ignore multiple bouncing of light at the expense of an error not exceeding 0.5%. In the Case 1 atmosphere, it is sufficient to subtract the path radiance from the TOA signal and account for the direct component of the bidirectional transmittance to get σ^2 properly. It is for that reason we focus on the Case 2 atmosphere only: $\tau_0 = 0.6301$ and $C_0 \approx 0.343$.

For the Case 2 atmosphere, we have a set of simulated polarized measurements for the VZA from 0° (nadir) to 60° with 5° step, $SAZ = 30^\circ$ and 60° , and $RAA = 0^\circ, 60^\circ, 180^\circ$, and 240° . The true windspeed is $w = 7$ (m/s); it corresponds to the slope variance $\sigma^2 = 0.03884$. This value is retrieved by minimizing the sum of square residuals between the TOA measurements m_i and simulations s_i at $N = 104$ mentioned solar/view geometries ($i = 1 \dots N$):

$$\chi^2 = \sum_{i=1}^N (m_i - s_i)^2. \quad (50)$$

We used two types of signals and simulations for Eq. (50): the total intensity I and the polarized intensity

$$I_p = \sqrt{Q^2 + U^2}. \quad (51)$$

The right column in Table 6 shows the dependence of χ^2 (Y-axis) on σ^2 (X-axis) for the total (red line) and polarized (blue line) signals. For each one, we indicate the percentile deviation of the retrieved value of σ^2 (as minimum of χ^2) from the true value $\sigma^2 = 0.03884$ (vertical dash line). The middle column shows what component we include to simulate the signal: from the direct bidirectional transmittance (row #1 – see left column) to all types of single bouncing of light from the ocean surface (bottom row). The latter one yields accurate retrieval of the surface parameter provided azimuthal dependence of the diffuse transmittance

Table 6

Right column: the squared residual as function of the surface parameter for the total I (red line) and polarized I_p (blue line) signals. Middle column: components of the surface TOA contribution used in simulation for the given row (like for the Lambertian and RTLS cases). Left column: row # for reference. Note, two options in row #4: 4A includes the azimuthal dependence for the diffuse transmittance, while 4B ignores it. For each signal, we indicate the percentile difference between the retrieved (circle) and the exact value (0.03884 or 7 m/s – vertical dash line).

Row #	TOA contribution component	Residual χ^2 (on Y) as function of wave slope distribution σ^2 (on X)
1		
2		
3		
4A		
4B		

is accounted for: row #4A. In row #4B we ignore the azimuthal dependence for the diffuse transmittance from BOA to TOA. Not only this gives inaccurate results, but also makes the polarized retrieval “worse” compared to the total intensity. We calculated the σ^2 -grid for the wind speeds w from 0.25 to 10 m/s, step 0.25 m/s (40 grid points). The exact solution is therefore included. We also note that in this noiseless example, the use of the polarized signal shows noticeably better accuracy both in terms of the retrieved

value and the shape of the χ^2 at its minimum. More distinct shape of the “polarized” (blue) curve in the vicinity of the minimum helps solution to converge, but only if the polarized and azimuthal properties of the atmospheric diffuse transmittance are properly simulated.

On practice, one uses iterative atmospheric correction algorithms with linearization of the RT solution w.r.t. the non-linear parameters of the surface reflectance model.

7. Summary

Using elements of the Python code, we have shown in this paper how the known matrix-operator method (MOM) helps address the problem of radiative (de-)coupling of a plane-parallel atmosphere and a spatially homogeneous surface. MOM simulates the atmospheric and surface optical properties, as matrices, independently of each other. This is convenient for atmospheric correction. A few matrix multiplications and one matrix inversion “attach” the surface model to the atmospheric one in order to get the TOA signal. That one matrix inversion simulates multiple reflection (bouncing) of light at BOA.

It is possible to limit the number of reflections between the atmosphere and surface. For the ocean studies, where the TOA signal is accurately represented by the path radiance, and the BOA→TOA transfer of the specular reflection (glint) and the polarized water leaving radiance, the MOM approach conveniently accounts for the azimuthal and polarized dependence of the atmospheric bidirectional transmittance. It is important to adhere to the same definition of the direction of positive rotation (e.g., of the relative azimuth or the plane of reference for polarization) in the vRT code and the ocean water leaving Stokes vector. Otherwise, the error in the simulated polarization noticeably exceeds that for the measurement.

MOM also yields the formulation of the TOA signal as the sum of the path radiance, and the TOA contribution from the single and multiple bouncing of the downwelling light. For both contributions, MOM gives analytical expressions. This form of solution generalizes the MAIAC's Green's function formalism for polarization. A slight difference from MAIAC is that we do not rely on the analytical formula for the multiple reflection (residual). We get the one from the vRT equation net the path radiance and the single bouncing contribution. Thus, the residual preserves the dependence on both polarization and azimuth.

If polarization is not used, the MOM approach is simply an alternative to the scalar Green's function technique. However, with MOM, one does not have to solve the adjoint problem separately if the RT code calculates the MOM elements as standard output. Using this output is beneficial even for the Lambertian surface model. The MOM-based RT code generates all necessary LUTs within one run, while the code with “traditional” output – radiance or reflectance – needs two (actual and adjoint atmosphere) or three (path radiance, atmospheric spherical albedo, bidirectional transmittance). As a drawback, MOM is more demanding for memory required to store matrices in the Fourier space ($m = 0, 1, 2, \dots, M$). One should keep in mind this memory problem for tasks like hyperspectral atmospheric correction, or linearization of solution w.r.t. atmospheric parameters.

One of the key features of this paper is the “equations + code” style. We explain the Python sources for computation of the surface MOM-matrices and show how to connect these to the atmospheric ones. We use independently reproducible numerical examples for the isotropic (Lambertian), directional (RTLS), and polarized ocean surface reflection models, covered by three different atmospheres: Rayleigh at two wavelengths, and a 2-layer atmosphere with a Rayleigh layer over another layer consisting of a Rayleigh-aerosol mixture. The results are tested vs. published benchmarks. These examples exhaustively describe how to couple the plane-parallel atmosphere with any existing surface model, including the effect of polarization of the monochromatic light.

For convenient use of the atmospheric matrices, we have modified our vRT code IPOL. In the new modification, IPOL no longer takes the surface model parameters on input. Instead, the code now provides as standard output the monochromatic path radiances **J** (Stokes vector) at TOA and BOA, the reflection matrix **R** at

BOA, and the BOA→TOA transmission matrix **T**. This output seems very convenient, but not very common, for the MOM-based codes, especially those designed for the atmospheric correction. Using **J**, **R**, and **T**, and the surface matrices created outside IPOL, one computes the total TOA signal using functions explained in this paper, i.e. solves the forward problem.

Once, the forward problem is solved, we illustrate how to perform the atmosphere correction for the three surface models. For the Lambertian surface we illustrate contribution of different components of the atmosphere-surface interaction to the TOA signal. For the kernel driven RTLS model, the linear least squares method fits the kernel weights to the TOA signal. One calculates the multiple bouncing contribution iteratively. At the first iteration, the multiple bouncing of light at BOA is ignored. We have illustrated the dependence of the retrieved surface model parameters on what component of the atmosphere-surface interaction is accounted for. Our numerical examples also show how iterations for the multiple bouncing component converge in case of three atmospheric scenarios. We also note that the RTLS model, despite it is depolarizing, contributes to the TOA polarization via atmospheric scattering, like the directional unpolarized solar beam gets polarized at scattering. In the considered scenarios, this contribution is at the level of the measurement accuracy of the modern passive polarimeters. Hence this effect may need further investigation.

One of the practical applications of the paper is to support the atmospheric correction of the polarized observations. Like RTLS, a common model of the polarized reflectance of land is represented as a linear mixture of the soil and vegetation components. Each component is expressed using the Fresnel Mueller matrix as a kernel. It is convenient to simulate the transfer of each Fresnel kernel from the surface to the TOA using multiplication of the corresponding surface matrix and the atmospheric MOM-matrices. The relative fractions of the soil and vegetation within the field of view appear as weights at the BOA→TOA transferred kernel. However, due to coarse spatial resolution of the current polarimeters (e.g., ~6 km for POLDER), good understanding of the surface polarized reflectance model is still missing.

This paper helps address another practical need to accelerate the process of the polarized atmospheric correction by saving the Fourier m -moments in the LUT. In case of a smooth azimuthal dependence of the multiple scattering path radiance, this approach may offer a reduction of the size of the LUT. In particular, for the Rayleigh scattering, one can compress an arbitrary azimuthal grid into just three Fourier moments. In the aerosol retrieval algorithms, the MOM approach may be used to remove the surface as a LUT-dimension and thus either relax the memory requirements, or use the freed memory, e.g., for keeping more aerosol parameters as variables, or for a finer grid over the zenith angles.

In prospect, we plan to apply the reported results to the measurement data processing. We have two potential goals in mind. First, over ocean, the polarization and azimuthal dependence of the bidirectional atmospheric transmittance still seems to be a problem. Second, we are planning to generalize the MOM-based approach to a non-homogeneous surface by accounting the spatial-dependent surface reflectance for light bounced once, and some “averaged” reflectance in multiple bouncing. This approach, without polarization, has been reported in other papers – we have cited some of them in the Introduction.

At the end of the paper, we'd like to emphasize once again that independent reproducibility of the reported results is possible by downloading our Python sources, precomputed atmospheric MOM matrices, and the benchmark data from the journal website or from our GitHub repository: https://github.com/korkins/jqsrt2022_mom_ipol.

Authors contribution

SK: methodology (matrix-operator form of the Green's function technique); coding (vRT code IPOL and Python functions); writing (original draft). AL: funding acquisition; methodology (the Green's function technique for atmospheric correction – algorithm MAIAC); writing (review and editing).

Declaration of Competing Interest

Authors of this manuscript declare no conflicts of interests.

Data Availability

No data was used for the research described in the article.

Acknowledgments

This research has been supported by the Earth Sciences Division grants NNH20ZDA001N-SNPPSP and 19-PACESAT19-0039 (PI: A. Lyapustin on both). This work was presented at the Advancement of POLarimetric Observations (APOLO-2022) conference in Section “Databases, software and artificial intelligence / machine learning applications”.

Supplementary materials

Supplementary material associated with this article can be found, in the online version, at doi:[10.1016/j.jqsrt.2023.108663](https://doi.org/10.1016/j.jqsrt.2023.108663).

Appendix A. Fourier expansion of the surface reflection models

The Fresnel reflection matrix

$$\mathbf{F} = \begin{bmatrix} r_+ & r_- & 0 \\ r_- & r_+ & 0 \\ 0 & 0 & r_0 \end{bmatrix} \quad (\text{A1})$$

depends on the reflection coefficients of the parallel, r_l (w.r.t. the plane of reflection), and perpendicular, r_r , components of the electromagnetic wave as $r_+ = r_l^2 + r_r^2$, $r_- = r_l^2 - r_r^2$, $r_0 = r_l r_r$. The rotation matrix through an angle $\alpha > 0$ is [84, p. 11]

$$\mathbf{L}(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(2\alpha) & \sin(2\alpha) \\ 0 & -\sin(2\alpha) & \cos(2\alpha) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c & s \\ 0 & -s & c \end{bmatrix}. \quad (\text{A2})$$

The rotated reflection matrix combines Eqs. (A1) and (A2) (note the subscripts) [84, p. 68]

$$\begin{aligned} \rho &= \mathbf{SPL}(\pi - \alpha_2) \mathbf{FL}(-\alpha_1) \\ &= \begin{bmatrix} r_+ & r_- c_1 & -r_- s_1 \\ r_- c_2 & r_+ c_1 c_2 - r_0 s_1 s_2 & -r_+ c_1 c_2 - r_0 s_1 s_2 \\ r_- s_2 & r_+ c_1 s_2 + r_0 s_1 c_2 & r_+ c_1 s_2 + r_0 s_1 c_2 \end{bmatrix} \\ &= \begin{bmatrix} \rho_{11} & \rho_{12} & \rho_{13} \\ \rho_{21} & \rho_{22} & \rho_{23} \\ \rho_{31} & \rho_{32} & \rho_{33} \end{bmatrix}. \end{aligned} \quad (\text{A3})$$

Reflection of the unpolarized direct solar beam requires only the first column of Eq. (A3), hence only the second rotator - see `rotator2()` in Fig. 16.

The elements ρ_{11} , ρ_{12} , ρ_{21} , ρ_{22} , and ρ_{23} are expanded in the Fourier series using the cosine function

$$\rho(x) = \sum_{m=0}^M (2 - \delta_{0,m}) \rho_m \cos(mx). \quad (\text{A4})$$

where the expansion moment is

$$\rho_m = \frac{1}{2\pi} \int_0^{2\pi} \rho(x) \cos(mx) dx = \int_0^1 \rho(2\pi y) \cos(2\pi my) dy. \quad (\text{A5})$$

It is a standard practice to use the Gaussian quadrature of the same order for all m -s in Eq. (A5). One figures out a sufficient order by trial-and-error. The Gaussian order usually varies from a few tens [94: p.331] to a few hundreds.

The elements ρ_{31} and ρ_{32} use the sine-function, and the elements ρ_{31} and ρ_{23} use the same with minus (see Eq. (37) in [32] and lines 118–127 in `ocean.phase.f`¹⁴). Consider the latter as an example:

$$\rho_{23}(x) = \sum_{m=0}^M (2 - \delta_{0,m}) \rho_{23}^m \sin(mx) \quad (\text{A6})$$

$$\rho_{23}^m = -\frac{1}{2\pi} \int_0^{2\pi} \rho(x) \sin(mx) dx \quad (\text{A7})$$

Note that the minus in Eq. (A7) depends on the definition of the relative azimuth (in general, on the system of coordinates): clockwise or counter-clockwise looking in- or into- the positive Z-direction. Depending on the combination, it is possible to get the same “correct” sign despite using different systems e.g., for the atmospheric path radiance and the surface model.

Appendix B. Measurement uncertainty for some passive polarimeters

Listed below are the sensor acronyms and band centers (in nanometers, nm) for their polarization-sensitive bands (P); dash “-” in the table means we were not able to find clear definition for the corresponding level of uncertainty.

- 3MI – Multi-viewing Multi-channel Multi-polarization Imager: 410P, 443P, 490P, 555P, 670P, 763, 760, 865P, 910, 1370P, 1650P, 2130P
- AirMSPI – Airborne Multiangle SpectroPolarimetric Imager: 470P, 660P, 865P
- APS – Aerosol Polarimetry Sensor: 410P, 443P, 555P, 670P, 865P, 910P, 1380P, 1610P, 2200P;
- DPC – Directional Polarimetric Camera: 490P, 670P, 865P
- HARP – Hyper-Angular Rainbow Polarimeter: 440P, 550P, 670P, 870P;
- MAIA – Multi-Angle Imager for Aerosols: 444P, 646P, 1044P,
- POLDER – POLarization and Directionality of the Earth's Reflectance: 443P, 670P, 865P;
- RSP – Research Scanning Polarimeter: 410P, 470P, 555P, 670P, 865P, 960P, 1590P, 1880P, 2250P;
- SGLI – Second generation GLobal Imager: 675P, 870P;
- SPEXone – Spectro-Polarimeter for Planetary EXploration: 385P-770P (hyperspectral)

Instrument	Radiometric accuracy	Polarimetric accuracy	Refs.
3MI	< 2%	1% - 2%	[95, p. 1802; 96, p. 4500]
AirMSPI	–	< 0.3%	[97, p. 2008]
APS	3%–5% (8% at 1380 nm)	0.2% - 0.4%	[98, p. 7; 99, p. 3]
DPC	< 5%	< 2%	[100, p. 23]
HARP	3% - 5%	0.5% - 1.5%	[101, p. 5210]
MAIA	–	0.3% - 0.5%	[96, p. 4500]
POLDER-3	< 5%	1% - 2%	[96, p. 4500; 102, p. 89]
RSP	< 5%	< 0.2%	[87, p. 684; 103, p. 36]
SGLI	< 3%	–	[104, p. 3]
SPEXone	2%	0.3%	[105, p. 5; 106, p. 172]

See also Table 1. “Salient Aerosol Sensor Characteristics” in [107].

¹⁴ <https://www.giss.nasa.gov/staff/mmishchenko/ftpcode/ocean.phase.f>

References

- [1] Chandrasekhar S. Radiative transfer. London: Oxford University Press; 1950.
- [2] Lenoble J. Radiative transfer in scattering and absorbing atmospheres: standard computational procedures editor. Hampton, VA: A Deepak Publishing; 1985.
- [3] de Haan JF, Hovenier JW, Kokke JMM, van Stokkom HTC. Removal of atmospheric influences on satellite-borne imagery: a radiative transfer approach. Remote Sens Environ 1991;37(1):1–21. doi:10.1016/0034-4257(91)90046-9.
- [4] Chowdhary J, et al. Modeling atmosphere-ocean radiative transfer: a PACE mission perspective. Front Earth Sci 2019;7(100):1–53. doi:10.3389/feart.2019.00100.
- [5] Remer LA, et al. Retrieving aerosol characteristics from the PACE mission, Part 2: multi-angle and polarimetry. Front Environ Sci 2019;7(94):1–21. doi:10.3389/fenvs.2019.00094.
- [6] Ottaviani M, Cairns B, Ferrare R, Rogers R. Iterative atmospheric correction scheme and the polarization color of alpine snow. J Quant Spectrosc Radiat Transf 2012;113(10):789–804. doi:10.1016/j.jqsrt.2012.03.014.
- [7] Tanre D, Herman M, Deschamps PY, de Lefle A. Atmospheric modeling for space measurements of ground reflectances, including bidirectional properties. Appl Opt 1979;18(21):3587–94. doi:10.1364/AO.18.003587.
- [8] Vermote EF, Kotchenova S. Atmospheric correction for the monitoring of land surfaces. J Geophys Res Atmos 2008;113(23):1–12. doi:10.1029/2007JD009662.
- [9] Levy RC, Munchak LA, Mattoo S, Patadia F, Remer LA, Holz RE. Towards a long-term global aerosol optical depth record: applying a consistent aerosol retrieval algorithm to MODIS and VIIRS-observed reflectance. Atmos Meas Tech 2015;8(10):4083–110. doi:10.5194/amt-8-4083-2015.
- [10] Hsu NC, Tsay SC, King MD, Herman JR. Aerosol properties over bright-reflecting source regions. IEEE Trans Geosci Remote Sens 2004;42(3):557–69. doi:10.1109/TGRS.2004.824067.
- [11] Jackson JM, et al. Suomi-NPP VIIRS aerosol algorithms and data products. J Geophys Res Atmos 2013;118(22):12,612–673,689Nov. doi:10.1002/2013JD020449.
- [12] Vasilkov A, Krotkov N, Haffner D, Fasnacht Z, Joiner J. Estimates of hyperspectral surface and underwater UV planar and scalar irradiances from OMI measurements and radiative transfer computations. Remote Sens 2022;14(9). doi:10.3390/rs14092278.
- [13] Lyapustin A, Wang Y, Korkin S, Huang D. MODIS Collection 6 MAIAC algorithm. Atmos Meas Tech 2018;11(10):5741–65. doi:10.5194/amt-11-5741-2018.
- [14] Hsu NC, Lee J, Sayer AM, Kim Y, Bettenhausen C, Tsay SC. VIIRS deep blue aerosol products over land: extending the EOS long-term aerosol data records. J Geophys Res Atmos 2019;124(7):4026–53 Apr. doi:10.1029/2018JD029688.
- [15] Lamsal LN, et al. Ozone Monitoring Instrument (OMI) Aura nitrogen dioxide standard product version 4.0 with improved surface and cloud treatments. Atmos Meas Tech 2021;14(1):455–79. doi:10.5194/amt-14-455-2021.
- [16] Lyapustin AI. Atmospheric and geometrical effects on land surface albedo. J Geophys Res Atmos 1999;104(D4):4127–43 Feb. doi:10.1029/1998JD000664.
- [17] Lyapustin A, Tedesco M, Wang Y, Aoki T, Hori M, Kokhanovsky A. Retrieval of snow grain size over Greenland from MODIS. Remote Sens Environ 2009;113(9):1976–87. doi:10.1016/j.rse.2009.05.008.
- [18] Lyapustin A, Muldashev T, Wang Y. Code SHARM: fast and accurate radiative transfer over spatially variable anisotropic surfaces. In: Light scattering reviews 5. Single light scattering and radiative transfer. Chichester, UK: Springer; 2010. p. 205–47. A. Kokhanovsky, editor.
- [19] Limbacher JA, Kahn RA, Lee J. The new MISR research aerosol retrieval algorithm: a multi-angle, multi-spectral, bounded-variable least squares retrieval of aerosol particle properties over both land and water. Atmos Meas Tech Discuss 2022;2022:1–34. doi:10.5194/amt-2022-95.
- [20] Gordon HR, Du T, Zhang T. Atmospheric correction of ocean color sensors: analysis of the effects of residual instrument polarization sensitivity. Appl Opt 1997;36(27):6938–48. doi:10.1364/AO.36.006938.
- [21] Gordon HR. Atmospheric correction of ocean color imagery in the earth observing system era. J Geophys Res Atmos 1997;102(D14):17081–106 Jul. doi:10.1029/96JD02443.
- [22] Vasilkov A, Lyapustin A, Mitchell BG, Huang D. UV reflectance of the ocean from DSCOVR/EPIC: comparisons with a theoretical model and Aura/OMI observations. J Atmos Ocean Technol 2023;36(11):2087–99. doi:10.1175/JTECH-D-18-0150.1.
- [23] Mobley CD. The oceanic optics book editor. Dartmouth, NS, Canada: International Ocean Colour Coordinating Group (IOCCG); 2022.
- [24] Frouin RJ, et al. Atmospheric correction of satellite ocean-color imagery during the PACE Era. Front Earth Sci 2019;7(July):1–43. doi:10.3389/feart.2019.00145.
- [25] Gordon HR, Franz BA. Remote sensing of ocean color: assessment of the water-leaving radiance bidirectional effects on the atmospheric diffuse transmittance for SeaWiFS and MODIS intercomparisons. Remote Sens Environ 2008;112(5):2677–85. doi:10.1016/j.rse.2007.12.010.
- [26] Kattawar GW, Plass GN, Guinn JA. Monte Carlo calculations of the polarization of radiation in the earth's atmosphere-ocean system. J Phys Oceanogr 1973;3(4):353–72. doi:10.1175/1520-0485(1973)003<0353:MCCOTP>2.0.CO;2.
- [27] Chowdhary J, Cairns B, Travis LD. Contribution of water-leaving radiances to multiangle, multispectral polarimetric observations over the open ocean: bio-optical model results for case 1 waters. Appl Opt 2006;45(22):5542–67. doi:10.1364/AO.45.005542.
- [28] Lyapustin A, Knyazikhin Y. Green's function method for the radiative transfer problem I Homogeneous non-Lambertian surface. Appl Opt 2001;40(21):3495. doi:10.1364/ao.40.003495.
- [29] Lyapustin A, Zhao F, Wang Y. A comparison of multi-angle implementation of atmospheric correction and MOD09 daily surface reflectance products from MODIS. Front Remote Sens 2021;2(December):1–15. doi:10.3389/frsen.2021.712093.
- [30] Wang Y, Lyapustin AI, Privette JL, Morissette JT, Holben B. Atmospheric correction at AERONET locations: a new science and validation data set. IEEE Trans Geosci Remote Sens 2009;47(8):2450–66. doi:10.1109/TGRS.2009.2016334.
- [31] Wang Y, et al. Assessment of biases in MODIS surface reflectance due to Lambertian approximation. Remote Sens Environ 2010;114(11):2791–801. doi:10.1016/j.rse.2010.06.013.
- [32] Mishchenko MI, Travis LD. Satellite retrieval of aerosol properties over the ocean using polarization as well as intensity of reflected sunlight. J Geophys Res Atmos Jul 1997;102(D14):16989–7013. doi:10.1029/96JD02425.
- [33] Waquet F, et al. Polarimetric remote sensing of aerosols over land. J Geophys Res Atmos Jan 2009;114(D1). doi:10.1029/2008JD010619.
- [34] Korkin S, Sayer AM, Ibrahim A, Lyapustin A. A practical guide to writing a radiative transfer code. Comput Phys Commun 2022;271:108198. doi:10.1016/j.cpc.2021.108198.
- [35] Schaepman-Strub G, Schaepman ME, Painter TH, Dangel S, Martonchik JV. Reflectance quantities in optical remote sensing—definitions and case studies. Remote Sens Environ 2006;103(1):27–42. doi:10.1016/j.rse.2006.03.002.
- [36] Stamnes K, Thomas GE, Stamnes JJ. Radiative transfer in the atmosphere and ocean. Cambridge University Press; 2017.
- [37] Stokes GG. On the intensity of the light reflected from or transmitted through a pile of plates. Proc R Soc Lond 1860;11:545–56.
- [38] Tuckerman LB. On the intensity of the light reflected from or transmitted through a pile of plates. J Opt Soc Am 1947;37(10):818–25. doi:10.1364/JOSA.37.000818.
- [39] Kattawar GW. Peregrinations through topics in light scattering and radiative transfer. J Quant Spectrosc Radiat Transf 2016;178:5–13. doi:10.1016/j.jqsrt.2015.10.013.
- [40] Evans KF, Stephens GL. Many polarized radiative transfer models. J Quant Spectrosc Radiat Transf 2010;111(11):1686–8. doi:10.1016/j.jqsrt.2010.01.029.
- [41] Emde C, et al. The libRadtran software package for radiative transfer calculations (version 2.0.1). Geosci Model Dev 2016;9(5):1647–72. doi:10.5194/gmd-9-1647-2016.
- [42] Mishchenko MI. The fast invariant imbedding method for polarized light: computational aspects and numerical results for Rayleigh scattering. J Quant Spectrosc Radiat Transf 1990;43(2):163–71. doi:10.1016/0022-4073(90)90045-8.
- [43] Mishchenko MI, Dlugach JM, Yanovitskij EG, Zakharova NT. Bidirectional reflectance of flat, optically thick particulate layers: an efficient radiative transfer solution and applications to snow and soil surfaces. J Quant Spectrosc Radiat Transf 1999;63(2):409–32. doi:10.1016/S0022-4073(99)00028-X.
- [44] Nakajima T, Tanaka M. Matrix formulations for the transfer of solar radiation in a plane-parallel scattering atmosphere. J Quant Spectrosc Radiat Transf 1986;35(1):13–21. doi:10.1016/0022-4073(86)90088-9.
- [45] Ota Y, Higurashi A, Nakajima T, Yokota T. Matrix formulations of radiative transfer including the polarization effect in a coupled atmosphere-ocean system. J Quant Spectrosc Radiat Transf 2010;111(6):878–94. doi:10.1016/j.jqsrt.2009.11.021.
- [46] Liu Q, Weng F. Advanced doubling-adding method for radiative transfer in planetary atmospheres. J Atmos Sci 2006;63(12):3459–65. doi:10.1175/JAS3808.1.
- [47] Liu Q, Weng F. Using Advanced Matrix Operator (AMOM) in community radiative transfer model. IEEE J Sel Top Appl. Earth Obs Remote Sens 2013;6(3):1211–18. doi:10.1109/JSTARS.2013.2247026.
- [48] Liu Q, Cao C. Analytic expressions of the Transmission, Reflection, and source function for the community radiative transfer model. J Quant Spectrosc Radiat Transf 2019;226:115–26. doi:10.1016/j.jqsrt.2019.01.019.
- [49] Korkin S, Lyapustin A. Matrix exponential in C/C++ version of vector radiative transfer code IPOL. J Quant Spectrosc Radiat Transf 2019;227:106–10. doi:10.1016/j.jqsrt.2019.02.009.
- [50] Emde C, et al. IPRT polarized radiative transfer model intercomparison project - Phase A. J Quant Spectrosc Radiat Transf 2015;164:8–36. doi:10.1016/j.jqsrt.2015.05.007.
- [51] Kokhanovsky A, et al. Retrieval of snow properties from the Sentinel-3 ocean and land colour instrument. Remote Sens 2019;11(19):2280. doi:10.3390/rs11192280.
- [52] Kokhanovsky AA, Smirnov A, Korkin SV, Wind G, Slutsker I. The retrieval of cloud properties based on spectral solar light diffuse transmittance measurements under optically thick cloud cover conditions. J Quant Spectrosc Radiat Transf 2020;251:107008. doi:10.1016/j.jqsrt.2020.107008.
- [53] Korkin S, Lyapustin A, Sinyuk A, Holben B, Kokhanovsky A. Vector radiative transfer code SORD: performance analysis and quick start guide. J Quant Spectrosc Radiat Transf 2017;200:295–310. doi:10.1016/j.jqsrt.2017.04.035.
- [54] Sinyuk A, et al. The AERONET Version 3 aerosol retrieval algorithm, associated uncertainties and comparisons to Version 2. Atmos. Meas. Tech. Discuss. 2020:1–80. doi:10.5194/amt-2019-474.
- [55] Moler C, Van Loan C. Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later. SIAM Rev 2003;45(1):3–49 Jan. doi:10.1137/S003614450240180.

- [56] Efremenko DS, Molina García V, García SG, Doicu A. A review of the matrix-exponential formalism in radiative transfer. *J Quant Spectrosc Radiat Transf* 2017;196:17–45. doi:[10.1016/j.jqsrt.2017.02.015](#).
- [57] Karp AH, Greenstadt J, Fillmore JA. Radiative transfer through an arbitrarily thick, scattering atmosphere. *J Quant Spectrosc Radiat Transf* 1980;24(5):391–406. doi:[10.1016/0022-4073\(80\)90074-6](#).
- [58] Doicu A, Trautman T, Schreier F. Numerical regularization for atmospheric inverse problems. Berlin Heidelberg: Springer; 2010.
- [59] Efremenko D, Doicu A, Loyola D, Trautmann T. Acceleration techniques for the discrete ordinate method. *J Quant Spectrosc Radiat Transf* 2013;114:73–81. doi:[10.1016/j.jqsrt.2012.08.014](#).
- [60] Rozanov VV, Rozanov AV, Kokhanovsky AA, Burrows JP. Radiative transfer through terrestrial atmosphere and ocean : software package SCIATRAN. *J Quant Spectrosc Radiat Transf* 2014;133:13–71. doi:[10.1016/j.jqsrt.2013.07.004](#).
- [61] Spurr R, Christi M. The LIDORT and VLIDORT linearized scalar and vectordiscrete ordinate radiative transfer models: updates in the Last 10 years. In: *Springer series in light scattering*. Springer; 2019. p. 1–62. A. A. Kokhanovsky.
- [62] Kawata Y. Circular polarization of sunlight reflected by planetary atmospheres. *Icarus* 1978;33(1):217–32. doi:[10.1016/0019-1035\(78\)90035-0](#).
- [63] Lenoble J, Herman M, Deuzé JL, Lafrance B, Santer R, Tanré D. A successive order of scattering code for solving the vector equation of transfer in the earth's atmosphere with aerosols. *J Quant Spectrosc Radiat Transf* 2007;107(3):479–507. doi:[10.1016/j.jqsrt.2007.03.010](#).
- [64] Smith OJ, Siewert CE. The Half-Space Green's function for an atmosphere with a polarized incident field. *J Math Phys* 1967;8(12):2467–74 Dec. doi:[10.1063/1.1705181](#).
- [65] Odell AP, Weinman JA. The effect of atmospheric haze on images of the Earth's surface. *J Geophys Res* 1975;80(36):5035–40 Dec. doi:[10.1029/JC080i036p05035](#).
- [66] Van Diedenhoven B, Hasekamp OP, Landgraf J. Efficient vector radiative transfer calculations in vertically inhomogeneous cloudy atmospheres. *Appl Opt* 2006;45(23):5993–6006. doi:[10.1364/AO.45.005993](#).
- [67] Xu F, et al. Joint retrieval of aerosol and water-leaving radiance from multispectral, multiangular and polarimetric measurements over ocean. *Atmos Meas Tech* 2016;9(7):2877–907. doi:[10.5194/amt-9-2877-2016](#).
- [68] Coulson KL. Effects of reflection properties of natural surfaces in aerial reconnaissance. *Appl Opt* 1966;5(6):905–17. doi:[10.1364/AO.5.000905](#).
- [69] Breon F, Tanre D, Lecomte P, Herman M. Polarized reflectance of bare soils and vegetation: measurements and models. *IEEE Trans Geosci Remote Sens* 1995;33(2):487–99. doi:[10.1109/TGRS.1995.8746030](#).
- [70] Waquet F, Léon JF, Cairns B, Goloub P, Deuzé JL, Auriol F. Analysis of the spectral and angular response of the vegetated surface polarization for the purpose of aerosol remote sensing over land. *Appl Opt* 2009;48(6):1228–36. doi:[10.1364/AO.48.001228](#).
- [71] Litvinov P, Hasekamp O, Cairns B. Models for surface reflection of radiance and polarized radiance: comparison with airborne multi-angle photopolarimetric measurements and implications for modeling top-of-atmosphere measurements. *Remote Sens Environ* 2011;115(2):781–92. doi:[10.1016/j.rse.2010.11.005](#).
- [72] Breon FM, Maignan F. A BRDF-BPDF database for the analysis of Earth target reflectances. *Earth Syst Sci Data* 2017;9(1):31–45. doi:[10.5194/essd-9-31-2017](#).
- [73] Lyapustin A, Knyazikhin Y. Green's function method in the radiative transfer problem. II. Spatially heterogeneous anisotropic surface. *Appl Opt* 2002;41(27):5600–6. doi:[10.1364/AO.41.005600](#).
- [74] Lyapustin A, Martonchik J, Wang Y, Laszlo I, Korkin S. Multiangle implementation of atmospheric correction (MAIAC): 1. Radiative transfer basis and look-up tables. *J Geophys Res Atmos* 2011;116(3). doi:[10.1029/2010JD014985](#).
- [75] Lyapustin AI, et al. Multi-angle implementation of atmospheric correction for MODIS (MAIAC): 3. Atmospheric correction. *Remote Sens Environ* 2012;127:385–93. doi:[10.1016/j.rse.2012.09.002](#).
- [76] Lyapustin A, et al. Atmospheric correction of DSCOVR EPIC: version 2 MAIAC algorithm. *Front Remote Sens* 2021;2(September):1–10. doi:[10.3389/frsen.2021.748362](#).
- [77] Lyapustin AI. Radiative transfer code SHARM for atmospheric and terrestrial applications. *Appl Opt* 2005;44(36):7764–72. doi:[10.1364/AO.44.007764](#).
- [78] Kokhanovsky AA, et al. Benchmark results in vector atmospheric radiative transfer. *J Quant Spectrosc Radiat Transf* 2010;111(12–13):1931–46. doi:[10.1016/j.jqsrt.2010.03.005](#).
- [79] Chowdhary J, Zhai PW, Xu F, Frouin R, Ramon D. Testbed results for scalar and vector radiative transfer computations of light in atmosphere-ocean systems. *J. Quant Spectrosc Radiat Transf* 2020;242. doi:[10.1016/j.jqsrt.2019.106717](#).
- [80] Lucht W, Schaaf CB, Strahler AH. An algorithm for the retrieval of albedo from space using semiempirical BRDF models. *IEEE Trans Geosci Remote Sens* 2000;38(2):977–98. doi:[10.1109/36.841980](#).
- [81] Roujean JL, Leroy M, Deschamps PY. A bidirectional reflectance model of the Earth's surface for the correction of remote sensing data. *J Geophys Res Atmos* 1992;97(D18):20455–68 Dec. doi:[10.1029/92JD01411](#).
- [82] Li X, Strahler AH. Geometric-optical bidirectional reflectance modeling of the discrete crown vegetation canopy: effect of crown shape and mutual shading. *IEEE Trans Geosci Remote Sens* 1992;30(2):276–92. doi:[10.1109/36.134078](#).
- [83] Shurkiff WA. Polarized light. production and use. Cambridge, MA: Harvard University Press; 1962.
- [84] Hovenier JW, der Mee CVM, Domke H. Transfer of polarized light in planetary atmospheres: basic concepts and practical methods. Dordrecht: Kluwer Academic Publishers; 2004.
- [85] Nakajima T, Tanaka M. Effect of wind-generated waves on the transfer of solar radiation in the atmosphere-ocean system. *J Quant Spectrosc Radiat Transf* 1983;29(6):521–37. doi:[10.1016/0022-4073\(83\)90129-2](#).
- [86] Gordon HR, Wang M. Surface-roughness considerations for atmospheric correction of ocean color sensors. 1: the Rayleigh-scattering component. *Appl Opt* 1992;31(21):4247–60. doi:[10.1364/AO.31.004247](#).
- [87] Mishchenko M, et al. Accurate monitoring of terrestrial aerosols and total solar irradiance: the NASA Glory mission. In: *Proceedings of the international geoscience and remote sensing symposium*; 2010. p. 758–60. doi:[10.1109/IGARSS.2010.5652996](#).
- [88] Bodhaine BA, Wood NB, Dutton EG, Slusser JR. On Rayleigh optical depth calculations. *J Atmos Ocean Technol* 1999;16(11 PART 2):1854–61. doi:[10.1175/1520-0426\(1999\)016<1854:orodc>2.0.co;2](#).
- [89] de Haan JF, Bosma PB, Hovenier JW. The adding method for multiple scattering calculations of polarized light. *Astron Astrophys* 1987;183:371–91.
- [90] Deirmendjian D. Electromagnetic scattering on spherical polydispersions. New York: Elsevier; 1969.
- [91] de Rooij WA, van der Stap CCAH. Expansion of Mie scattering matrices in generalized spherical functions. *Astron Astrophys* 1984;131:237–48. Feb[Online]. Available: <https://ui.adsabs.harvard.edu/abs/1984A&A...131..237D>.
- [92] Hasekamp OP, Landgraf J. Retrieval of aerosol properties over land surfaces: capabilities of multiple-viewing-angle intensity and polarization measurements. *Appl Opt* 2007;46(16):3332–43. doi:[10.1364/AO.46.003332](#).
- [93] Sykes JB. Approximate integration of the equation of transfer. *Mon Not R Astron Soc* 1951;111(4):377–86 Aug. doi:[10.1093/mnras/111.4.377](#).
- [94] Spurr RJD. VLIDORT: a linearized pseudo-spherical vector discrete ordinate radiative transfer code for forward model and retrieval studies in multilayer multiple scattering media. *J Quant Spectrosc Radiat Transf* 2006;102(2):316–42. doi:[10.1016/j.jqsrt.2006.05.005](#).
- [95] Hioki S, Riedi J, Djellali MS. A study of polarimetric error induced by satellite motion: application to the 3MI and similar sensors. *Atmos Meas Tech* 2021;14(3):1801–16 Mar. doi:[10.5194/amt-14-1801-2021](#).
- [96] van Harten G, et al. Calibration and validation of Airborne Multiangle Spectropolarimetric Imager (AirMSPi) polarization measurements. *Appl Opt* 2018;57(16):4499. doi:[10.1364/ao.57.004499](#).
- [97] Diner DJ, et al. The Airborne Multiangle Spectropolarimetric Imager (AirMSPi): a new tool for aerosol and cloud remote sensing. *Atmos Meas Tech* 2013;6(8):2007–25. doi:[10.5194/amt-6-2007-2013](#).
- [98] Peralta RJ, et al. Aerosol polarimetry sensor for the Glory Mission. *Proc SPIE* 2007;6786 Nov. doi:[10.1117/12.783307](#).
- [99] Persh S, et al. Ground performance measurements of the Glory Aerosol Polarimetry Sensor. *Proc SPIE* 2010;7807 Aug. doi:[10.1117/12.862029](#).
- [100] Li Z, et al. Directional Polarimetric Camera (DPC): monitoring aerosol spectral optical properties over land from satellite observation. *J Quant Spectrosc Radiat Transf* 2018;218:21–37. doi:[10.1016/j.jqsrt.2018.07.003](#).
- [101] Puthukkudy A, et al. Retrieval of aerosol properties from Airborne Hyper-Angular Rainbow Polarimeter (AirHARP) observations during ACEPOL 2017. *Atmos Meas Tech* 2020;13(10):5207–36. doi:[10.5194/amt-13-5207-2020](#).
- [102] Andre Y, Laherrere JM, Bret-Dibat T, Jouret M, Martinuzzi JM, Perbos JL. Instrumental concept and performances of the POLDER instrument. *Proc SPIE* 1995;2572 Aug. doi:[10.1117/12.216932](#).
- [103] B. Cairns, E.E. Russell, J.D. LaVeigne, and P.M.W. Tennant, "Research scanning polarimeter and airborne usage for remote sensing of aerosols," *Polariz. Sci. Remote Sens.*, vol. 5158, p. 33, 2003, doi: [10.1117/12.518320](#).
- [104] Okamura Y, Tanaka K, Amano T, Hiramatsu M, Shiratama K. Design and breadboarding activities of the second-generation Global imager (SGLI) on GCOM-C. *Proc SPIE* 2017;10566 Nov. doi:[10.1117/12.2308222](#).
- [105] van Amerongen A, et al. SPExone: a compact multi-angle polarimeter. *Proc SPIE* 2019;11180 Jul. doi:[10.1117/12.2535940](#).
- [106] Hasekamp OP, et al. Aerosol measurements by SPExone on the NASA PACE mission: expected retrieval capabilities. *J Quant Spectrosc Radiat Transf* 2019;227(2019):170–84. doi:[10.1016/j.jqsrt.2019.02.006](#).
- [107] Diner DJ, Davis A, Hancock B, Gutt G, Chipman RA, Cairns B. Dual-photoelastic-modulator-based polarimetric imaging concept for aerosol remote sensing. *Appl Opt* 2007;46(35):8428–45. doi:[10.1364/AO.46.008428](#).