User's Guide

VLIDORT
Version 2.7

Robert Spurr

RT Solutions, Inc.
9 Channing Street, Cambridge, MA 02138, USA
Tel. +1 617 492 1183
Fax: +1 617 492 1183 (request only)
email: rtsolutions@verizon.net

Foreword

This is the User's Guide to VLIDORT Version 2.7 issued in September 2014 in conjunction with the release of the Version 2.7 FORTRAN 90 software package and accompanying license, and an accompanying document containing a description of the test data sets in the installation package. The associated license closely follows the GNU public license formulation. Version 2.7 is the 7th official release. It follows the distribution of Version 1.0 in autumn 2004, Version 2.0 in January 2006, Version 2.3 in October 2007, Version 2.4 in September 2009, Version 2.5 in December 2010 and Version 2.6 in June 2012.

Table of Contents

1. Introduction to VLIDORT

1.1 Historical and background overview

1.1.1 Polarization in radiative transfer

The modern treatment of the equations of radiative transfer for polarized light dates back to the pioneering work by Chandrasekhar in the 1940s [*Chandrasekhar*, 1960]. Using a formulation in terms of the Stokes vector for polarized light, Chandrasekhar was able to solve completely the polarization problem for an atmosphere with Rayleigh scattering, and benchmark calculations from the 1950s are still appropriate today [*Coulson et al.*, 1960]. Researchers started looking at the scattering properties of polarized light by particles, and new more general formulations of the scattering matrices were developed independently by Hovenier [*Hovenier*, 1971] and Dave [*Dave*, 1970], and subsequently used in studies of polarization by Venus.

With the advent of more powerful computers, a series of numerical RTMs were developed through the 1980s; many of these have become standards. In particular, the DISORT discrete ordinate model developed by Stamnes and co-workers was released in 1988 for general use [*Stamnes, et al.*, 1988]. Most RTMs today are either discrete ordinate codes or doubling-adding methods, and vector models are no exception. In the 1980s, Siewert and colleagues made a number of detailed mathematical examinations of the vector RT equations. The development of the scattering matrix in terms of generalized spherical functions was reformulated in a convenient analytic manner [*Siewert*, 1981; *Siewert*, 1982; *Vestrucci and Siewert*, 1984], and most models now follow this work (this includes VLIDORT). Siewert and co-workers then carried out an examination of the discrete ordinate eigenspectrum for the vector equations, and developed complete solutions for the slab problem using the spherical harmonics method [*Garcia and Siewert*, 1986] and the F_N method [*Garcia and Siewert*, 1989]. These last two solutions have generated benchmark results for the slab problem.

Also in the 1980s, a group in the Netherlands carried out some parallel developments. Following detailed mathematical studies by Hovenier and others [*Hovenier and van der Mee*, 1983; *de Rooij and van der Stap*, 1984], a general doubling-adding model was developed for atmospheric radiative transfer modeling [*de Haan et al.*, 1987; *Stammes et al.*, 1989]. This group was also able to provide benchmark results for the slab problem [*Wauben and Hovenier*, 1992]. Vector discrete ordinate models were developed in the 1990s, with VDISORT [*Schultz et al.*, 2000] and its generalization [*Schultz and Stamnes*, 2000] to include the post processing function. In 1998, Siewert revisited the slab problem from a discrete ordinate viewpoint, and developed new and elegant solutions for the scalar [*Siewert*, 2000a] and vector [*Siewert*, 2000b] problems. One new ingredient in these solutions was the use of Green's functions to develop particular solutions for the solar scattering term [*Barichello et al.*, 2000]. For the vector problem, Siewert's analysis showed that complex eigensolutions for the homogeneous RT equations must be considered. Siewert also provided a new set of benchmark results [*Siewert*, 2000b]; this set and the results from [*Garcia and Siewert*, 1989] constitute our standards for slab-problem validation with aerosols.

1.1.2 Development of Linearized Vector RT models

In the last fifteen years, there has been increasing recognition of the need for RT models to generate fields of analytic radiance derivatives (Jacobians) with respect to atmospheric and surface variables, in addition to simulated radiances. Such "linearized" models are extremely useful in classic inverse problem retrievals involving iterative least-squares minimization (with and without regularization). At each iteration step, the simulated radiation field is expanded in a Taylor series about the given state of the atmosphere-surface system. Only the linear term in this expansion is retained, and this requires partial derivatives of the simulated radiance with respect to atmospheric and surface parameters that make up the state vector of retrieval elements and the vector of assumed model parameters that are not retrieved but are sources of error in the retrieval.

It is well known that the use of scalar radiative transfer (neglecting polarization) can lead to considerable errors for modeling backscatter spectra in the UV [*Mishchenko et al.*, 1994; *Lacis et al.*, 1998; *Sromovsky,* 2005]. Studies with atmospheric chemistry instruments such as GOME, SCIAMACHY and OMI have shown that the treatment of polarization is critical for the successful retrieval of ozone profiles from UV backscatter [*Schutgens and Stammes*, 2003; *Hasekamp et al.*, 2002]. The role of polarization has been investigated for retrieval scenarios involving important backscatter regions such as the oxygen *A* band [*Stam et al.*, 1999, *Jiang et al.*, 2003; *Natraj et al.*, 2006]. It has also been demonstrated that the use of passive sensing instruments with polarization capabilities can greatly enhance retrievals of aerosol information in the atmosphere [*Mishchenko and Travis*, 1997; *Deuze et al.*, 2000]; this is becoming a very important issue as the scientific community tries to understand the effects of aerosol forcing [*Heintzenberg et al.*, 1996; *Mishchenko et al.*, 2004].

Satellite instruments such as GOME-2 (launched in October 2006) [*EPS/METOP*, 1999] and OCO (Orbital Carbon Observatory) [*Crisp et al.*, 2004] are polarizing spectrometers; vector radiative transfer is an essential ingredient of the forward modeling component of their retrieval algorithms. Vector RT modeling is slower than its scalar counterpart, and the treatment of polarization in forward modeling has often involved the creation of look-up tables of "polarization corrections" to total intensity. However, with the advent of new and planned instruments measuring polarization, there is a need for linearized vector models to deal directly with retrieval issues.

Historically, a number of linearized RT models were developed for the scalar RTE some years ago [*Rozanov et al.*, 1998; *Landgraf et al.*, 2001; *Spurr et al.*, 2001]. This includes the LIDORT linearization (see below). Linearized vector radiative transfer models include the Gauss-Seidel code [Landgraf et al, 2005 and reference therein], and the linearized VLIDORT model [*Spurr*, 2006].

1.2 Overview of the LIDORT and VLIDORT models

In this section we present a developmental review of the LIDORT and VLIDORT models. In sections 1.2.1 and 1.2.2 respectively, we summarize the earlier Fortran 77 versions up to the year 2010 The most recent versions with re-organized codes and full Fortran 90 capability are summarized separately in section 1.2.3.

Table 1.1 gives a quick overview of the main developments and associated version numbers.

Table 1.1 Major features of LIDORT and VLIDORT.

Feature	LIDORT Version	VLIDORT Version
Pseudo-spherical (solar beam attenuation)	2.1	1.0
[Enhanced spherical (line-of-sight)]	2.2+	2.1
Green's function treatment	2.3	n/a
3-kernel BRDF + linearization	2.4	2.2
Multiple solar zenith angles	3.0	2.2
Solution saving, BVP telescoping	3.0	2.3
Linearized thermal & surface emission	3.2	2.4
Outgoing sphericity correction	3.2	2.3
Total Column Jacobian facility	3.3	2.4
Transmittance-only thermal mode	3.3	2.4
Fortran 90 release	3.5	2.5
BRDF supplement	3.5	2.5
Structured I/O	3.5	2.5
External SS	3.6	2.6
BRDF upgrade and surface-leaving supplements	3.6	2.6
Atmospheric and surface blackbody Jacobians	3.7	2.7
Codes made thread-safe for parallel computing	3.7	2.7
Introduction of Taylor series expansions	3.7	2.7
BRDF and surface-leaving supplement upgrades	3.7	2.7

1.2.1 LIDORT Scalar Model, Versions 1.0 to 3.3

The first version of LIDORT was developed in 1999 with the linearization of the complete discrete ordinate multiple-scattering RT solutions in a multi-layer atmosphere. Production of weighting functions was restricted to TOA (top-of-atmosphere) upwelling output, with the atmospheric medium treated for solar beam propagation in a plane-parallel medium [*Spurr et al.*, 2001]. Version 1.1 of LIDORT was able to generate atmospheric profile weighting functions and surface albedo weighting functions (Lambertian). It also included an initial treatment of atmospheric thermal emission source terms. The linearization was done by perturbation analysis.

In 2000 and 2001, the second versions of LIDORT were developed, to include pseudo-spherical treatment of the solar beam attenuation in a curved atmosphere, and to extend the model for the output of weighting functions at arbitrary optical depths for downwelling and upwelling fields. In these models, the linearization formalism was cast in terms of analytic differentiation of the complete discrete ordinate solution. Green's function methods were developed for solving the radiative transfer equation (RTE) for solar beam source terms. This work culminated in the release of Versions 2.3S (radiance only) and 2.3E (with Jacobians) [*Spurr,* 2002; *Van Oss & Spurr*, 2002].

In 2003, the LIDORT Version 2.2+ code was developed as a super-environment for LIDORT [*Spurr*, 2003]; this code has an exact treatment of single scattering for curved line-of-sight paths, thus giving LIDORT an "enhanced sphericity" treatment suitable for important satellite applications involving wide off-nadir viewing geometry (such as that for the Ozone Monitoring Instrument (OMI) which has a 2600 km swath). Version 2.4 developed in 2002-2003 provided a

number of extensions to deal in particular with bidirectionally reflecting surfaces [*Spurr*, 2004]. BRDF functions were set up for a number of surface types using a linear combination of pre-set BRDF kernels (these are semi-empirical functions developed for particular types of surfaces), and a complete differentiation of the BRDF formulation was developed to generate Jacobians with respect to surface variables such as the wind speed.

Support for maintaining LIDORT Versions 1 and 2 came from a series of contracts over the period 1999-2004 which provided the sources of funding for R. Spurr while at SAO. In recognition of the need for a consistent set of supported RT codes for use at NASA-GSFC, a contract was set up between SSAI Inc. and RT Solutions Inc. for the developmental release of LIDORT Versions 3.0 and beyond; all subsequent User Guides were written under this aegis.

From Version 3.0 onwards, all previous LIDORT codes are integrated. The last of the older versions to be incorporated was Version 2.5; this had the specialist ability for fast generation of total column (as opposed to profile) Jacobians, and this capability was integrated into LIDORT 3.3 in May 2007. New features for Versions 1.0 to 3.3 are (1) a Green's function treatment of the RTE with atmospheric thermal emission; (2); a new outgoing sphericity correction that replaces the old treatment of single scatter along the line-of-sight path; (3) the inclusion of a bulk property (total column) weighting function treatment. A number of other improvements were added, including a stand-alone facility for returning the single scatter radiance and Jacobians, an additional scaling procedure for the single scatter RTE, and an internal adjustment to utilize geometrical variables at any height in the atmosphere.

1.2.2 VLIDORT Development, Versions 1.0 to 2.4

In December 2003, a proposal was made to FMI for R. Spurr to develop the vector model VLIDORT as part of the O3SAF Visiting Scientist (VS) program in 2004. The first version of VLIDORT was completed in July 2004, given shakedown tests and validated against the Coulson/Dave/Sekera Rayleigh results [*Coulson et al.*, 1960] and Siewert's [*Siewert*, 2000b] benchmark results. The first application started in August 2004 with polarization sensitivity studies on the UV product algorithm at FMI, and the Version 1.0 User's guide appeared in September 2004.

In January 2005, a proposal was made and accepted for the continuation of VLIDORT studies as part of the Ozone SAF Visiting Scientist Work in 2005. A number of VLIDORT improvements (refractive geometry, single scatter corrections, and performance enhancements) were made in spring 2005, and this was followed by an end-to-end linearization of the code, so that the new version of VLIDORT now possessed a complete weighting-function capability. This December 2005 version marked the completion of the initial VLIDORT development [*Spurr*, 2006] A further validation against the benchmark results of [*Garcia and Siewert*, 1989] was performed at this time.

Support for VLIDORT maintenance and development from 2006 has come from an RT Solutions' contract with SSAI and NASA GSFC. The first new development for LIDORT and VLIDORT was the introduction of a new and more accurate single scatter scheme to allow for spherical geometry along the view path as well as the solar paths. The model has been used extensively at NASA GSFC in OMI-related studies, and in spring 2007, it was carefully validated against the older TOMRAD code at GSFC.

For VLIDORT version 2.4R, the linearization facility was extended to include column or bulk property Jacobians, a facility that was introduced into the scalar LIDORT code in 2003. In

addition to the new bulk Jacobian facility, version 2.4R also contains some new BRDF specifications for polarized reflectance from land surfaces. Thermal emission was introduced into this version of the code..

In 2006, R. Spurr was invited to contribute a chapter on the LIDORT and VLIDORT models in the book Light Scattering Reviews 3. This article [*Spurr*, 2008] contains a complete exposition of the theory behind the models, and the mathematical description in the present volume follows this review article closely.

1.2.3 LIDORT / VLIDORT Fortran 90 versions (3.5-3.7, 2.5-2.7)

In recent years, many users have moved over to Fortran 90 programming for LIDORT and VLIDORT applications; among other reasons, this has necessitated a complete revision of the software. In 2010, the LIDORT code was translated to Fortran 90 (Version 3.5), followed by VLIDORT in 2011 (Version 2.5). [Fortran 77 versions of these packages are still supported, though Versions 2.6 and 2.7 of VLIDORT are only available in Fortran 90]. Although there has been some new physics introduced in these versions, the VLIDORT and LIDORT organization and coding has been overhauled in order to bring the codes in line with modern computing standards. An important consideration has been the need for the codes to function in a parallel computing environment; this has meant that all COMMON blocks and associated "include" files have been scrapped, to be replaced by explicit argument declarations for all inputs and outputs.

The LIDORT and VLIDORT packages have equivalent capability. Here, we list the following attributes for VLIDORT Versions 2.5 and 2.6 (upgrades for the latter version are noted separately):

1. With the exception of the file VLIDORT.PARS, which contains only parameter statements for symbolic array dimensioning, fixed indices and fixed numerical constants, all "include" files in previous Fortran 77 versions of VLIDORT have been removed.

2. All variables are explicitly declared, and all input and output arguments clearly notated as such. All routines have "implicit none" opening statements. All "GO TO" statements have been removed.

3. All Fortran 90 subroutine argument declarations have the intent(in), intent(out) and intent(in out) characterizations. Fortran 90 input and output arguments to the top-level VLIDORT calling routines are organized into a number of Type structures.

4. A new exception handling system has been introduced. Formerly, input-check and calculation errors were written to file as they occurred during model execution. This is not convenient for applications where VLIDORT is embedded in a larger system; now, VLIDORT 2.5 will collect messages for output, and return error traces.

5. The multi-kernel BRDF implementation has been moved out of the main VLIDORT model, and now exists as a supplement. VLIDORT will now ingest exact BRDFs (for use in single scatter corrections) and for the multiple scatter field, all Fourier components of the total BRDF (and any surface property derivatives) at discrete ordinate, solar and viewing angle stream directions. The BRDF supplement provides these inputs.

6. The use of "normalized" weighting functions output has been discontinued for surface linearization. This makes it possible for example to define an albedo weighting function in the limit of zero albedo.

7. The new VLIDORT package has 2 master routines: one for intensity simulations alone, and a second (called the "LPCS" master) for calculations of atmospheric *profile* or column weighting functions and surface property Jacobians.

8. (Version 2.6). The BRDF supplement has been upgraded to include a facility for generating surface glitter BRDFs to include multiple reflections from wave facets.

9. (Version 2.6). There is a new "surface-leaving" capability (e.g. for ocean water-leaving or fluorescence applications), and VLIDORT can ingest the correct functions for modeling this physical effect in the RTE. This additional "VSLEAVE" supplement provides these functions.

10. (Version 2.6). Single-scatter calculations are now optional; the model can ingest single scatter fields from external sources.

11. (Version 2.6). An observational geometry facility has been added to improve computational efficiency when doing satellite applications. The facility was incorporated into VLIDORT's main code as well as the BRDF and VSLEAVE supplements.

Next, we list the upgrades for VLIDORT Version 2.7:

1. (Version 2.7). VLIDORT now has the capability to run in an OpenMP parallel computing environment suitable for multi-core machines. This performance enhancement is useful for hyperspectral applications involving many calls to VLIDORT over a spectral window.

2. (Version 2.7). A number of Taylor series expansions have been introduced to avoid numerical instability arising when there is a close coincidence between two polar angle directions.

3. (Version 2.7). A new facility for the generation of Black-Body Jacobians has been introduced as a proxy for temperature Jacobians in the thermal scattering regime.

4. An alternative single-scatter "first-order" (FO) code is now available to VLIDORT Version 2.7; this code is stand-alone, with no dependency on the rest of the package.

1.3 LIDORT-RRS, 2-stream, Linearized Mie/Tmatrix codes

The LIDORT linearization techniques have been applied to the CAO_DISORT coupled atmospheric-ocean code, and it is now possible to generate weighting function with respect to marine constituents such as chlorophyll concentration and CDOM [*Spurr, et al.*, 2007]. This has opened the way for a new approach to simultaneous retrieval of atmospheric and ocean quantities from MODIS and related instruments. [*Li et al*., 2007].

In 2002, a version of LIDORT with inelastic rotational Raman scattering (RRS) was developed from first principles, using an analytic solution of the discrete ordinate field in the presence of additional source terms due to RRS. This work was written up in [*Spurr et al*., 2008], and encompasses Versions 1.5 through 2.1. of the LRRS (LIDORT-RRS) code package. The LRRS code has been used in a number of applications involving ozone profile and column retrievals from instruments such as GOME and OMI. In 2009, a major new development for the LRRS code was the complete linearization of the entire model for profile, column and surface Jacobians. A separate User's Guide is available LRRS, and the package is also available in Fortran 90. LRRS is currently at Version 2.3.

A dedicated 2-stream version of the multiple-scattering LIDORT code was written in 2010, for use in low-stream interpolation and performance enhancement in hyperspectral retrieval applications involving many radiative transfer techniques [*Spurr and Natraj*, 2011]. This 2S code is entirely analytical, avoiding the use of LAPACK or other numerical schemes.

More recently, some new work on the linearization of the T-matrix electromagnetic scattering theory has been published [*Spurr et al*., 2012] in connection with the development of VLIDORT-based packages to retrieve aerosol microphysical properties. This work also includes a linearized Mie code formulation.

1.4 Scope of document

A theoretical description of the model is given in chapter 2 - this contains several sections summarizing the essential mathematics and the solution methods of the discrete ordinate multiple scattering radiative transfer formalism in a multi-layer medium. Some of the discrete ordinate theory may be found in the literature, and many more details are found in the papers by R. Spurr. The linearization process and the derivation of Jacobians for atmospheric and surface quantities are described in some detail, and there are treatments of exact single-scatter corrections, and sphericity corrections for the incoming solar beam and the outgoing line-of-sight.

In Chapter 3, we go over the derivation of Inherent Optical Property (IOP) input preparation. The derivation of the standard set of optical properties required for the computation of the Stokes 4-vector field is outlined, and derivations of linearized optical property inputs for the generation of atmospheric Jacobians are discussed. Also in Chapter 3, we discuss benchmarking the model against literature datasets, and there is another section on the use of performance enhancements, including the "solution saving" and "BVP telescoping" options; these are labor-saving devices designed to eliminate unnecessary computation. We also review the Fourier convergence aspects pertaining to the exact treatments of single scattering and direct beam contributions, and the use of a multiple-SZA facility for look-up table generation. Section 3.4 contains a description of Taylor series expansions used in VLIDORT for the purpose of ensuring numerical stability in certain cases where numerical artifacts could arise when solving the radiative transfer equation.

Chapter 4 describes the specifics of the VLIDORT 2.7 package. In section 4.1, we give an overview of the package; section 4.2 has a description VLIDORT's source code modules. In section 4.3, we discuss the input configuration file, "makefile" production of executables, and installation of the code. In this regard, a number of tests have been written for this release of the code, and proper installation of the package will result in the confirmation of the test data set that accompanies the release. In section 4.4, we summarize the important software standards adopted for the code and describe exception handling. This version of VLIDORT is in the public domain; copyright and licensing issues are discussed in section 4.5. Chapter 4 concludes with some acknowledgements in section 4.6. Chapter 5 contains references cited in the guide.

Appendices for VLIDORT may be found in Chapter 6. Section 6.1 has the major tables describing VLIDORT input and output Fortran 90 type-structure variables (both basic and linearized), along with a second set of tables which associate these input variables with the corresponding file-read character strings found in VLIDORT's input configuration file. Section 6.2 discusses the environment programs which not only serve as package installation tests, but also provide the user with examples of how to incorporate VLIDORT into a desired application. Section 6.3 gives a complete description of the BRDF supplement: information on how the BRDFs are constructed, the inputs and outputs of the supplement software, and descriptions of the water- and land-surface BRDFs included in the supplement. Section 6.4 gives similar

information regarding the land and water surface-leaving (SLEAVE) radiance supplement. Finally, section 6.5 has additional information which may be helpful to the user when using VLIDORT for certain applications.

2. Description of the VLIDORT model

2.1 Theoretical Framework

2.1.1 The vector RTE

A first-principles derivation of the vector RTE has been given in the analysis of Mishchenko [*Mishchenko*, 2003]. The basic vector RTE is:

$$\mu \frac{\partial}{\partial x} \mathbf{I}(x,\mu,\phi) = \mathbf{I}(x,\mu,\phi) - \mathbf{J}(x,\mu,\phi) \,. \tag{2.1.1}$$

Here, x is the optical thickness measured from the top of the layer, μ is the polar angle cosine measured from the upward vertical, and ϕ is the azimuth angle relative to some fixed direction. The 4-vector $\mathbf{I}$ is the diffuse field of Stokes components $\{I, Q, U, V\}$ [*Chandrasekhar*, 1960], with I the total intensity, Q and U describing linearly polarized radiation, and V characterizing circularly polarized radiation. Vector $\mathbf{I}$ is defined with respect to a reference plane (usually, the local meridian plane). The degree of polarization P of the radiation is:

$$P = I^{-1} \sqrt{Q^2 + U^2 + V^2} \,. \tag{2.1.2}$$

The vector source term $\mathbf{J}(x,\mu,\phi)$ has the form:

$$\mathbf{J}(x,\mu,\phi) = \frac{\omega(x)}{4\pi} \int_{-1}^{1} \int_{0}^{2\pi} \mathbf{\Pi}(x,\mu,\mu',\phi-\phi') \mathbf{I}(x,\mu',\phi') d\phi' d\mu' + \mathbf{Q}(x,\mu,\phi) \,. \tag{2.1.3}$$

Here, ω is the single scattering albedo and $\mathbf{\Pi}$ the phase matrix for scattering. The first term in Eq. (2.1.3) represents multiple scattering contributions. For scattering of the attenuated solar beam, the inhomogeneous source term $\mathbf{Q}(x,\mu,\phi)$ is written:

$$\mathbf{Q}(x,\mu,\varphi) = \frac{\omega(x)}{4\pi} \mathbf{\Pi}(x,\mu,-\mu_0,\varphi-\varphi_0) \mathbf{I}_0 T_a \exp[-\lambda x] \,. \tag{2.1.4}$$

Here, $-\mu_0$ is the cosine of the solar zenith angle (with respect to the upward vertical); ϕ_0 is the solar azimuth angle and $\mathbf{I}_0$ the Stokes vector of the incoming solar beam before attenuation.

The pseudo-spherical (P-S) beam attenuation in equation (2.1.4) is written $T_a \exp[-\lambda x]$, where T_a is the transmittance to the top of the layer, and λ is a geometrical factor (the "average secant"). In the P-S formulation, all scattering takes place in a plane-parallel medium, but the solar beam attenuation is treated for a curved atmosphere. For plane-parallel attenuation, we have $\lambda = -1/\mu_0$. It has been shown that the P-S approximation is accurate for solar zenith angles up to 90°, provided the viewing path is not too far from the nadir [*Dahlback and Stamnes*, 1991]. Details on the pseudo-spherical formulation are found in Section 2.4.1.

In the model, we consider an atmosphere illuminated by natural (unpolarized) sunlight, so that the downwelling direct solar irradiance at TOA is given by Stokes vector $\mathbf{I}_0 = \{I_0,0,0,0\}$. We assume that the medium comprises a stratification of optically uniform layers; for each layer, the single scattering albedo ω and the phase matrix $\mathbf{\Pi}$ in Eq. (2.1.3) do not depend on the optical thickness x, and we henceforth drop this dependence.

Matrix $\mathbf{\Pi}$ relates scattering and incident Stokes vectors defined with respect to the meridian plane. The equivalent matrix for Stokes vectors with respect to the *scattering* plane is the scattering matrix $\mathbf{F}$. In this work, we restrict ourselves to scattering for a medium that is "macroscopically isotropic and symmetric" [*Mishchenko et al.*, 2000], with scattering for ensembles of randomly oriented particles having at least one plane of symmetry. In this case, $\mathbf{F}$ depends only on the scattering angle Θ between scattered and incident beams. Matrix $\mathbf{\Pi}$ is related to $\mathbf{F}(\Theta)$ through application of two rotation matrices $\mathbf{L}(\pi-\sigma_2)$ and $\mathbf{L}(-\sigma_1)$ (for definitions of these matrices and the angles of rotation σ_1 and σ_2, see [*Mishchenko et al*, 2000]):

$$\mathbf{\Pi}(\mu,\phi,\mu',\phi') = \mathbf{L}(\pi - \sigma_2)\mathbf{F}(\Theta)\mathbf{L}(-\sigma_1);$$ (2.1.5)

$$\cos\Theta = \mu\mu' + \sqrt{1-\mu^2}\sqrt{1-\mu'^2}\cos(\phi-\phi').$$ (2.1.6)

In our case, $\mathbf{F}(\Theta)$ has the well-known form:

$$\mathbf{F}(\Theta) = \begin{pmatrix} a_1(\Theta) & b_1(\Theta) & 0 & 0 \\ b_1(\Theta) & a_2(\Theta) & 0 & 0 \\ 0 & 0 & a_3(\Theta) & b_2(\Theta) \\ 0 & 0 & -b_2(\Theta) & a_4(\Theta) \end{pmatrix}.$$ (2.1.7)

The upper left entry in this matrix is the phase function and satisfies the normalization condition:

$$\frac{1}{2}\int_0^\pi a_1(\Theta)\sin\Theta d\Theta = 1.$$ (2.1.8)

2.1.2 Azimuthal separation

For the special form of $\mathbf{F}$ in Eq. (2.1.7), the dependence on scattering angle allows us to develop expansions of the six independent scattering functions in terms of a set of generalized spherical functions $P_{mn}^l(\cos\Theta)$ [*Mishchenko, et al.*, 2000]:

$$a_1(\Theta) = \sum_{l=0}^{LM}\beta_l P_{00}^l(\cos\Theta);$$ (2.1.9)

$$a_2(\Theta) + a_3(\Theta) = \sum_{l=0}^{LM}(\alpha_l + \zeta_l)P_{2,2}^l(\cos\Theta);$$ (2.1.10)

$$a_2(\Theta) - a_3(\Theta) = \sum_{l=0}^{LM}(\alpha_l - \zeta_l)P_{2,-2}^l(\cos\Theta);$$ (2.1.11)

$$a_4(\Theta) = \sum_{l=0}^{LM}\delta_l P_{00}^l(\cos\Theta);$$ (2.1.12)

$$b_1(\Theta) = \sum_{l=0}^{LM}\gamma_l P_{02}^l(\cos\Theta);$$ (2.1.13)

$$b_2(\Theta) = -\sum_{l=0}^{LM}\varepsilon_l P_{02}^l(\cos\Theta).$$ (2.1.14)

The six sets of "Greek constants" $\{\alpha_l, \beta_l, \gamma_l, \delta_l, \varepsilon_l, \zeta_l\}$ must be specified for each moment l in these spherical-function expansions. The number of terms *LM* depends on the level of numerical accuracy. Values $\{\beta_l\}$ are the phase function Legendre expansion coefficients as used in the scalar RTE. These "Greek constants" are commonly used to specify the polarized-light single-scattering law, and there are a number of efficient analytical techniques for their computation, not only for spherical particles (see for example [*de Rooij and van der Stap*, 1984]) but also for randomly oriented homogeneous and inhomogeneous non-spherical particles and aggregated scatterers [*Hovenier et al.*, 2004; *Mackowski and Mishchenko*, 1996; *Mishchenko and Travis*, 1998].

With this representation in Eqs. (2.1.9) to (2.1.14), one can then develop a Fourier decomposition of $\mathbf{\Pi}$ to separate the azimuthal dependence (cosine and sine series in the relative azimuth $\phi-\phi_0$). The same separation is applied to the Stokes vector itself. A convenient formalism for this separation was developed by Siewert and co-workers [*Siewert*, 1981; *Siewert*, 1982; *Vestrucci and Siewert*, 1984], and we summarize the results here for illumination by natural light. The Stokes vector Fourier decomposition is:

$$\mathbf{I}(x,\mu,\phi) = \frac{1}{2}\sum_{l=m}^{LM}(2-\delta_{m,0})\mathbf{\Phi}^m(\phi-\phi_0)\mathbf{I}^m(x,\mu) ; \tag{2.1.15}$$

$$\mathbf{\Phi}^m(\phi) = diag\{\cos m\phi, \cos m\phi, \sin m\phi, \sin m\phi\} . \tag{2.1.16}$$

The phase matrix decomposition is:

$$\mathbf{\Pi}(\mu,\phi,\mu',\phi') = \frac{1}{2}\sum_{l=m}^{LM}(2-\delta_{m,0})\Big[\mathbf{C}^m(\mu,\mu')\cos m(\phi-\phi') + \mathbf{S}^m(\mu,\mu')\sin m(\phi-\phi')\Big]; \tag{2.1.17}$$

$$\mathbf{C}^m(\mu,\mu') = \mathbf{A}^m(\mu,\mu') + \mathbf{D}\mathbf{A}^m(\mu,\mu')\mathbf{D} ; \tag{2.1.18}$$

$$\mathbf{S}^m(\mu,\mu') = \mathbf{A}^m(\mu,\mu')\mathbf{D} - \mathbf{D}\mathbf{A}^m(\mu,\mu') ; \tag{2.1.19}$$

$$A^m(\mu,\mu') = \sum_{l=m}^{LM}\mathbf{P}_l^m(\mu)\mathbf{B}_l\mathbf{P}_l^m(\mu') ; \tag{2.1.20}$$

$$\mathbf{D} = diag\{1,1,-1,-1\} . \tag{2.1.21}$$

This yields the following RTE for the Fourier component:

$$\mu\frac{d\mathbf{I}^m(x,\mu)}{dx} + \mathbf{I}^m(x,\mu) = \frac{\omega}{2}\sum_{l=m}^{LM}\mathbf{P}_l^m(\mu)\mathbf{B}_l\int_{-1}^{1}\mathbf{P}_l^m(\mu')\mathbf{I}^m(x,\mu')d\mu' + \mathbf{Q}^m(x,\mu) . \tag{2.1.22}$$

Here, the source term is written:

$$\mathbf{Q}^m(x,\mu) = \frac{\omega}{2}\sum_{l=m}^{LM}\mathbf{P}_l^m(\mu)\mathbf{B}_l\mathbf{P}_l^m(-\mu_0)\mathbf{I}_0 T_a e^{-\lambda x} . \tag{2.1.23}$$

The phase matrix expansion is expressed through the two matrices:

$$\mathbf{B}_l = \begin{pmatrix} \beta_l & \gamma_l & 0 & 0 \\ \gamma_l & \alpha_l & 0 & 0 \\ 0 & 0 & \varsigma_l & -\varepsilon_l \\ 0 & 0 & \varepsilon_l & \delta_l \end{pmatrix}; \tag{2.1.24}$$

$$\mathbf{P}_l^m(\mu) = \begin{pmatrix} P_l^m(\mu) & 0 & 0 & 0 \\ 0 & R_l^m(\mu) & -T_l^m(\mu) & 0 \\ 0 & -T_l^m(\mu) & R_l^m(\mu) & 0 \\ 0 & 0 & 0 & P_l^m(\mu) \end{pmatrix}. \tag{2.1.25}$$

The "Greek matrices" $\mathbf{B}_l$ for $0 \le l \le LM$ contain the sets of expansion coefficients that define the scattering law. The $\mathbf{P}_l^m(\mu)$ matrices contain entries of normalized Legendre functions $P_l^m(\mu)$ and functions $R_l^m(\mu)$ and $T_l^m(\mu)$ which are related to $P_{mn}^l(\mu)$ (for details, see for example [*Siewert*, 2000b]).

2.1.3 Boundary conditions

Discrete ordinate RT is pure scattering theory: in a multilayer medium, it is only necessary to specify the layer total optical thickness values Δ_n, the layer total single scatter albedo ω_n, and the layer 4 x 4 matrices $\mathbf{B}_{nl}$ of expansion coefficients (*l* being the moment number) for the total scattering. To complete the calculation of the radiation field in a stratified multilayer medium, we have the following boundary conditions:

   (I)   No diffuse downwelling radiation at TOA. Thus for the first layer we have:

$$\mathbf{I}_n^+(0, \mu, \phi) = 0. \qquad\qquad (n = 1) \tag{2.1.26}$$

   (II)   Continuity of the upwelling and downwelling radiation fields at intermediate boundaries. If N_{TOTAL} is the number of layers in the medium, then:

$$\mathbf{I}_{n-1}^\pm(\Delta_{n-1}) = \mathbf{I}_n^\pm(0). \qquad\qquad (n = 2, \dots N_{TOTAL}) \tag{2.1.27}$$

   (III)   A surface reflection condition relating the upwelling and downwelling radiation fields at the bottom of the atmosphere:

$$\mathbf{I}_n^-(\Delta_n, \mu, \phi) = \mathbf{R}(\mu, \phi; \mu', \phi') \mathbf{I}_n^+(\Delta_n, \mu', \phi'). \quad (n = N_{TOTAL}) \tag{2.1.28}$$

   Here, reflection matrix $\mathbf{R}$ relates incident and reflected directions.

The convention adopted here is to use a "+" suffix for downwelling solutions, and a "–" suffix for upwelling radiation. Conditions (I) and (II) are obeyed by all Fourier components in the azimuthal series. For condition (III), it is necessary to construct a Fourier decomposition of the BRDF operator $\mathbf{R}$ to separate the azimuth dependence; we return to this issue in section 2.5.4. The Lambertian case (isotropic reflectance) only applies for Fourier component $m = 0$ and Eq. (2.1.28) then becomes:

$$\mathbf{I}_n^-(\Delta_n, \mu) = 2\delta_{m,0} R_0 \mathbf{E}_1 \left[\mu_0 \mathbf{I}_0 T_{n-1} \exp(-\lambda_n \Delta_n) + \int_0^1 \mathbf{I}_n^+(\Delta_n, \mu') \mu' d\mu' \right]. \tag{2.1.29}$$

Here, R_0 is the Lambertian albedo, $\mathbf{E}_1 = diag\{1,0,0,0\}$, and $T_{n-1}\exp(-\lambda_n\Delta_n)$ is the whole-atmosphere slant path optical depth for the solar beam.

2.1.4 Jacobian definitions

Atmospheric Jacobians (also known as weighting functions) are *normalized analytic derivatives* of the Stokes vector field with respect to any atmospheric property ξ_n defined in layer n:

$$\mathbf{K}_\xi(x,\mu,\phi) = \xi\frac{\partial\mathbf{I}(x,\mu,\phi)}{\partial\xi}. \tag{2.1.30}$$

The Fourier series azimuth dependence (c.f. Eq. (2.1.15)) is also valid:

$$\mathbf{K}_\xi(x,\mu,\phi) = \frac{1}{2}\sum_{l=m}^{LM}(2-\delta_{m,0})\mathbf{C}^m(\phi-\phi_0)\mathbf{K}_\xi^m(x,\mu). \tag{2.1.31}$$

We use the linearization notation:

$$\mathcal{L}_p(y_n) = \xi_p\frac{\partial y_n}{\partial\xi_p}, \tag{2.1.32}$$

to indicate the normalized derivative of y_n in layer n with respect to variable ξ_p in layer p.

As noted in section 2.1.3, for the radiation field, input optical properties are $\{\Delta_n, \omega_n, \mathbf{B}_{nl}\}$ for each layer n in a multilayer medium. For Jacobians, we require an additional set of *linearized optical property inputs* $\{\mathcal{V}_n, \mathcal{U}_n, \mathcal{Z}_{nl}\}$ defined with respect to variable ξ_n in layer n for which we require weighting functions. These are:

$$\mathcal{V}_n \equiv \mathcal{L}_n(\Delta_n); \qquad \mathcal{U}_n \equiv \mathcal{L}_n(\omega_n); \qquad \mathcal{Z}_{nl} \equiv \mathcal{L}_n(\mathbf{B}_{nl}). \tag{2.33}$$

In section 3.2 we give an example of the construction of the input set $\{\Delta_n, \omega_n, \mathbf{B}_{nl}\}$ and its linearizations $\{\mathcal{V}_n, \mathcal{U}_n, \mathcal{Z}_{nl}\}$ for a typical atmospheric scenario with molecular and aerosol scattering. One can also define weighting functions with respect to basic optical properties themselves: for example, if $\xi_n = \Delta_n$, then $\mathcal{V}_n \equiv \mathcal{L}_n(\Delta_n) = \Delta_n$.

For surface weighting functions, we need to know how the BRDF matrix operator $\mathbf{R}$ in Eq. (2.1.28) is parameterized. In VLIDORT, we have adopted a 3-kernel BRDF formulation of surface reflectance similar to the scheme developed in [*Spurr*, 2003] for LIDORT. In section 2.3, we confine our attention to the Lambertian case; BRDF implementation is discussed in the appendices.

2.1.5 Solution strategy

The solution strategy has two stages. First, for each layer, we establish discrete ordinate solutions to the homogeneous RTE (in the absence of sources) and to the RTE with solar and thermal source term (sections 2.2-2.4). Second, we complete the solution by application of boundary conditions and by source function integration of the RTE in order to establish solutions away from discrete ordinate directions (section 2.5). Additional implementations are discussed in sections 2.6 through 2.8. These include the pseudo-spherical approximation and outgoing

sphericity and exact single scatter corrections (2.6) and the implementation for total atmosphere and blackbody weighting functions (sections 2.7 and 2.8).

The complete vector RT solution for a plane-parallel slab was developed by Siewert [*Siewert*, 2000b], and we follow some elements in this formulation. Our description also adheres closely to the LIDORT treatment, especially concerning this particular integral solution, formulation of the boundary-value problem and linearization methodology.

In the following sections, we suppress the Fourier index m unless noted explicitly, and wavelength dependence is implicit throughout. We sometimes suppress the layer index n in the interests of clarity. For matrix notation, ordinary 4 x 1 vectors and 4 x 4 matrices are written in bold typeface, while $4N$ x 1 vectors and $4N$ x $4N$ matrices are written in bold typeface with a tilde symbol (N is the number of discrete ordinate directions in the half-space).

2.2 Discrete Ordinate Solutions and Linearizations

2.2.1 Homogeneous RTE, Eigenproblem reduction

First, we solve Eq. (2.1.22) without the solar source term. For each Fourier term m, the multiple scatter integral over the upper and lower polar direction half-spaces is approximated by a double Gaussian quadrature scheme [*Thomas and Stamnes*, 1999], with stream directions $\{\pm\mu_i\}$ and Gauss-Legendre weights $\{w_i\}$ for $i = 1,...N$. The resulting vector RTE for Fourier component m is then:

$$\pm\mu_i\frac{d\mathbf{I}_i^\pm(x)}{dx}\pm\mathbf{I}_i^\pm(x) = \frac{\omega_n}{2}\sum_{l=m}^{LM}\mathbf{P}_l^m(\pm\mu_i)\mathbf{B}_l\sum_{j=1}^{N}w_j\left\{\mathbf{I}_j^+(x)\mathbf{P}_l^m(\mu_j)+\mathbf{I}_j^-(x)\mathbf{P}_l^m(-\mu_j)\right\}.$$ (2.2.1)

Eq. (2.2.1) is a set of $8N$ coupled first-order linear differential equations for $\mathbf{I}_i^\pm(x)$. As with the scalar case, these are solved by eigenvalue methods. We follow [*Siewert*, 2000b] for the most part. Solutions for these homogeneous equations are found with the *ansatz*:

$$\mathbf{I}_\alpha^\pm(x,\pm\mu_i) = \mathbf{W}_\alpha(\pm\mu_i)\exp[-k_\alpha x].$$ (2.2.2)

We define the ($4N$ x 1) vector (superscript "T" denotes matrix transpose):

$$\tilde{\mathbf{W}}_\alpha^\pm = \left[\mathbf{W}_\alpha^{\mathrm{T}}(\pm\mu_1), \mathbf{W}_\alpha^{\mathrm{T}}(\pm\mu_2),...., \mathbf{W}_\alpha^{\mathrm{T}}(\pm\mu_N)\right]^{\mathrm{T}}.$$ (2.2.3)

Equations (2.2.1) are decoupled using $\tilde{\mathbf{X}}_\alpha = \tilde{\mathbf{W}}_\alpha^+ + \tilde{\mathbf{W}}_\alpha^-$ and $\tilde{\mathbf{Y}}_\alpha = \tilde{\mathbf{W}}_\alpha^+ - \tilde{\mathbf{W}}_\alpha^-$ (sum and difference vectors), and the order of the system can then be reduced from $8N$ to $4N$. This gives an eigenproblem for the collection of separation constants $\{k_\alpha\}$ and associated solution $4N$-vectors { $\tilde{\mathbf{X}}_\alpha$ }, where $\alpha = 1,...4N$. The eigenmatrix $\tilde{\Gamma}$ is constructed from optical property inputs ω and $\mathbf{B}_l$ and products of the matrices $\mathbf{P}_l^m(\mu_j)$. The eigenproblem is:

$$\tilde{\mathbf{X}}_\alpha^\perp\tilde{\Gamma} = k_\alpha^2\tilde{\mathbf{X}}_\alpha^\perp ; \qquad \tilde{\Gamma}\tilde{\mathbf{X}}_\alpha = k_\alpha^2\tilde{\mathbf{X}}_\alpha ;$$ (2.2.4)

$$\tilde{\Gamma} = \tilde{\mathbf{S}}^+\tilde{\mathbf{S}}^- ;$$ (2.2.5)

$$\tilde{\mathbf{S}}^\pm = \left[\tilde{\mathbf{E}} - \frac{\omega}{2}\sum_{l=m}^{LM}\tilde{\Pi}(l,m)\mathbf{B}_l\mathbf{A}^\pm\tilde{\Pi}^{\mathrm{T}}(l,m)\tilde{\Omega}\right]\tilde{\mathbf{M}}^{-1} ;$$ (2.2.6)

$$\tilde{\mathbf{\Pi}}(l,m) = diag\left[\mathbf{P}_l^m(\mu_1), \mathbf{P}_l^m(\mu_2), \ldots, \mathbf{P}_l^m(\mu_N)\right]^{\mathrm{T}};$$ (2.2.7)

$$\tilde{\mathbf{M}} = diag\left[\mu_1 \mathbf{E}, \mu_2 \mathbf{E}, \ldots, \mu_N \mathbf{E}\right];$$ (2.2.8)

$$\tilde{\mathbf{\Omega}} = diag\left[w_1 \mathbf{E}, w_2 \mathbf{E}, \ldots, w_N \mathbf{E}\right];$$ (2.2.9)

$$\mathbf{A}^{\pm} = \mathbf{E} \pm (-1)^{l-m} \mathbf{D}.$$ (2.2.10)

Here, $\mathbf{E}$ is the 4 x 4 identity matrix, and $\tilde{\mathbf{E}}$ the $4N$ x $4N$ identity matrix. The ($\perp$) superscript indicates the conjugate transpose. The link between the eigenvector $\tilde{\mathbf{X}}_\alpha$ and the solution vectors in Eq. (2.2.2) is through the auxiliary equations:

$$\tilde{\mathbf{W}}_\alpha^{\pm} = \frac{1}{2}\tilde{\mathbf{M}}^{-1}\left[\tilde{\mathbf{E}} \pm \frac{1}{k_\alpha}\tilde{\mathbf{S}}^+\right]\tilde{\mathbf{X}}_\alpha.$$ (2.2.11)

Eigenvalues occur in pairs $\{\pm k_\alpha\}$. As noted by Siewert [*Siewert*, 2000b], both complex variable and real-variable eigensolutions may be present. Left and right eigenvectors share the same spectrum of eigenvalues. Solutions may be determined with the complex-variable eigensolver DGEEV from the LAPACK suite [*Anderson, et al.*, 1995]. DGEEV returns eigenvalues plus left- and right-eigenvectors with unit modulus.

In the scalar case, the formulation of the eigenproblem is simpler (see [*Spurr*, 2002] for example). The eigenmatrix is symmetric and all eigensolutions are real-valued. In this case, the eigensolver module ASYMTX [*Stamnes et al.*, 1988] is used. ASYMTX is a modification of the LAPACK routine for real roots; it delivers only the right eigenvectors. For the vector case, there are circumstances (pure Rayleigh scattering for example) where complex eigensolutions are absent, and one may then use the faster ASYMTX routine. We return to this point in section 3.4.3.

The complete homogeneous solution in one layer is a linear combination of all positive and negative eigensolutions:

$$\tilde{\mathbf{I}}_+(x) = \tilde{\mathbf{D}}^+ \sum_{\alpha=1}^{4N}\left\{L_\alpha \tilde{\mathbf{W}}_\alpha^+ \exp[-k_\alpha x] + M_\alpha \tilde{\mathbf{W}}_\alpha^- \exp[-k_\alpha(\Delta - x)]\right\};$$ (2.2.12)

$$\tilde{\mathbf{I}}_-(x) = \tilde{\mathbf{D}}^- \sum_{\alpha=1}^{4N}\left\{L_\alpha \tilde{\mathbf{W}}_\alpha^- \exp[-k_\alpha x] + M_\alpha \tilde{\mathbf{W}}_\alpha^+ \exp[-k_\alpha(\Delta - x)]\right\}.$$ (2.2.13)

Here, $\tilde{\mathbf{D}}^- = diag\{\mathbf{D}, \mathbf{D}, \ldots, \mathbf{D}\}$ and $\tilde{\mathbf{D}}^+ = \tilde{\mathbf{E}}$. The use of optical thickness $\Delta - x$ in the second exponential ensures that solutions remain bounded [*Stamnes and Conklin*, 1984]. The quantities $\{L_\alpha, M_\alpha\}$ are the constants of integration, and must be determined by the boundary conditions.

In equations (2.2.12) and (2.2.13), some eigensolutions will be complex, some real. It is understood that when we use these expressions in the boundary value problem (section 2.3.1), we compute the real parts of any contributions to the Stokes vectors resulting from complex eigensolutions. Thus if $\{k_\alpha, \tilde{\mathbf{W}}_\alpha^+\}$ is a complex solution with (complex) integration constant L_α, we require:

$$\mathrm{Re}[L_\alpha \tilde{\mathbf{W}}_\alpha^- e^{-k_\alpha x}] = \mathrm{Re}[L_\alpha]\mathrm{Re}[\tilde{\mathbf{W}}_\alpha^- e^{-k_\alpha x}] - \mathrm{Im}[L_\alpha]\mathrm{Im}[\tilde{\mathbf{W}}_\alpha^- e^{-k_\alpha x}].$$ (2.2.14)

From a bookkeeping standpoint, one must keep count of the number of real and complex solutions, and treat them separately in the numerical implementation. In the interests of clarity, we have not made an explicit separation of complex variables, and it will be clear from the context whether real or complex variables are under consideration.

2.2.2 Linearization of the eigenproblem

We require derivatives of the above eigenvectors and separation constants with respect to some *atmospheric* variable ξ in layer *n*. From (2.2.5) and (2.2.6), the eigenmatrix $\tilde{\mathbf{\Gamma}}$ is a linear function of the single scatter albedo ω and the matrix of expansion coefficients $\mathbf{B}_l$, and its (real-variable) linearization $\mathcal{L}(\tilde{\mathbf{\Gamma}})$ is easy to establish from chain-rule differentiation:

$$\mathcal{L}(\tilde{\mathbf{\Gamma}}) = \mathcal{L}(\tilde{\mathbf{S}}^+)\tilde{\mathbf{S}}^- + \tilde{\mathbf{S}}^+ \mathcal{L}(\tilde{\mathbf{S}}^-) ; \tag{2.2.15}$$

$$\mathcal{L}(\tilde{\mathbf{S}}^\pm) = \left[\sum_{l=m}^{LM} \left\{ \frac{\mathcal{L}(\omega)}{2} \tilde{\mathbf{\Pi}}(l,m)\mathbf{B}_l + \frac{\omega}{2} \tilde{\mathbf{\Pi}}(l,m)\mathcal{L}(\mathbf{B}_l) \right\} \mathbf{A}^\pm \tilde{\mathbf{\Pi}}^{\mathrm{T}}(l,m)\tilde{\mathbf{\Omega}} \right] \tilde{\mathbf{M}}^{-1} . \tag{2.2.16}$$

In Eq. (2.2.16), $\mathcal{L}(\omega) = \mathcal{U}$ and $\mathcal{L}(\mathbf{B}_l) = \mathcal{Z}_l$ are the linearized optical property inputs (Eq. (2.1.33)). Next, we differentiate both the left and right eigensystems (2.2.4) to find:

$$\mathcal{L}(\tilde{\mathbf{X}}_\alpha^\perp)\tilde{\mathbf{\Gamma}} + \tilde{\mathbf{X}}_\alpha^\perp \mathcal{L}(\tilde{\mathbf{\Gamma}}) = 2k_\alpha \mathcal{L}(k_\alpha)\tilde{\mathbf{X}}_\alpha^\perp + k_\alpha^2 \mathcal{L}(\tilde{\mathbf{X}}_\alpha^\perp) ; \tag{2.2.17}$$

$$\tilde{\mathbf{\Gamma}}\mathcal{L}(\tilde{\mathbf{X}}_\alpha) + \mathcal{L}(\tilde{\mathbf{\Gamma}})\tilde{\mathbf{X}}_\alpha = 2k_\alpha \mathcal{L}(k_\alpha)\tilde{\mathbf{X}}_\alpha + k_\alpha^2 \mathcal{L}(\tilde{\mathbf{X}}_\alpha) . \tag{2.2.18}$$

We form a dot product by pre-multiplying (2.2.18) with the transpose vector $\tilde{\mathbf{X}}_\alpha^\perp$, rearranging to get:

$$2k_\alpha \mathcal{L}(k_\alpha)\langle \tilde{\mathbf{X}}_\alpha^\perp, \tilde{\mathbf{X}}_\alpha \rangle - \langle \tilde{\mathbf{X}}_\alpha^\perp, \mathcal{L}(\tilde{\mathbf{\Gamma}})\tilde{\mathbf{X}}_\alpha \rangle = k_\alpha^2 \langle \tilde{\mathbf{X}}_\alpha^\perp, \mathcal{L}(\tilde{\mathbf{X}}_\alpha) \rangle - \langle \tilde{\mathbf{X}}_\alpha^\perp, \tilde{\mathbf{\Gamma}}\mathcal{L}(\tilde{\mathbf{X}}_\alpha) \rangle . \tag{2.2.19}$$

From the definitions in Eq. (2.2.4), we have:

$$\langle \tilde{\mathbf{X}}_\alpha^\perp, \tilde{\mathbf{\Gamma}}\mathcal{L}(\tilde{\mathbf{X}}_\alpha) \rangle = \langle \tilde{\mathbf{X}}_\alpha^\perp \tilde{\mathbf{\Gamma}}, \mathcal{L}(\tilde{\mathbf{X}}_\alpha) \rangle = k_\alpha^2 \langle \tilde{\mathbf{X}}_\alpha^\perp, \mathcal{L}(\tilde{\mathbf{X}}_\alpha) \rangle , \tag{2.2.20}$$

and hence the right hand side of (2.2.19) is identically zero. We thus have:

$$\mathcal{L}(k_\alpha) = \frac{\langle \tilde{\mathbf{X}}_\alpha^\perp, \mathcal{L}(\tilde{\mathbf{\Gamma}})\tilde{\mathbf{X}}_\alpha \rangle}{2k_\alpha \langle \tilde{\mathbf{X}}_\alpha^\perp, \tilde{\mathbf{X}}_\alpha \rangle} . \tag{2.2.21}$$

Next, we substitute Eq. (2.2.21) in (2.2.19) to obtain the following 4*N* x 4*N* linear algebra problem for each eigensolution linearization:

$$\tilde{\mathbf{H}}_\alpha \mathcal{L}(\tilde{\mathbf{X}}_\alpha) = \tilde{\mathbf{C}}_\alpha ; \tag{2.2.22}$$

$$\tilde{\mathbf{H}}_\alpha = \tilde{\mathbf{\Gamma}} - k_\alpha^2 \tilde{\mathbf{E}} ; \tag{2.2.23}$$

$$\tilde{\mathbf{C}}_\alpha = 2k_\alpha \mathcal{L}(k_\alpha)\tilde{\mathbf{X}}_\alpha - \mathcal{L}(\tilde{\mathbf{\Gamma}})\tilde{\mathbf{X}}_\alpha . \tag{2.2.24}$$

Implementation of Eq. (2.2.22) "as is" is not possible due to the degeneracy of the eigenproblem, and we need additional constraints to find the unique solution for $\mathcal{L}(\tilde{\mathbf{X}}_\alpha)$. The treatment for real and complex solutions is different.

Real solutions. The unit-modulus eigenvector normalization can be expressed as $\langle \tilde{\mathbf{X}}_\alpha, \tilde{\mathbf{X}}_\alpha \rangle = 1$ in dot-product notation. Linearizing, this yields one equation:

$$\mathcal{L}(\tilde{\mathbf{X}}_\alpha)\tilde{\mathbf{X}}_\alpha + \tilde{\mathbf{X}}_\alpha\mathcal{L}(\tilde{\mathbf{X}}_\alpha) = 0. \tag{2.2.25}$$

The solution procedure uses $4N-1$ equations from (2.2.22), along with Eq. (2.2.25) to form a slightly modified linear system of rank $4N$. This system is then solved by standard means using the DGETRF and DGETRS LU-decomposition routines from the LAPACK suite.

This procedure was not used in the scalar LIDORT code [*Spurr et al.*, 2001; *Spurr*, 2002]. This is because ASYMTX has no adjoint solution, so there is no determination of $\mathcal{L}(k_\alpha)$ as in Eq. (2.2.21). Instead, LIDORT uses the complete set (2.2.22) *in addition* to the constraint (2.2.25) to form a system of rank $N+1$ for the unknowns $\mathcal{L}(k_\alpha)$ and $\mathcal{L}(\tilde{\mathbf{X}}_\alpha)$.

Complex solutions. In this case, Eq. (2.2.22) is a complex-variable system for both the real and imaginary parts of the linearized eigenvectors. There are $8N$ equations in all, but now we require two constraint conditions to remove the eigenproblem arbitrariness. The first is Eq. (2.2.25). The second condition is imposed by the following DGEEV normalization: for that element of an eigenvector with the largest real value, the corresponding imaginary part is always set to zero. Thus for an eigenvector $\tilde{\mathbf{X}}$, if element $\mathrm{Re}[X_J] = \max\{\mathrm{Re}[X_j]\}$ for $j = 1,\dots 4N$, then $\mathrm{Im}[X_J] = 0$. In this case, it is also true that $\mathcal{L}(\mathrm{Im}[X_J]) = 0$. This is the second condition.

The solution procedure is then (1) in Eq. (2.2.22) to strike out the row and column J in matrix $\tilde{\mathbf{H}}_\alpha$ for which the quantity $\mathrm{Im}[X_J]$ is zero, and strike out the corresponding row in the right-hand vector $\tilde{\mathbf{C}}_\alpha$; and (2) in the resulting $8N-1$ system, replace one of the rows with the normalization constraint Eq. (2.2.25). $\mathcal{L}(\tilde{\mathbf{X}}_\alpha)$ is then the solution of the resulting linear system.

We have gone into detail here, as the above procedure for eigensolution differentiation is the most crucial step in the linearization process, and there are several points of departure from the equivalent procedure in the scalar case. Having derived the linearizations $\mathcal{L}(k_\alpha)$ and $\mathcal{L}(\tilde{\mathbf{X}}_\alpha)$, we complete this section by differentiating the auxiliary result in Eq. (2.2.11) to establish $\mathcal{L}(\tilde{\mathbf{W}}_\alpha^\pm)$:

$$\mathcal{L}(\tilde{\mathbf{W}}_\alpha^\pm) = \frac{1}{2}\tilde{\mathbf{M}}^{-1}\left[\mp\frac{\mathcal{L}(k_\alpha)}{k_\alpha^2}\tilde{\mathbf{S}}^+ \pm \frac{1}{k_\alpha}\mathcal{L}(\tilde{\mathbf{S}}^+)\right]\tilde{\mathbf{X}}_\alpha + \frac{1}{2}\tilde{\mathbf{M}}^{-1}\left[\tilde{\mathbf{E}} \pm \frac{1}{k_\alpha}\tilde{\mathbf{S}}^+\right]\mathcal{L}(\tilde{\mathbf{X}}_\alpha). \tag{2.2.26}$$

Finally, we have linearizations of the transmittance derivatives in Eqs. (2.2.12) and (2.2.13):

$$\mathcal{L}(\exp[-k_\alpha x]) = -x\{\mathcal{L}(k_\alpha) + k_\alpha\mathcal{L}(x)\}\exp[-k_\alpha x]. \tag{2.2.27}$$

Here, x and Δ_n are proportional for an optically uniform layer, so that

$$\mathcal{L}_\xi(x) = \frac{x}{\Delta_n}\mathcal{L}_\xi(\Delta_n) = \frac{x}{\Delta_n}\mathcal{V}_\xi. \tag{2.2.28}$$

2.2.3 Particular Integral of the vector RTE, solar term

Solving the RTE by substitution

In the treatment of the particular integral solutions of the vector RTE, we use a more traditional substitution method rather than the Green's function formalism of Siewert [*Siewert*, 2000b]. This

is mainly for reasons of clarity and ease of exposition. Referring to Eq. (2.1.23), inhomogeneous source terms in the discrete ordinate directions are:

$$\mathbf{Q}_n^m(x,\pm\mu_i) = \frac{\omega}{2}\sum_{l=m}^{L}\mathbf{P}_l^m(\pm\mu_i)\mathbf{B}_{nl}\mathbf{P}_l^m(-\mu_0)\mathbf{I}_0 T_{n-1}\exp(-\lambda_n x) \, . \tag{2.2.29}$$

Here T_{n-1} is the solar beam transmittance to the top of layer n, and in the pseudo-spherical approximation, λ_n is the average secant (section 2.4.1). Particular solutions may be found by substitution:

$$\mathbf{I}^{\pm}(x,\pm\mu_i) = \mathbf{Z}_n(\pm\mu_i)T_{n-1}\exp[-\lambda_n x] \, , \tag{2.2.30}$$

and by analogy with the homogeneous case, we define the 4*N* x 1 vectors:

$$\widetilde{\mathbf{Z}}_n^{\pm} = \left[\mathbf{Z}_n^{\mathrm{T}}(\pm\mu_1), \mathbf{Z}_n^{\mathrm{T}}(\pm\mu_2),....,\mathbf{Z}_n^{\mathrm{T}}(\pm\mu_N)\right]^{\mathrm{T}} \, . \tag{2.2.31}$$

We decouple the resulting equations by using sum and difference vectors $\widetilde{\mathbf{G}}_n^{\pm} = \widetilde{\mathbf{Z}}_n^{+} \pm \widetilde{\mathbf{Z}}_n^{-}$, and reduce the order from 8*N* to 4*N* (see [*Van Oss and Spurr*, 2002] for the scalar case). We obtain the following 4*N* x 4*N* linear-algebra problem:

$$\widetilde{\mathbf{A}}_n^{(2)}\widetilde{\mathbf{G}}_n^{+} = \widetilde{\mathbf{C}}_n^{(2)} \, ; \tag{2.2.32}$$

$$\widetilde{\mathbf{A}}_n^{(2)} = \lambda_n^2\widetilde{\mathbf{E}} - \widetilde{\mathbf{\Gamma}}_n; \quad \widetilde{\mathbf{C}}_n^{(2)} = \left[\widetilde{\mathbf{S}}_n^{-}\widetilde{\mathbf{Q}}_n^{+} + \lambda_n\widetilde{\mathbf{Q}}_n^{-}\right]\widetilde{\mathbf{M}}^{-1} \, ; \tag{2.2.33}$$

$$\widetilde{\mathbf{Q}}_n^{\pm} = \omega\sum_{l=m}^{LM}\widetilde{\mathbf{\Pi}}_0(l,m)\mathbf{B}_l\mathbf{A}^{\pm}\widetilde{\mathbf{\Pi}}^{\mathrm{T}}(l,m)\widetilde{\mathbf{M}}^{-1} \, ; \tag{2.2.34}$$

Here, $\widetilde{\mathbf{\Pi}}_0(l,m)$ is defined as in Eq. (2.2.7) but for matrices $\mathbf{P}_l^m(-\mu_0)$. This system (2.2.32-2.2.34) has some similarities to the eigensolution linearization in equations (2.2.22-2.2.25). It is also solved using the LU-decomposition modules DGETRF and DGETRS from LAPACK; the formal solution is $\widetilde{\mathbf{G}}_n^{+} = \left[\widetilde{\mathbf{A}}_n^{(2)}\right]^{-1}\widetilde{\mathbf{C}}_n^{(2)}$. The particular integral is completed through the auxiliary equations:

$$\widetilde{\mathbf{Z}}_n^{\pm} = \frac{1}{2}\widetilde{\mathbf{M}}^{-1}\left[\widetilde{\mathbf{E}} \pm \frac{1}{\lambda_n}\widetilde{\mathbf{S}}_n^{+}\right]\widetilde{\mathbf{G}}_n^{+} \, . \tag{2.2.35}$$

We note that the particular solution consists only of real variables.

Linearizing the particular solution

For the linearization, the most important point is the presence of cross-derivatives: the particular solution is differentiable with respect to atmospheric variables ξ_p in all layers $p \geq n$. The solar beam has passed through layer $p \geq n$ before scattering, so transmittance factor T_{n-1} depends on variables in layers $p > n$ and the average secant λ_n (in the pseudo-spherical approximation) on variables ξ_p for $p \geq n$ In addition, the solution vectors $\widetilde{\mathbf{Z}}_n^{\pm}$ depend on λ_n, so *their* linearizations contain cross-derivatives.

Linearization of the pseudo-spherical approximation is treated in Appendix A, and this fixes the quantities $\mathcal{L}_p(T_{n-1})$ and $\mathcal{L}_p(\lambda_n) \; \forall \; p \geq n$. For the plane-parallel case, $\mathcal{L}_p(\lambda_n) \equiv 0$ since $\lambda_n = -1/\mu_0$ (constant). In addition, the eigenmatrix $\widetilde{\mathbf{\Gamma}}_n$ is constructed from optical properties only defined in

layer n, so that $\mathcal{L}_p(\tilde{\mathbf{\Gamma}}_n) = 0 \ \forall \ p \neq n$. Differentiation of Eqs. (2.2.32-2.2.34) yields a related linear problem:

$$\tilde{\mathbf{A}}_n^{(2)}\mathcal{L}_p(\tilde{\mathbf{G}}_n^+) \equiv \tilde{\mathbf{C}}_{np}^{(3)} = \mathcal{L}_p(\tilde{\mathbf{C}}_n^{(2)}) - \mathcal{L}_p(\tilde{\mathbf{A}}_n^{(2)})\tilde{\mathbf{G}}_n^+ \ ; \tag{2.2.36}$$

$$\mathcal{L}_p(\tilde{\mathbf{A}}_n^{(2)}) = -\delta_{pn}\mathcal{L}_p(\tilde{\mathbf{\Gamma}}_n) + 2\lambda_n\mathcal{L}_p(\lambda_n)\tilde{\mathbf{E}};$$

$$\mathcal{L}_p(\tilde{\mathbf{C}}_n^{(2)}) = \delta_{np}\left[\mathcal{L}_n(\tilde{\mathbf{S}}_n^-)\tilde{\mathbf{Q}}_n^+ + \tilde{\mathbf{S}}_n^-\mathcal{L}_n(\tilde{\mathbf{Q}}_n^+) + \frac{1}{\lambda_n}\mathcal{L}_n(\tilde{\mathbf{Q}}_n^-)\right] - \frac{\mathcal{L}_p(\lambda_n)}{\lambda_n^2}\tilde{\mathbf{Q}}_n^- ; \tag{2.2.37}$$

$$\mathcal{L}_n(\tilde{\mathbf{Q}}_n^\pm) = \sum_{l=m}^{LM}\left[\mathcal{U}_n\tilde{\mathbf{\Pi}}_0(l,m)\mathbf{B}_l + \omega_n\tilde{\mathbf{\Pi}}_0(l,m)Z_{nl}\right]\mathbf{A}^\pm\tilde{\mathbf{\Pi}}^{\mathrm{T}}(l,m)\tilde{\mathbf{M}}^{-1}. \tag{2.2.38}$$

In Eq. (2.2.37), the quantity $\mathcal{L}_n(\tilde{\mathbf{S}}_n^-)$ comes from (2.2.16). Equation (2.2.36) has the same matrix $\tilde{\mathbf{A}}_n^{(2)}$ as in Eq. (2.2.32), but with a different source vector on the right hand side. The solution is then found by back-substitution, given that the inverse of the matrix $\tilde{\mathbf{A}}_n^{(2)}$ has already been established for the original solution $\tilde{\mathbf{G}}_n^+$. Thus $\mathcal{L}_p(\tilde{\mathbf{G}}_n^+) = \left[\tilde{\mathbf{A}}_n^{(2)}\right]^{-1}\tilde{\mathbf{C}}_{np}^{(3)}$. Linearization of the particular integral is then completed through differentiation of the auxiliary equations (2.2.35):

$$\mathcal{L}_p(\tilde{\mathbf{Z}}_n^\pm) = \frac{1}{2}\tilde{\mathbf{M}}^{-1}\left[\tilde{\mathbf{E}} \pm \frac{1}{\lambda_n}\tilde{\mathbf{S}}_n^+\right]\mathcal{L}_p(\tilde{\mathbf{G}}_n^+) \mp \frac{1}{2\lambda_n^2}\tilde{\mathbf{M}}^{-1}\left[\lambda_n\delta_{pn}\mathcal{L}_p(\tilde{\mathbf{S}}_n^+) - \mathcal{L}_p(\lambda_n)\tilde{\mathbf{S}}_n^+\right]\tilde{\mathbf{G}}_n^+. \tag{2.2.39}$$

This completes the RTE solution determination and the corresponding linearizations with respect to atmospheric variables.

2.2.4 Particular Integral of the vector RTE, thermal emission

In this section, we determine solution of the RTE in the presence of atmospheric thermal emission sources. This formalism is based on the substitution approach used in the original LIDORT work [*Spurr et al.*, 2001] and in the DISORT formalism [*Stamnes et al.*, 1988], but with a newly worked out reduction in the order of the corresponding linear algebra system. We also present a linearization of this solution with respect to the atmospheric profile variables. [Linearization with respect to the Black Body temperatures themselves is another story, and is currently being worked on].

The source is now isotropic thermal emission, with amplitude is equal to $q_n(x) = (1 - \omega_n)\eta_n(x)$, where $\eta_n(x)$ is the Black Body Planck function expressed as a function of vertical optical thickness within layer n. The phase function for scattering is 1, and the thermal term is only present for the azimuthal series term $m = 0$. There is no polarization, so we deal with only the (1,1) component of the phase matrix.

In order to obtain solutions, the Planck function is expressed as a polynomial in x across the layer. For convenience, we assume the linear form $\eta_n(x) = a_n + b_n x$. Then, the thermal emission is piecewise continuous through the whole atmosphere and may be completely specified by values of the Planck function B_n at the layer boundaries. We find that $a_n = B_{n-1}$, and $b_n\Delta_n = B_n - B_{n-1}$, where Δ_n is the whole-layer optical thickness. We expect the discrete ordinate field to show the same dependency on optical thickness x, so we look for solutions of the form

$\mathbf{I}_n^\pm(x) = \tilde{\mathbf{T}}_n^{(1)\pm} + x\tilde{\mathbf{T}}_n^{(2)\pm}$. We decouple the resulting equations by using sum and difference vectors: $\tilde{\mathbf{T}}_n^{(1)\pm} = \frac{1}{2}(\tilde{\mathbf{H}}_n^{(1)} \pm \tilde{\mathbf{J}}_n^{(1)}); \quad \tilde{\mathbf{T}}_n^{(2)\pm} = \frac{1}{2}\tilde{\mathbf{H}}_n^{(2)}$; this reduces the order from $2N$ to N. Substitution in the RTE, and equating powers of x yields the following solution using linear algebra:

$$\left(\tilde{\mathbf{S}}_n^+\tilde{\mathbf{M}}\right)\tilde{\mathbf{H}}_n^{(1)} = a_n(1 - \omega_n)\tilde{\mathbf{E}}; \quad \left(\tilde{\mathbf{S}}_n^+\tilde{\mathbf{M}}\right)\tilde{\mathbf{H}}_n^{(2)} = b_n(1 - \omega_n)\tilde{\mathbf{E}}; \quad \left(\tilde{\mathbf{S}}_n^-\right)\tilde{\mathbf{J}}_n^{(1)} = -\tilde{\mathbf{H}}_n^{(2)}. \qquad (2.2.40)$$

Here, the solution vectors are for the discrete ordinates, and the $\tilde{\mathbf{S}}_n^\pm$ matrices are given by Equation (2.2.6) but with entries restricted to the (1,1) component of the 4 x 4 polarization matrices. Also, in this result, $\tilde{\mathbf{M}} = diag[\mu_1, \mu_2, \dots \mu_N]$, and $\tilde{\mathbf{E}}$ the N x N identity matrix.

Linearization of these solutions is straightforward. The Planck functions depend only on the Blackbody emission temperature, and for now we will leave out consideration of the temperature-field weighting function. In terms of our linearization notation,

$$\mathcal{L}_n(a_n) \equiv \xi_n \frac{\partial a_n}{\partial \xi_n} = 0; \quad \mathcal{L}_n(b_n) \equiv \xi_n \frac{\partial b_n}{\partial \xi_n} = -\frac{(B_n - B_{n-1})}{\Delta_n^2}v_n = -\frac{v_n b_n}{\Delta_n} . \qquad (2.2.41)$$

Thus, linearizing the three systems in Eq. (2.2.40), we find

$$
\begin{aligned}
\left(\tilde{\mathbf{S}}_n^+\tilde{\mathbf{M}}\right)\mathcal{L}_n\left[\tilde{\mathbf{H}}_n^{(1)}\right] &= -a_n u_n\tilde{\mathbf{E}} - \left(\mathcal{L}_n\tilde{\mathbf{S}}_n^+\tilde{\mathbf{M}}\right)\tilde{\mathbf{H}}_n^{(1)} \\
\left(\tilde{\mathbf{S}}_n^+\tilde{\mathbf{M}}\right)\mathcal{L}_n\left[\tilde{\mathbf{H}}_n^{(2)}\right] &= -b_n u_n\tilde{\mathbf{E}} + \mathcal{L}_n\left[b_n\right](1 - \omega_n)\tilde{\mathbf{E}} - \left(\mathcal{L}_n\tilde{\mathbf{S}}_n^+\tilde{\mathbf{M}}\right)\tilde{\mathbf{H}}_n^{(2)} . \\
\left(\tilde{\mathbf{S}}_n^-\right)\mathcal{L}_n\left[\tilde{\mathbf{J}}_n^{(1)}\right] &= -\mathcal{L}_n\left[\tilde{\mathbf{H}}_n^{(2)}\right] - \left(\mathcal{L}_n\tilde{\mathbf{S}}_n^-\right)\tilde{\mathbf{J}}_n^{(1)}
\end{aligned}
\qquad (2.2.42)
$$

The RTE also admits solutions in the absence of scattering. In this case, $\omega_n = 0$, $\tilde{\mathbf{S}}_n^\pm\tilde{\mathbf{M}} = \tilde{\mathbf{E}}$, and the solutions are trivial: $\tilde{\mathbf{H}}_n^{(1)} = a_n\tilde{\mathbf{E}}$, $\tilde{\mathbf{H}}_n^{(2)} = b_n\tilde{\mathbf{E}}$, and $\tilde{\mathbf{J}}_n^{(1)} = -\tilde{\mathbf{M}}^{-1}\tilde{\mathbf{H}}_n^{(2)}$. The linearization (2.2.41) still applies, with the linearized solutions in Eq. (2.2.42) simplified accordingly. This "thermal transmittance" solution has been included in the model in order that fast solutions to the RTE may be obtained in the infrared and beyond.

2.3 The post-processed solution

2.3.1 Boundary value problem (BVP) and linearization

From Section 2.1.3, the complete Stokes vector discrete ordinate solutions in layer n may be written:

$$\tilde{\mathbf{I}}_n^\pm(x) = \tilde{\mathbf{D}}^\pm \sum_{\alpha=1}^{4N}\left[L_{n\alpha}\tilde{\mathbf{W}}_{n\alpha}^\pm e^{-k_{n\alpha}x} + M_{n\alpha}\tilde{\mathbf{W}}_{n\alpha}^\mp e^{-k_{n\alpha}(\Delta_n - x)}\right] + \tilde{\mathbf{Z}}_n^\pm T_{n-1}e^{-\lambda_n x}. \qquad (2.3.1)$$

Quantities $L_{n\alpha}$ and $M_{n\alpha}$ are _constants of integration_ for the homogeneous solutions, and they are determined by the imposition of three boundary conditions as noted in section 2.1.3. For boundary condition (I), we have $\tilde{\mathbf{I}}_n^+(0) = 0$ for $n = 1$, which yields ($T_0 = 1$):

$$\tilde{\mathbf{D}}^+\sum_{\alpha=1}^{4N}\left[L_{n\alpha}\tilde{\mathbf{W}}_{n\alpha}^+ + M_{n\alpha}\tilde{\mathbf{W}}_{n\alpha}^- K_{n\alpha}\right] = -\tilde{\mathbf{Z}}_n^+ . \qquad (2.3.2)$$

For boundary condition (II), the continuity at layer boundaries, we have:

$$\tilde{\mathbf{D}}^{\pm}\sum_{\alpha=1}^{4N}\left[\left\{L_{n\alpha}\tilde{\mathbf{W}}_{n\alpha}^{\pm}\mathrm{K}_{n\alpha}+M_{n\alpha}\tilde{\mathbf{W}}_{n\alpha}^{\mp}\right\}-\left\{L_{p\alpha}\tilde{\mathbf{W}}_{p\alpha}^{\pm}+M_{p\alpha}\tilde{\mathbf{W}}_{p\alpha}^{\mp}\mathrm{K}_{p\alpha}\right\}\right]$$

$$=-\tilde{\mathbf{Z}}_{n}^{\pm}T_{n-1}\Lambda_{n}+\tilde{\mathbf{Z}}_{p}^{\pm}T_{p-1}. \tag{2.3.3}$$

In Eq. (2.3.3), $p = n + 1$. For surface condition (III), staying for convenience with the Lambertian condition in Eq. (2.1.29), we find (for layer $n = N_{\text{TOTAL}}$):

$$\tilde{\mathbf{D}}^{-}\sum_{\alpha=1}^{4N}\left[L_{n\alpha}\tilde{\mathbf{V}}_{\alpha}^{-}\mathrm{K}_{n\alpha}+M_{n\alpha}\tilde{\mathbf{V}}_{\alpha}^{+}\right]=T_{n-1}\Lambda_{n}\left[-\tilde{\mathbf{U}}^{-}+2R_{0}\mu_{0}\tilde{\mathbf{E}}_{1}I_{0}\right]. \tag{2.3.4}$$

Here we have defined the following auxiliary quantities:

$$\tilde{\mathbf{V}}_{\alpha}^{\pm}=\tilde{\mathbf{W}}_{n\alpha}^{\pm}-2R_{0}\tilde{\mathbf{E}}_{1}^{\mathrm{T}}\tilde{\mathbf{M}}\tilde{\mathbf{\Omega}}\tilde{\mathbf{W}}_{n\alpha}^{\mp}\tilde{\mathbf{E}}_{1}; \qquad (n=N_{\text{TOTAL}}) \tag{2.3.5}$$

$$\tilde{\mathbf{U}}^{-}=\tilde{\mathbf{Z}}_{n}^{-}-2R_{0}\tilde{\mathbf{E}}_{1}^{\mathrm{T}}\tilde{\mathbf{M}}\tilde{\mathbf{\Omega}}\tilde{\mathbf{Z}}_{n}^{+}\tilde{\mathbf{E}}_{1}; \qquad (n=N_{\text{TOTAL}}) \tag{2.3.6}$$

$$\tilde{\mathbf{E}}_{1}=diag\{\mathbf{E}_{1},\mathbf{E}_{1},...\mathbf{E}_{1}\}; \tag{2.3.7}$$

$$\mathrm{K}_{n\alpha}=e^{-k_{n\alpha}\Delta_{n}}; \qquad \Lambda_{n}=e^{-\lambda_{n}\Delta_{n}}. \quad (n=1,...N_{\text{TOTAL}}) \tag{2.3.8}$$

Application of Eqs. (2.3.2-2.3.4) yields a large, sparse banded linear system with rank $8N$ x N_{TOTAL}. This system consists only of real variables, and may be written in the symbolic form:

$$\mathbf{\Phi}*\mathbf{\Xi}=\mathbf{\Psi}. \tag{2.3.9}$$

Here $\mathbf{\Psi}$ is constructed from the right hand side variables in Eqs. (2.3.2-2.3.4) and $\mathbf{\Phi}$ is constructed from suitable combinations of $\tilde{\mathbf{V}}_{\alpha}^{\pm}, \tilde{\mathbf{W}}_{n\alpha}^{\pm}$ and $\mathrm{K}_{n\alpha}$. For a visualization of the BVP in the scalar case, see [*Spurr et al.*, 2001]. The vector $\mathbf{\Xi}$ of integration constants is made up of the unknowns $\{L_{n\alpha},M_{n\alpha}\}$ and will be partitioned into contributions from real and complex parts. A schematic of this partitioning is shown in **Figure 2.1**.

The solution proceeds first by the application of a compression algorithm to reduce the order and eliminate redundant zero entries. LU-decomposition is then applied using the banded-matrix LAPACK routine DGBTRF to find the inverse $\mathbf{\Phi}^{-1}$, and the final answer $\mathbf{\Xi}=\mathbf{\Phi}^{-1}*\mathbf{\Psi}$ is then obtained by back-substitution (using DGBTRS). For the slab problem, boundary condition (II) is absent; the associated linear problem is then solved using the DGETRF/DGETRS combination.

Linearizing Eq. (2.3.9) with respect to a variable ξ_p in layer p, we obtain:

$$\mathbf{\Phi}*\mathcal{L}_{p}(\mathbf{\Xi})=\mathbf{\Psi}_{p}'\equiv\mathcal{L}_{p}(\mathbf{\Psi})-\mathcal{L}_{p}(\mathbf{\Phi})*\mathbf{\Xi}. \tag{2.3.10}$$

We notice that this is the same linear-algebra problem, but now with a different source vector $\mathbf{\Psi}_{p}'$ on the right hand side. Since we already have the inverse $\mathbf{\Phi}^{-1}$ from the solution to the original BVP, back-substitution gives the linearization $\mathcal{L}_{p}(\mathbf{\Xi})=\mathbf{\Phi}^{-1}*\mathbf{\Psi}_{p}'$ of the boundary value constants. Although this linearization is straightforward in concept, there are many algebraic details arising with chain rule differentiation required to establish $\mathcal{L}_{p}(\mathbf{\Psi})$ and $\mathcal{L}_{p}(\mathbf{\Phi})$ in Eq. (2.3.10).
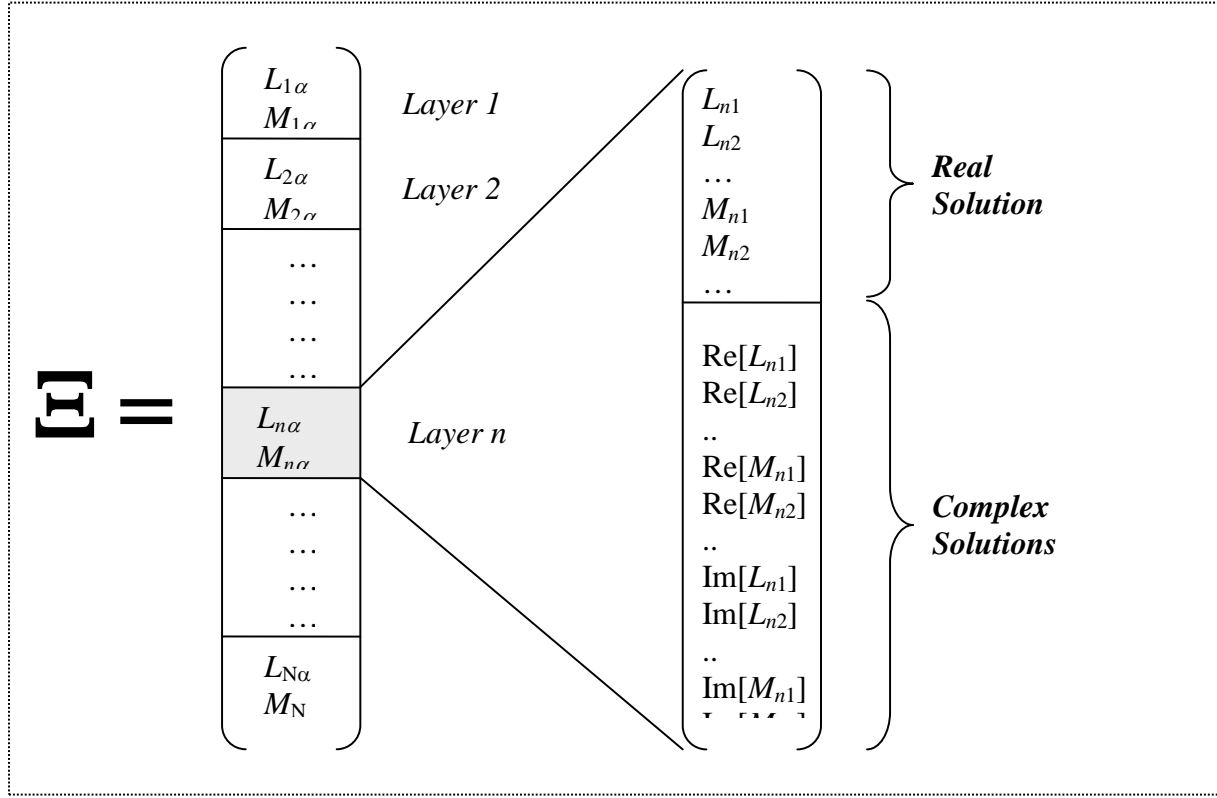
Figure 2.1: Schematic breakdown of the vector of integration constants to be determined as the solution to the boundary value problem in a multilayer atmosphere.

2.3.2 Source function integration

The source function integration technique is used to determine solutions at off-quadrature polar directions μ and at arbitrary optical thickness values in the multi-layer medium. The technique dates back to the work of Chandrasekhar [*Chandrasekhar*, 1960], and has been demonstrated to be superior to numerical interpolation. We substitute layer discrete ordinate solutions (2.3.1) into the multiple scattering integral in Eq. (2.1.22), then integrate over optical thickness. The methodology follows closely that used for the scalar LIDORT code [*Spurr et al.*, 2001; *Spurr*, 2002; *Van Oss and Spurr*, 2002], so long as we remember with the Stokes-vector formulation to use the real part of any quantity derived from combinations of complex-variable entities. Here, we note down the principal results for the upwelling field in the presence of solar scattering.

The solution in layer n at direction μ for optical thickness x (as measured from the top of the layer) is given by:

$$\mathbf{I}_n^-(x,\mu) = \mathbf{I}_n^-(\Delta,\mu)e^{-(\Delta-x)/\mu} + \mathbf{H}_n^-(x,\mu) + \left(\mathbf{Z}_n^-(\mu) + \mathbf{Q}_n^-(\mu)\right)E_n^-(x,\mu) .$$

(2.3.11)

The first term is the upward transmission of the lower-boundary Stokes vector field through a partial layer of optical thickness $\Delta-x$. The other three contributions together constitute the *partial layer source term* due to scattered light contributions. The first of these three is due to the homogeneous solutions and has the form:

$$\mathbf{H}_n^-(x,\mu) = \sum_{\alpha=1}^{4N}\left[L_{n\alpha}\mathbf{X}_{n\alpha}^+(\mu)\mathcal{H}_{n\alpha}^{-+}(x,\mu) + M_{n\alpha}\mathbf{X}_{n\alpha}^-(\mu)\mathcal{H}_{n\alpha}^{--}(x,\mu)\right],$$

(2.3.12)

30

where we have defined the following auxiliary quantities:

$$\mathbf{X}_{n\alpha}^{\pm}(\mu) = \frac{\omega}{2}\sum_{l=m}^{LM}\mathbf{P}_l^m(\mu)\mathbf{B}_{nl}\sum_{j=1}^{N}w_j\left\{\mathbf{P}_l^m(\mu_j)\mathbf{X}_{n\alpha}^{\pm}(\mu_i) + \mathbf{P}_l^m(-\mu_j)\mathbf{X}_{n\alpha}^{\pm}(-\mu_i)\right\};$$ (2.3.13)

$$\mathcal{H}_{n\alpha}^{-+}(x,\mu) = \frac{e^{-xk_{n\alpha}} - e^{-\Delta_n k_{n\alpha}}e^{-(\Delta_n - x)/\mu}}{1 + \mu k_{n\alpha}}$$

$$\mathcal{H}_{n\alpha}^{--}(x,\mu) = \frac{e^{-(\Delta_n - x)k_{n\alpha}} - e^{-(\Delta_n - x)/\mu}}{1 - \mu k_{n\alpha}}$$ (2.3.14a,b)

Here, $\mathbf{X}_{n\alpha}^{\pm}(\mu)$ are homogeneous solutions defined at stream cosine μ, and $\mathcal{H}_{n\alpha}^{-\pm}(x,\mu)$ are the *homogeneous solution multipliers* for the upwelling field. These multipliers arise from the layer optical thickness integration. In (2.3.12), we consider only the real value of the resulting expressions.

The other two layer source term contributions in (2.3.11) come from the diffuse and direct solar source scattering respectively. For the solar source terms, all variables are real numbers, and the relevant quantities are:

$$\mathbf{Z}_n^-(\mu) = \frac{\omega}{2}\sum_{l=m}^{LM}\mathbf{P}_l^m(\mu)\mathbf{B}_{nl}\sum_{j=1}^{N}w_j\left\{\mathbf{P}_l^m(\mu_j)\mathbf{Z}_n^-(\mu_j) + \mathbf{P}_l^m(-\mu_j)\mathbf{Z}_n^-(-\mu_j)\right\};$$ (2.3.15)

$$\mathbf{Q}_n^-(\mu) = \frac{\omega(2 - \delta_{m0})}{2}\sum_{l=m}^{LM}\mathbf{P}_l^m(\mu_i)\mathbf{B}_{nl}\mathbf{P}_l^m(-\mu_0)\mathbf{I}_0;$$ (2.3.16)

$$\mathcal{E}_n^-(x,\mu) = T_{n-1}\frac{e^{-x\lambda_n} - e^{-\Delta_n\lambda_n}e^{-(\Delta_n - x)/\mu}}{1 + \mu\lambda_n}.$$ (2.3.17)

These expressions have counterparts in the scalar code (see for example [*Spurr*, 2002]). Similar expressions can be written for post-processing of downwelling solutions. All source term quantities can be expressed in terms of the basic optical property inputs to VLIDORT {Δ_n, ω_n, $\mathbf{B}_{nl}$}, the pseudo-spherical beam transmittance quantities {T_n, λ_n}, the homogeneous solutions {$k_{n\alpha}, \widetilde{\mathbf{X}}_{n\alpha}^{\pm}$}, the particular solutions $\widetilde{\mathbf{Z}}_n^{\pm}$, and the BVP integration constants {$L_{n\alpha}, M_{n\alpha}$}.

For thermal source terms, the treatment is similar. For simplicity, we consider integration over the whole layer for the upwelling field,. We write:

$$\mathbf{I}_n^-(0,\mu) = \mathbf{I}_n^-(\Delta,\mu)e^{-\Delta/\mu} + \mathbf{H}_n^-(0,\mu) + \mathbf{Z}_n^-(\mu) + \mathbf{D}_n^-(\mu).$$ (2.3.18)

Here, $\mathbf{H}_n^-(0,\mu)$ is defined similarly to the expression in (2.3.12), and the diffuse scattering contribution is given by the following.

$$\mathbf{Z}_n^-(\mu) = \Theta_n^{(1)}(\mu) + \mu\Theta_n^{(2)}(\mu)\left(1 - e^{-\Delta/\mu}\right) - \Delta_n\Theta_n^{(2)}(\mu)e^{-\Delta/\mu}$$

$$\Theta_n^{-(s)}(\mu) = \frac{\omega}{2}\sum_{l=m}^{LM}P_l^m(\mu)\beta_{nl}\sum_{j=1}^{N}w_j\left\{P_l^m(\mu_j)\widetilde{T}_n^{-(s)}(\mu_j) + P_l^m(-\mu_j)\widetilde{T}_n^{-(s)}(-\mu_j)\right\}$$; (2.3.19a,b)

In (2.3.19) we have used components of the thermal discrete ordinate solutions $\widetilde{T}_n^{-(s)}(\pm\mu_j)$, and reduced the definitions to the (1,1) component of any Mueller matrices (thus, $P_l^m(\mu)$ are

Legendre polynomials, and β_n phase function expansion coefficients). The direct term contribution arises from an integration of the Planck source term:

$$\mathbf{D}_n^-(\mu) = (1 - \omega_n)\left[a_n + \mu b_n\left(1 - e^{-\Delta/\mu}\right) - \Delta_n b_n(\mu) e^{-\Delta/\mu}\right]. \tag{2.3.20}$$

Linearizations. Derivatives of all these expressions may be determined by differentiation with respect to variable ξ_n in layer n. The end-points of the chain rule differentiation are the linearized optical property inputs $\{\mathcal{V}_n, \mathcal{U}_n, \mathcal{Z}_{nl}\}$ from Eq. (2.1.33). For linearization of the *homogeneous* post-processing source term in layer n, there is no dependency on any quantities outside of layer n; in other words, $\mathcal{L}_p[\mathbf{H}_n^-(x,\mu)] \equiv 0$ for $p \neq n$. The *particular* solution post-processing source terms in layer n depend on optical thickness values in all layers above and equal to n through the presence of the average secant and the solar beam transmittances, so there will be cross-layer derivatives. However, the chain-rule differentiation method is the same, and requires a careful exercise in algebraic manipulation.

Multiplier expressions (2.3.14), (2.3.15) and (2.3.17) have appeared a number of times in the literature. The linearizations were discussed in [*Spurr*, 2002] and [*Van Oss and Spurr*, 2002], and we need only make two remarks here. Firstly, the real and complex homogeneous solution multipliers are treated separately, with the real part of the complex variable result to be used in the final reckoning. Second, the solar source term multipliers (for example in Eq. (2.3.17) are *the* same as those in the scalar model.

Linearizations of the thermal post-processed solution are straightforward; details for the scalar solution in LIDORT were noted in the review paper [*Spurr*, 2008].

2.4 Spherical and single-scatter corrections in LIDORT

2.4.1 Pseudo-spherical approximation

The pseudo-spherical (P-S) approximation assumes solar beam attenuation for a curved atmosphere. All scattering takes place in a plane-parallel situation. The approximation is a standard feature of many radiative transfer models. We follow the formulation in [*Spurr*, 2002]. **Figure 2.2** provides geometrical sketches appropriate to this section.

We consider a stratified atmosphere of optically uniform layers, with extinction optical depths $\{\Delta_n\}$, $n = 1, N_{\text{TOTAL}}$ (the total number of layers). We take points V_{n-1} and V_n on the vertical (Figure 1, upper panel), and the respective solar beam transmittances to these points are then:

$$T_{n-1} = \exp\left[-\sum_{k=1}^{n-1} s_{n-1,k}\Delta_k\right]; \quad T_n = \exp\left[-\sum_{k=1}^{n} s_{n,k}\Delta_k\right]. \tag{2.4.1}$$

Here, $s_{n,k}$ is the path distance geometrical factor (Chapman factor), equal to the path distance covered by the V_n beam as it traverses through layer k divided by the corresponding vertical height drop (geometrical thickness of layer k). At the top of the atmosphere, $T_0 = 1$. In the *average secant* parameterization, the transmittance to any intermediate point between V_{n-1} and V_n is parameterized by:

$$T(x) = T_{n-1}\exp\left[-\lambda_n x\right], \tag{2.4.2}$$

where x is the vertical optical thickness measured downwards from V_{n-1} and λ_n the average secant for this layer. Substituting (2.4.2) into (2.4.1) and setting $x = \Delta_n$ we find:

$$\lambda_n = \frac{1}{\Delta_n} \left[\sum_{k=1}^{n} s_{n,k} \Delta_k - \sum_{k=1}^{n-1} s_{n-1,k} \Delta_k \right].$$

(2.4.3)

In the plane-parallel case, we have $\lambda_n = \mu_0^{-1}$ for all n.

Linearization: We require derivatives with respect to an atmospheric property ξ_k in layer k. The basic linearized optical property input is the normalized derivative V_n of the layer optical depth extinction Δ_n. Applying the linearization operator to (2.4.3) and (2.4.1), we find:
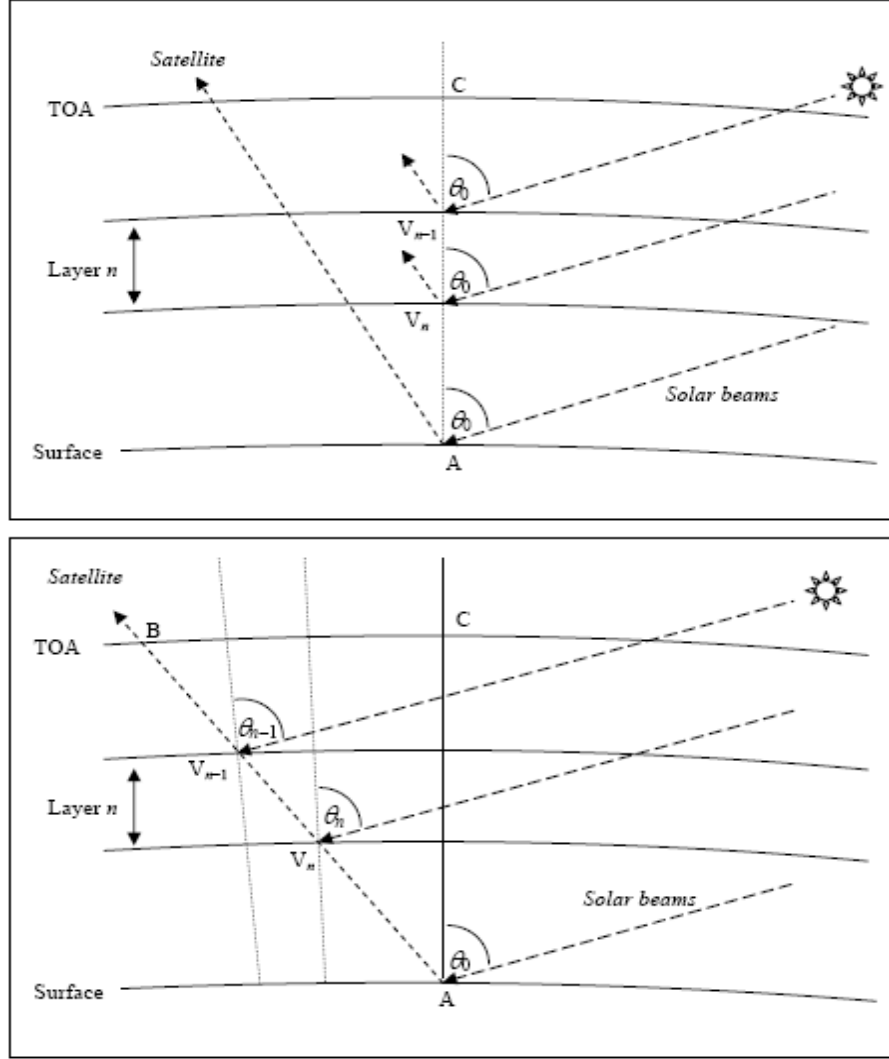
Figure 2.2. (Upper panel) Pseudo-spherical viewing geometry for scattering along the zenith AC. (Lower panel) Line of sight path AB in a curved atmosphere, with viewing and solar angles changing along the path from A to B.

$$\mathcal{L}_k[\lambda_n] = \frac{\mathcal{V}}{\Delta_n}\left(s_{n,n} - \lambda_n\right); \qquad \mathcal{L}_k[T_n] = 0; \qquad\qquad (k = n)$$

$$\mathcal{L}_k[\lambda_n] = \frac{\mathcal{V}_k}{\Delta_n}\left(s_{n,k} - s_{n-1,k}\right); \qquad \mathcal{L}_k[T_n] = -\mathcal{V}_k s_{n-1,k} T_n; \qquad (k < n) \qquad (2.4.4)$$

$$\mathcal{L}_k[\lambda_n] = 0; \qquad\qquad\qquad \mathcal{L}_k[T_n] = 0; \qquad\qquad (k > n)$$

For the plane-parallel case, we have:

$$\mathcal{L}_k[\lambda_n] = 0 \ (\forall k, \forall n); \ \ \mathcal{L}_k[T_n] = -\frac{\mathcal{V}_k T_n}{\mu_0} \ (k < n); \ \ \mathcal{L}_k[T_n] = 0 \ (k \geq n). \qquad (2.4.5)$$

2.4.2 Exact single scatter solutions

In VLIDORT, we include an exact single-scatter computation based on the Nakajima-Tanaka (NT) procedure [*Nakajima and Tanaka*, 1988]. The internal single scatter computation in VLIDORT will use a truncated subset of the complete scatter-matrix information, the number of usable Legendre coefficient matrices $\mathbf{B}_l$ being limited to $2N - 1$ for N discrete ordinate streams.

A more accurate computation results when the post-processing calculation of the truncated single scatter contribution (the term $\mathbf{Q}_n^-(\mu)\mathcal{E}_n^-(x,\mu)$ in Eq. (2.3.11) for example) is suppressed in favor of an accurate single scatter computation, which uses the complete phase function. This is the so called TMS procedure [*Nakajima and Tanaka*, 1988]. This N-T correction procedure appears in the DISORT Version 2.0 [*Stamnes et al.*, 2000] and LIDORT [*Spurr*, 2002] codes. A related computation has been implemented for the doubling-adding method [*Stammes et al.*, 1989].

The (upwelling) post-processed solution in stream direction μ is now written (c.f. Eq. (2.3.11)):

$$\mathbf{I}_n^-(x,\mu) = \mathbf{I}_n^-(\Delta,\mu)e^{-(\Delta-x)/\mu} + \mathbf{H}_n^-(x,\mu) + \left(\mathbf{Z}_n^-(\mu) + \mathbf{Q}_{n,exact}^-(\mu)\right)\mathcal{E}_n^-(x,\mu), \qquad (2.4.6)$$

$$\mathbf{Q}_{n,exact}^-(\mu) = \frac{\omega_n}{4\pi(1-\omega_n f_n)}\mathbf{\Pi}_n(\mu,\mu_0,\phi-\phi_0)\mathbf{I}_0. \qquad (2.4.7)$$

Note the presence of in the denominator of the expression $(1-\omega_n f_n)$ which is required when the delta-M approximation is in force; f_n is the truncation factor (see section 3.4.1). From section 2.1.1, $\mathbf{\Pi}_n$ is obtained from the scattering matrix $\mathbf{F}_n(\Theta)$ through application of rotation matrices. There is no truncation: $\mathbf{\Pi}_n$ can be constructed to any degree of accuracy using all available *unscaled* Greek matrices $\mathbf{B}_{nl}$.

Linearization. Chain-rule differentiation of Eq. (2.4.7) yields the linearization of the exact single scatter correction term. Linearization of the multiplier $\mathcal{E}_n^-(x,\mu)$ has already been established. Since the elements of $\mathbf{\Pi}_n$ consist of linear combinations of $\mathbf{B}_{nl}$, the linearization $\mathcal{L}_n(\mathbf{\Pi}_n)$ is straightforward to write down in terms of the inputs $\mathcal{L}_n(\mathbf{B}_{nl})$.

2.4.3 Sphericity along the line-of-sight

For nadir-geometry satellite instruments with wide-angle off-nadir viewing, one must consider the Earth's curvature along the line of sight from the ground to the satellite. This applies to instruments such as OMI on the Aura platform (swath 2600 km, scan angle 114° at the satellite) [*Stammes et al.*, 1999] and GOME-2 (swath 1920 km) [*EPS/METOP*, 1999]. Failure to account for this effect can lead to errors of 5-10% in the satellite radiance for TOA viewing zenith angles in the range 55-70° [*Spurr*, 2003; *Rozanov et al.*, 2000; *Caudill et al.*, 1997]. For LIDORT, a simple correction for this effect was introduced for satellite geometries in [*Spurr*, 2003]. Correction involves an exact single scatter calculation along the line of sight from ground to TOA: in this case, Eq. (2.4.7) is still valid, but now the geometry is changing from layer to layer. The same correction has been adopted for VLIDORT.

In section 2.4.1, scattering was assumed to take place along the nadir, so that the scattering geometry $\Omega \equiv \{\mu_0, \mu, \phi - \phi_0\}$ is unchanged along the vertical. For a slant line-of-sight path (Figure 2, lower panel), the scattering geometry varies along the path. For layer n traversed by this path, the upwelling Stokes vector at the layer-top is (to a high degree of accuracy) given by:

$$\mathbf{I}^{\uparrow}(\Omega_{n-1}) \cong \mathbf{I}^{\uparrow}(\Omega_n)T(\Omega_n) + \mathbf{\Lambda}_n^{\uparrow}(\Omega_n) + \mathbf{M}_n^{\uparrow}(\Omega_n). \tag{2.4.8}$$

Here, $\mathbf{I}^{\uparrow}(\Omega_n)$ is the upwelling Stokes vector at the layer bottom, $T(\Omega_n)$ the layer transmittance along the line of sight, and $\mathbf{\Lambda}_n^{\uparrow}(\Omega_n)$ and $\mathbf{M}_n^{\uparrow}(\Omega_n)$ are the single- and multiple-scatter layer source terms respectively. The transmittances and layer source terms are evaluated with scattering geometries Ω_n at positions V_n. Equation (2.4.8) is applied recursively, starting with the upwelling Stokes vector $\mathbf{I}_{BOA}^{\uparrow}(\Omega_{NTOTAL})$ evaluated at the surface for geometry Ω_{NTOTAL}, and finishing with the field at top of atmosphere ($n = 0$). The single-scatter layer source terms $\mathbf{\Lambda}_n^{\uparrow}(\Omega_n)$ may be determined through an accurate single scatter calculation (cf. Eq. (2.4.7)) allowing for changing geometrical angles along the line of sight. To evaluate the multiple scatter sources, we run VLIDORT in "multiple-scatter mode" successively for each of the geometries from Ω_{NTOTAL} to Ω_1, retaining only the appropriate multiple scatter layer source terms, and, for the first VLIDORT calculation with the lowest-layer geometry Ω_{NTOTAL}, the surface upwelling Stokes vector $\mathbf{I}_{BOA}^{\uparrow}(\Omega_{NTOTAL})$.

For N_{TOTAL} layers in the atmosphere, we require N_{TOTAL} separate calls to VLIDORT, and this is much more time consuming that a single call with geometry Ω_{NTOTAL} (this would be the default in the absence of a line-of-sight correction). However, since scattering is strongest near the surface, the first VLIDORT call (with geometry Ω_{NTOTAL}) is the most important as it provides the largest scattering source term $\mathbf{M}_{NTOTAL}^{\uparrow}(\Omega_{NTOTAL})$.

An even simpler line-of-sight correction is to assume that *all* multiple scatter source terms are taken from this first VLIDORT call; in this case, we require only the accurate single scatter calculation to complete $\mathbf{I}_{TOA}^{\uparrow}$. This approximation is known as the "outgoing" sphericity correction; it requires very little extra computational effort compared to a single VLIDORT call. The sphericity correction can also be set up with just two calls to VLIDORT made with the start and finish geometries Ω_{NTOTAL} and Ω_1; in this case, multiple scatter source terms at other

geometries are *interpolated* at all levels between results obtained for the two limiting geometries. In the scalar case, accuracies for all these corrections were investigated in [*Spurr*, 2003].

In VLIDORT 2.0, the facility for generating multiple layer source terms has been dropped, as there has been little usage. However, the outgoing sphericity correction is important, and a new formulation has been developed for this release. This has been validated against the TOMRAD code and is applicable also to the vector VLIDORT model. We now describe this.

2.4.4 A more accurate outgoing sphericity correction

In this section, the exposition applies to the scalar intensity, but the treatment is the same for the VLIDORT implementation.

One of the features of the above outgoing sphericity correction is that the average-secant formulation is still assumed to hold for solar beam attenuation. In Eq. (2.4.7), the layer single scattering source terms still contain the post-processing multipliers $\mathcal{E}_n^-(x,\mu)$ which were derived using the average-secant exponential parameterization in terms of the vertical optical thickness coordinate for that layer (Eq. (2.3.17)).

Figure 2 (lower panel) shows the geometry for the single-scattering outgoing sphericity correction along the line of sight. In a non-refractive atmosphere, the solar zenith angle, the line-of-sight zenith angles and the relative azimuth angle between the incident and scattering planes will vary along path AB, but the scattering angle Θ is constant for straight-line geometry.

For single scatter along AB, the current implementation in the LIDORT and VLIDORT codes is based on the average secant approximation of the solar attenuation $A(x)$ to a point along the path with vertical optical thickness x measured from layer-top:

$$A(x) = A_{n-1}\exp[-x\lambda_n]; \qquad \lambda_n = \frac{1}{\Delta_n}\ln\left[\frac{A_{n-1}}{A_n}\right]. \tag{2.4.9}$$

Here A_{n-1} and A_n are transmittances to layer top and layer bottom respectively, and Δ_n is the whole-layer vertical optical thickness. As noted above, the RTE in layer n is:

$$\mu_n\frac{dI(x)}{dx} = I(x) + \frac{\omega_n FP(\Theta)}{4\pi}A_n(x). \tag{2.4.10}$$

Here, μ_n is the averaged line-of-sight cosine defined as the vertical height difference of layer n divided by the line-of-sight slant distance; ω_n is the layer single scattering albedo with phase function $P_n(\Theta)$ (both are constants within the layer), and F is the solar flux. In the average-secant approximation, Eq. (2.4.10) has a straightforward solution that utilizes the exponential dependence of the attenuation to deliver the following closed-form result:

$$I_{n-1}^{\uparrow} = I_n^{\uparrow}\exp[-s_n\Delta_n] + J_n; \tag{2.4.11}$$

$$J_n = H_n A_{n-1}\frac{1-\exp[-(s_n+\lambda_n)\Delta_n]}{\mu_n(s_n+\lambda_n)}. \tag{2.4.12}$$

Here, $4\pi H_n = \omega_n FP(\Theta)$ and $s_n = \mu_n^{-1}$. The TOA result for the upwelling single scattering radiation field is then found by recursion of Eq. (2.4.11).

36

$$I_0^\uparrow = I_{surface}^\uparrow C_N + \sum_{n=1}^{N} J_n C_{n-1} \; ; \tag{2.4.13}$$

$$C_n = \prod_{p=1}^{n} \exp[-s_p \Delta_p] \; ; \qquad C_0 = 1 . \tag{2.4.14}$$

For densely layered atmospheres, the average secant approximation is accurate enough for the range of solar angles considered so far. However, it has been shown [*Spurr*, 2002] that this treatment loses accuracy for very high SZA and for layers that are optically or geometrically thick. A better treatment of outgoing single scatter is therefore required for the LIDORT codes – one that will produce reliable answers for all applications (not just the Rayleigh scattering UV-ozone scenarios), and in particular in the presence of cloud layers and for situations involving broader layers at high SZA. Further, we want this treatment to be fully linearized, in that the resulting single scatter radiation field can be differentiated with respect to any profile variable for which we require Jacobians.

It is a feature of the LIDORT and VLIDORT codes that layers are considered optically uniform, but for outgoing sphericity corrections, the geometry is variable along the viewing path. We rewrite (2.4.10) as:

$$\mu(z) \frac{dI(z)}{dz} = \varepsilon_n I(z) + \varepsilon_n H_n A_n(z) . \tag{2.4.15}$$

Here, the attenuation $A_n(z)$ is computed precisely as a function of the vertical height z, and ε_n is the extinction coefficient for the layer (a constant). From geometry, the viewing zenith angle θ is related to z through:

$$\sin \theta(z) = \frac{(R + z_0) \sin \theta_0}{R + z} . \tag{2.4.16}$$

Here, R is the Earth's radius and the subscript "0" indicates values at TOA. Thus, since $\mu(z) = \cos \theta(z)$, we can change the variable in (2.4.15) to get:

$$\sin^2 \theta \frac{dI(\theta)}{d\theta} = k_n I(\theta) + k_n H_n A_n(\theta) . \tag{2.4.17}$$

Here, $k_n = \varepsilon_n(R + z_0) \sin \theta_0$. An integrating factor for this differential equation is $k_n \cot \theta(z)$, and the whole-layer solution is then:

$$I_{n-1}^\uparrow = I_n^\uparrow e^{k_n(\cot \theta_n - \cot \theta_{n-1})} + \hat{J}_n \; ; \tag{2.4.18}$$

$$\hat{J}_n = -k_n H_n e^{-k_n \cot \theta_{n-1}} \int_{\theta_{n-1}}^{\theta_n} d\theta \frac{A_n(\theta) e^{k_n \cot \theta}}{\sin^2 \theta} . \tag{2.4.19}$$

The integral in (2.4.19) can be done to a very high degree of accuracy by trapezium-rule summation. The TOA upwelling intensity is computed with a recursion similar to that in Eq. (2.4.13). Equation (2.4.19) defines the source term for a whole layer; it is also possible to define a partial-layer source term for output at some intermediate point between the layer boundaries.

Linearization. We consider differentiation with respect to the inherent optical properties (IOPs) defined for each layer – the extinction coefficients ε_n, the scattering coefficients σ_n and the

phase function expansion coefficients β_{nl}. We define a linearization operator with respect to a variable ξ_p in layer p:

$$\mathcal{L}_p(y_n) = \xi_p \frac{\partial y_n}{\partial \xi_p} . \qquad (2.4.20)$$

If $\xi_p = \varepsilon_p$, then $\mathcal{L}_p(k_n) = \delta_{np} k_n$ from the definition of k_n (see after (2.4.17)), and $\mathcal{L}_p(H_n) = 0$. Here, δ_{np} is the Kronecker delta. Differentiating (2.4.18) and (2.4.19) with respect to ε_p yields:

$$\mathcal{L}_p[I_{n-1}^\uparrow] = \{\mathcal{L}_p[I_n^\uparrow] + I_n^\uparrow k_n \delta_{np} (\cot \theta_n - \cot \theta_{n-1})\} e^{k_n(\cot \theta_n - \cot \theta_{n-1})}$$

$$- k_n(1 - k_n \cot \theta_{n-1}) H_n e^{-k_n \cot \theta_{n-1}} \int_{\theta_{n-1}}^{\theta_n} d\theta \frac{A_n(\theta) e^{k_n \cot \theta}}{\sin^2 \theta}$$

$$- k_n H_n e^{-k_n \cot \theta_{n-1}} \left[k_n \int_{\theta_{n-1}}^{\theta_n} d\theta \frac{A_n(\theta) e^{k_n \cot \theta} \cot \theta}{\sin^2 \theta} + \int_{\theta_{n-1}}^{\theta_n} d\theta \frac{\mathcal{L}_p[A_n(\theta)] e^{k_n \cot \theta}}{\sin^2 \theta} \right]. \quad (2.4.21)$$

The only new quantity here is $\mathcal{L}_p[A_n(\theta)]$ in the final integral. To evaluate this, we note that the attenuation of the solar beam to a point z with zenith angle in layer n can be written as:

$$A_n(\theta) = \exp\left[-\sum_{p=1}^{NL} d_{np}(\theta) \varepsilon_p \right]. \qquad (2.4.22)$$

Here, $d_{np}(\theta)$ are geometrical distances independent of the optical properties. The total number of layers in the atmosphere is NL. It follows that:

$$\mathcal{L}_p[A_n(\theta)] = -d_{np}(\theta) \varepsilon_p A_n(\theta) . \qquad (2.4.23)$$

All integrals in the last line of Eq. (2.4.21) can again be done accurately using trapezium rule summations. The linearization was checked using finite difference estimates.

For most geometrical situations, $d_{np}(\theta) = 0$ for all layers $p > n$; this corresponds to points on the line of sight that are illuminated from above. In this case, the attenuation does not depend on extinction coefficients in layers below n, and hence $\mathcal{L}_p[A_n(\theta)] = 0$ for $p > n$. However, there are situations (near the top of the atmosphere for a wide off-nadir viewing path and a high solar zenith angle) in which some points along the line-of-sight are illuminated by direct sunlight coming from below the horizontal. In this case, the solar path has gone through a tangent height in the atmosphere, and $d_{np}(\theta)$ is not necessarily zero for $p > n$.

For a differentiation with respect to other optical properties, the situation is simpler. Defining now a linearization:

$$\mathcal{L}_q = \sigma_q \frac{\partial}{\partial \sigma_q} ; \qquad (2.4.24)$$

with respect to the scattering coefficient σ_q in layer q, the only non-vanishing term arising in the linearization of (2.4.19) is $\mathcal{L}_q[H_n] = H_n \delta_{nq}$, so that we have:

$$\mathcal{L}_q[I^{\uparrow}_{n-1}] = \mathcal{L}_q[I^{\uparrow}_n]e^{k_n(\cot\theta_n - \cot\theta_{n-1})} - \delta_{nq}k_n H_n e^{-k_n \cot\theta_{n-1}} \int_{\theta_{n-1}}^{\theta_n} d\theta \frac{A(\theta)e^{k_n \cot\theta}}{\sin^2\theta}. \qquad (2.4.25)$$

NOTE. VLIDORT as a single-scatter RT model

Following user feedback from a number of individuals, VLIDORT now has the capability to return just the single-scatter Stokes vector. This option is controlled by a single flag (called DO_SSFULL), which if set, ensures that the VLIDORT multiple scatter calculation is avoided altogether and the model returns only the singly scattered radiances and Jacobians. The above single scatter corrections then apply for atmospheric scattered light; and it is only necessary to include for the upwelling field the transmitted direct-beam reflectance. We also note that an additional delta-M scaling procedure has been optionally included for the single scatter computation. When the model is running with multiple scatter included, then this additional scaling can be applied on top of the usual delta-M scaling applied to the truncated phase function in the diffuse field equations; in this case the Nakajima-Tanaka TMS procedure still applies. With VLIDORT in single-scatter-only mode, the original diffuse-field delta-M scaling is not required, though the additional truncation can still be applied if desired.

REMARK on the FO code

For Version 2.7, an alternative stand-alone single-scatter module has been added to VLIDORT. This is the FO ("first-order") module. This is completely stand-alone, with its own I/O which is completely separate from that of VLIDORT itself. There is a dedicated interface routine for calling this module - this interface will prepare FO inputs from their VLIDORT equivalents, call the FO module itself, and copy the FO output into the VLIDORT_SS arrays for further use inside VLIDORT.

The user does not see this operation, which is controlled by a single flag called DO_FO_CALC. As noted in section 6, this DO_FO_CALC Boolean flag must be hard-wired by the user, as there is no file-read character-string input at the present time.

The FO module has the same single scatter options as the "VLIDORT-native" single-scatter corrections described above. The FO code will calculate a "Nadir" SS correction when just the pseudo-spherical calculation by itself is required (i.e. only the incoming solar beam treated in spherical geometry), as opposed to a more accurate "Outgoing sphericity" calculation in which *both* the incoming solar beam and outgoing line-of-sight paths are treated spherically. This choice is governed by the usual VLIDORT "SSCORR" flags.

For nadir SS corrections, the FO results are the same as those from VLIDORT's native correction software. However, results for "outgoing" SS corrections are slightly different, as the FO code uses a more natural Gaussian quadrature scheme for evaluating the integrals in Eqs. (2.4.19), (2.4.21) and 2.4.25). The number of quadrature points is set by the VLIDORT input variable NFINELAYERS which also applies to the internal SSCORR_OUTGOING computation.

From user feedback, it was noted that the internal "outgoing" SS correction produced spurious results in the presence of optically thick (>5) cloud layers - this was caused by inaccurate SS integration when solar beam attenuation is vanishingly small. This bug has been remedied in the FO code through the use of a "criticality" mechanism, which detects cases when solar beam extinction occurs in optically thick layers, and truncates the scatter integrals to reduced layers for which attenuation is non-vanishing.

2.5 Bulk (total column) atmospheric Jacobians.

One of the most important applications for VLIDORT has been in forward model simulations for ozone profile and total column retrieval. The use of total column as a proxy for the ozone profile was recognized a number of years ago by scientists at NASA, and column-classified ozone profile climatologies were created for the TOMS Version 7 [*Wellemayer et al.*, 1997] and Version 8 retrieval algorithms [*Bhartia*, 2003]. If the profile is represented as a set $\{U_j\}$ of partial columns in Dobson Units [DU], then the total column (also in [DU]) is $C = \sum_j U_j$. For two adjacent TOMS profiles $\{U_j^{(1)}\}$ and $\{U_j^{(2)}\}$ with total columns $C^{(1)}$ and $C^{(2)}$ we define an intermediate profile with column amount C according to:

$$U_j(C) = \left(\frac{C - C^{(1)}}{C^{(2)} - C^{(1)}}\right)U_j^{(2)} + \left(\frac{C^{(2)} - C}{C^{(2)} - C^{(1)}}\right)U_j^{(1)}. \tag{2.5.1}$$

This defines the profile-column map; it is linear in C. Total column weighting functions are related to profile Jacobians by means of chain rule differentiation and the partial derivative:

$$\frac{\partial U_j(C)}{\partial C} = \frac{U_j^{(2)} - U_j^{(1)}}{C^{(2)} - C^{(1)}}. \tag{2.5.2}$$

This map allows us to interpolate smoothly between profile entries in the climatology. In effect, we are drawing on an ensemble of possible profiles of which the climatology is a sample. Other maps are possible. TOMS Version 8 profiles are specified for 18 latitude bands from pole to pole (10° intervals), and for each month of the year.

Suppose now we have a Rayleigh atmosphere with Rayleigh scattering cross-section $\sigma^{Ray}(\lambda)$, air column density D_p in layer p, ozone partial columns U_p, and temperature-dependent ozone cross sections $\sigma_p^{O3}(\lambda)$; then the bulk property IOPs are:

$$\Delta_p = \sigma^{Ray}(\lambda)D_p + \sigma_p^{O3}(\lambda)U_p; \qquad \omega_p = \frac{\sigma^{Ray}(\lambda)D_p}{\Delta_p}; \tag{2.5.3}$$

Differentiating (2.5.2) with respect to U_p gives the linearized IOP inputs for the profile Jacobian:

$$\frac{\partial \Delta_p}{U_p} = \sigma_p^{O3}(\lambda); \qquad \frac{\partial \omega_p}{\partial U_p} = -\frac{\omega_p}{\Delta_p}\frac{\partial \Delta_p}{\partial U_p}. \tag{2.5.4}$$

Finally, we compute the column Jacobian using the chain rule:

$$K_{col}(\Omega) \equiv \frac{\partial I(0,\Omega)}{\partial C} = \sum_{p=1}^{n}\frac{\partial I(0,\Omega)}{\partial U_p}\frac{\partial U_p}{\partial C}. \tag{2.5.5}$$

Note the use of the profile-column mapping derivatives $\partial U_p/\partial C$. Merely adding up the partial column weighting functions is equivalent to assuming that the response of the TOA field to variations in total ozone is the same for all layers – the profile shape remains the same. Equation (2.5.5) is the correct formula to account for shape variation.

It is perfectly possible to set up VLIDORT to deliver a set of *profile* Jacobians; a *column* weighting function would then be created *externally* from the sum in Eq. (2.5.5). This is not very

efficient, since for a 13-layer atmosphere we must calculate 13 separate profile weighting functions and then sum them. However, a facility was introduced in the scalar LIDORT code Version 2.5 to have LIDORT calculate the column Jacobian directly; in effect, the summation in Eq. (2.5.5) is done *internally*. This is a much more efficient procedure. In this case the linearized IOP inputs are expressed in terms of the profile-column mapping derivatives:

$$\frac{\partial \Delta_p}{\partial C} = \sigma_p^{O3}(\lambda)\frac{\partial U_p}{\partial C}; \qquad\qquad \frac{\partial \omega_p}{\partial C} = -\frac{\omega_p}{\Delta_p}\sigma_p^{O3}(\lambda)\frac{\partial U_p}{\partial C}. \qquad (2.5.6)$$

This feature has been retained in all LIDORT Version 3 codes, and the single-scatter corrections (outgoing and nadir), surface treatments and performance enhancements (in particular the linearization of the reduced BVP problem) have been upgraded to ensure that the column differentiation is done internally inside LIDORT if the requisite flag is turned on for column linearization. The model will generate either profile or column Jacobians for atmospheric quantities.

2.6 Facility for blackbody Jacobians

2.6.1 Introduction

In this section, we are concerned with one of the last remaining analytic Jacobian derivations for the VLIDORT radiative transfer model, namely, the derivation of radiation-field Jacobians with respect to Black-Body (BB) Planck Functions; the latter are specified at layer boundaries when calculating the radiation field in the presence of atmospheric and surface thermal emission (in LTE). For given wave number, these Planck functions depend solely on the corresponding temperatures at level boundaries, so that this BB-Jacobian derivation is ultimately of great importance in deriving analytic temperature Jacobians in the thermal scattering regime.

In [*Spurr and Christi*, 2014], the procedure for generation of VLIDORT-based Jacobians with respect to temperatures specified at *level* boundaries was discussed. VLIDORT is a linearized RT model for atmospheres with optically-uniform layer stratification, and that *layer* Jacobians can be derived with respect to any layer optical properties. Level-boundary temperature Jacobians are then found using the functional dependence of these layer properties on level-boundary temperatures and the chain-rule applied to VLIDORT Jacobian output. This process applies to the solar regime, where temperature dependence is restricted to optical properties, but the additional Planck-function temperature dependence in the presence of thermal sources is not accounted for.

Version 2.7 of VLIDORT has been given the capability for delivering BB Jacobians; this procedure is outlined in the next three subsections. First, we review the solution for the RTE in the presence of thermal sources (atmospheric and surface), focusing on the Green's function methodology used in LIDORT. Second, we carry out an explicit linearization of this thermal solution with respect to the Planck functions. Thirdly, we show how a complete set of level-boundary temperature derivatives can be determined for the thermal scattering regime from VLIDORT outputs and suitable chain-rule dependencies. The Planck function temperature derivatives are briefly reviewed in the final subsection.

2.6.2 LIDORT RTE solutions in the thermal scattering regime.

We assume a stratified atmosphere with *N* layers. Without loss of generality, we ignore solar beam scattering. The RTE in layer *n* in the presence of thermal sources may be written:

$$\mu \frac{dI(x,\mu)}{dx} = -I(x,\mu) + \frac{\omega_n}{2} \int_{-1}^{1} \Phi_n(\mu,\mu') I(x,\mu') d\mu' + \frac{(1-\omega_n)}{4\pi} B_n(x), \qquad (2.6.1)$$

where $\Phi_n(\mu,\mu')$ is the azimuth-independent contribution to the Fourier-series expansion of the scattering phase function, ω_n is the layer single scattering albedo, x the optical thickness measured from the top of the atmospheric layer, and $B_n(x)$ the Planck function source in that layer. Scattering properties $\{\omega_n, \Phi_n\}$ are assumed independent of x.

From the discrete-ordinate RT theory, the (particular integral) Green's function solution to the RTE with thermal sources for a single Fourier component in the cosine-azimuth expansion of the radiation field is [*Spurr*, 2002]:

$$\mathbf{W}_n^{\pm}(x) = \Omega_n \sum_{\alpha=1}^{N} \left[A_{n\alpha}^{-} C_{n\alpha}^{-}(x) \mathbf{X}_{n\alpha}^{\mp} + A_{n\alpha}^{+} C_{n\alpha}^{+}(x) \mathbf{X}_{n\alpha}^{\pm} \right], \qquad (2.6.2)$$

Here, $\{\pm k_{n\alpha}, \mathbf{X}_{n\alpha}^{\pm}\}$ are the down-/up-welling ($+/-$) separation constants and eigensolutions to the homogeneous multiple-scatter RTE (in the absence of sources), and $\Omega_n = 1 - \omega_n$. There are N_d discrete ordinates in the polar-direction half-space [0, 1] with Gaussian-quadrature abscissa and weights $\{\mu_j, w_j\}$, and homogeneous solutions have index α. The other terms in Eq. (2.6.2) are

$$A_{n\alpha}^{\pm} = \frac{1}{\langle R_{n\alpha} \rangle} \sum_{j=1}^{N_d} w_j \left[X_{jn\alpha}^{+} \pm X_{jn\alpha}^{-} \right]; \qquad \langle R_{n\alpha} \rangle = \sum_{j=1}^{N_d} \mu_j w_j \left[\left(X_{jn\alpha}^{+} \right)^2 - \left(X_{jn\alpha}^{-} \right)^2 \right]; \qquad (2.6.3)$$

$$C_{n\alpha}^{+}(x) = e^{-x k_{n\alpha}} \int_{0}^{x} e^{+y k_{n\alpha}} B_n(y) dy \; ; \qquad C_{n\alpha}^{-}(x) = e^{+x k_{n\alpha}} \int_{x}^{\Delta_n} e^{-y k_{n\alpha}} B_n(y) dy. \qquad (2.6.4)$$

Here, $C_{n\alpha}^{\pm}(x)$ are the downwelling and upwelling Green's function multipliers, which contain all the dependence on optical thickness. Terms in Eq. (2.6.3) are constructed solely from homogeneous RTE solutions. Assuming a piecewise-linear dependence of the Planck function on x, the source term is

$$B_n(x) = a_n + b_n x, \qquad (2.6.5)$$

$$a_n = B_{n-1} \quad ; b_n = \frac{B_n - B_{n-1}}{\Delta_n}, \qquad (2.6.6)$$

in terms of the Planck functions B_{n-1} and B_n at the top and bottom boundaries of the layer. The set $\{B_n\}$ for $n = 0, 1, \cdots N$ is a fundamental thermal-emission input to the RT model.

Integrals in Eq. (2.6.4) are straightforward using the linear regime in Eqs. (2.6.5) and (2.6.6). We evaluate the two integrals at $x = \Delta_n$ and $x = 0$ respectively, and express the results as linear combinations of B_{n-1} and B_n:

$$C_{n\alpha}^{+}(\Delta_n) = \mathcal{P}_{n\alpha}^{+} B_{n-1} + \mathcal{Q}_{n\alpha}^{+} B_n \,; \qquad (2.6.7a)$$

$$C_{n\alpha}^{-}(0) = \mathcal{P}_{n\alpha}^{-} B_{n-1} + \mathcal{Q}_{n\alpha}^{-} B_n \,. \qquad (2.6.7b)$$

where

$$\mathcal{P}_{n\alpha}^{+} = \Phi_{n\alpha} - \frac{1}{\Delta_n k_{n\alpha}} (\Delta_n - \Phi_{n\alpha}); \quad \mathcal{Q}_{n\alpha}^{+} = +\frac{1}{\Delta_n k_{n\alpha}} (\Delta_n - \Phi_{n\alpha}); \qquad (2.6.8a)$$

$$\mathcal{P}_{n\alpha}^{-} = \Phi_{n\alpha} + \frac{1}{\Delta_n k_{n\alpha}} (\Delta_n Z_{n\alpha} - \Phi_{n\alpha}); \quad \mathcal{Q}_{n\alpha}^{-} = -\frac{1}{\Delta_n k_{n\alpha}} (\Delta_n Z_{n\alpha} - \Phi_{n\alpha}); $$

$$(2.6.8b)$$

$$Z_{n\alpha} = e^{-\Delta_n k_{n\alpha}}; \; \Phi_{n\alpha} = \frac{1}{k_{n\alpha}} (1 - Z_{n\alpha}). \qquad (2.6.8c)$$

We note also that

$$C_{n\alpha}^{+}(0) = 0; \ C_{n\alpha}^{-}(\Delta_n) = 0.\tag{2.6.9}$$

We can now compute the particular integrals at layer boundaries, by substituting these results into Eq. (2.6.8); again we write the answers as linear combinations of Planck functions:

$$\mathbf{W}_n^{\pm}(0) = B_{n-1}\,\mathbf{P}_{n\alpha}^{\pm} + B_n\,\mathbf{Q}_{n\alpha}^{\pm}$$

$$\equiv B_{n-1}\,\Omega_n \sum_{\alpha=1}^{N}\left[A_{n\alpha}^{-}\mathcal{P}_{n\alpha}^{-}\mathbf{X}_{n\alpha}^{\mp}\right] + B_n\,\Omega_n\sum_{\alpha=1}^{N}\left[A_{n\alpha}^{-}\mathcal{Q}_{n\alpha}^{-}\mathbf{X}_{n\alpha}^{\mp}\right];\tag{2.6.10a}$$

$$\mathbf{W}_n^{\pm}(\Delta_n) = B_{n-1}\,\mathbf{R}_{n\alpha}^{\pm} + B_n\,\mathbf{S}_{n\alpha}^{\pm}$$

$$\equiv B_{n-1}\,\Omega_n\sum_{\alpha=1}^{N}\left[A_{n\alpha}^{+}\mathcal{P}_{n\alpha}^{+}\mathbf{X}_{n\alpha}^{\pm}\right] + B_n\,\Omega_n\sum_{\alpha=1}^{N}\left[A_{n\alpha}^{+}\mathcal{Q}_{n\alpha}^{+}\mathbf{X}_{n\alpha}^{\pm}\right].\tag{2.6.10b}$$

Although we have chosen values at the layer boundaries, the integrals in Eq. (2.6.4) can be done readily enough for arbitrary optical thickness x, and quantities $\mathbf{W}_n^{\pm}(x)$ can then be derived as linear combinations of Planck functions as in Eqs. (2.6.10).

The discrete ordinate solution of the RTE in layer n is then a linear combination of the homogeneous solutions plus the Green's function particular integral:

$$\mathbf{I}_n^{\pm}(x) = \sum_{\alpha=1}^{N}\left[L_{n\alpha}\mathbf{X}_{n\alpha}^{\pm}e^{-xk_{n\alpha}} + M_{n\alpha}\mathbf{X}_{n\alpha}^{\mp}e^{-(\Delta_n-x)k_{n\alpha}}\right] + \mathbf{W}_n^{\pm}(x)\,.\tag{2.6.11}$$

Here, the constants of integration $\{L_{n\alpha}, M_{n\alpha}\}$ will be determined by imposition of three boundary conditions: (1) at the top boundary $n = 0$ there is no diffuse downwelling radiation; (2) at the bottom boundary $n = N$ there is a surface reflection condition plus surface emission; and (3) there is continuity of the diffuse field at intermediate layer boundaries $n = 1, \dots N - 1$.

The boundary-value results in the linear algebra problem $\mathbb{M}.\mathbb{X} = \mathbb{Y}$ for the vector $\mathbb{X}$ of all unknown layer integration constants; here, matrix $\mathbb{M}$ is a sparsely populated band-matrix constructed solely from RTE homogeneous solutions, while vector $\mathbb{Y}$ expresses the boundary conditions applied to the Green's function particular integrals [*Spurr et al.*, 2001]. For simplicity, we assume a dark surface with no reflection; surface emissivity is then unity (in the following, B_{sur} is the surface emission Planck function). The explicit form for $\mathbb{Y}$ may be written:

$$\mathbb{Y} = \begin{pmatrix} -\mathbf{W}_1^{+}(0) \\ \mathbf{W}_2^{\pm}(0) - \mathbf{W}_1^{\pm}(\Delta_1) \\ \dots \\ \mathbf{W}_N^{\pm}(0) - \mathbf{W}_{N-1}^{\pm}(\Delta_{N-1}) \\ \mathbf{W}_N^{-}(\Delta_N) + \mathbf{E}B_{sur} \end{pmatrix} = \begin{pmatrix} -B_0\,\mathbf{P}_{1\alpha}^{+} - B_1\,\mathbf{Q}_{1\alpha}^{+} \\ -B_2\,\mathbf{Q}_{2\alpha}^{\pm} - B_0\,\mathbf{R}_{1\alpha}^{\pm} + B_1\left(\mathbf{P}_{2\alpha}^{\pm} - \mathbf{S}_{1\alpha}^{\pm}\right) \\ \dots \\ -B_N\,\mathbf{Q}_{N\alpha}^{\pm} - B_{N-2}\,\mathbf{R}_{N-1,\alpha}^{\pm} + B_{N-1}\left(\mathbf{P}_{N\alpha}^{\pm} - \mathbf{S}_{N-1,\alpha}^{\pm}\right) \\ B_{N-1}\,\mathbf{R}_{N\alpha}^{-} + B_N\mathbf{S}_{N\alpha}^{-} + \mathbf{E}B_{sur} \end{pmatrix}.$$

$$\tag{2.6.12}$$

Here, $\mathbf{E}$ is vector of rank N_d with unit entries; this represents the emissivity.

The solution vector $\mathbb{X} = \mathbb{M}^{-1}.\mathbb{Y}$ gives us $\{L_{n\alpha}, M_{n\alpha}\}$ for all layers; we note that each of these quantities is dependent on the complete set $\{B_n\}$, $n = 0, \dots N$, and also on B_{sur}.

In VLIDORT the "post-processed" solution of the radiation field at arbitrary polar direction μ is obtained by source function integration. Essentially, for each layer, the discrete-ordinate field in Eq. (2.6.11) is inserted into the multiple-scatter integral in Eq. (2.6.1), and the resulting first-order differential equation integrated over optical thickness. Details may be found in the LIDORT literature [*Spurr*, 2008]. We remark only that the post-processed solution may be readily expressed as linear combinations of the atmospheric and surface Planck function inputs.

2.6.3 Linearization with respect to Planck Functions

We wish to carry out an analytic differentiation of the thermal RT solution with respect to $\{B_k\}$ in order to establish the required Jacobians. From Eq. (2.6.10), the derivatives of the Green's function solutions at the level boundaries could not be simpler:

$$\frac{\partial \mathbf{W}_n^\pm(0)}{\partial B_{n-1}} = \mathbf{P}_{n\alpha}^\pm; \quad \frac{\partial \mathbf{W}_n^\pm(0)}{\partial B_n} = \mathbf{Q}_{n\alpha}^\pm; \tag{2.6.13a}$$

$$\frac{\partial \mathbf{W}_n^\pm(\Delta_n)}{\partial B_{n-1}} = \mathbf{R}_{n\alpha}^\pm; \quad \frac{\partial \mathbf{W}_n^\pm(\Delta_n)}{\partial B_n} = \mathbf{S}_{n\alpha}^\pm. \tag{2.6.13b}$$

Derivatives $\frac{\partial \mathbf{W}_n^\pm(x)}{\partial B_k}$ (where $k = n$ or $k = n-1$) of the more general expression $\mathbf{W}_n^\pm(x)$ can be found in a similar manner.

Given boundary-value problem $\mathbb{M}.\mathbb{X} = \mathbb{Y}$ and its solution $\mathbb{X} = \mathbb{M}^{-1}.\mathbb{Y}$, the linearization of this is $\delta_k \mathbb{X} = \mathbb{M}^{-1}.\delta_k \mathbb{Y}$, where we have used the shorthand $\delta_k = \frac{\partial}{\partial B_k}$. Matrix $\mathbb{M}$ has no dependence on the Planck functions ($\delta_k \mathbb{M} = \mathbf{0}$), and $\delta_k \mathbb{Y}$ may be written down by an explicit examination of the entries in Eq. (2.6.10) and the Green's function derivatives in Eq. (2.6.13). Indeed, for the first two levels:

$$\delta_0 \mathbb{Y} = \begin{pmatrix} -\mathbf{P}_{1\alpha}^+ \\ -\mathbf{R}_{1\alpha}^+ \\ 0 \\ 0 \\ \dots \\ 0 \end{pmatrix}; \quad \delta_1 \mathbb{Y} = \begin{pmatrix} -\mathbf{Q}_{1\alpha}^+ \\ \mathbf{P}_{2\alpha}^\pm - \mathbf{S}_{1\alpha}^\pm \\ -\mathbf{R}_{2\alpha}^+ \\ 0 \\ \dots \\ 0 \end{pmatrix}., \text{etc} \dots \tag{2.6.14}$$

The trick here is to make sure that the vector entries $\delta_k \mathbb{Y}$ are filled up correctly - a variable defined at level boundary k will have an effect on the two adjacent layers above and below this boundary. Since we have the inverse matrix $\mathbb{M}^{-1}$ from the original solution of the radiance field, the linearized vector $\delta_k \mathbb{X}$ is readily computed.

The linearized discrete ordinate solution in layer n with respect to the Planck function B_k at level k can now be written

$$\frac{\partial \mathbf{I}_n^\pm(x)}{\partial B_k} = \sum_{\alpha=1}^N \left[\frac{\partial L_{n\alpha}}{\partial B_k} \mathbf{X}_{n\alpha}^\pm e^{-x k_{n\alpha}} + \frac{\partial M_{n\alpha}}{\partial B_k} e^{-(\Delta_n - x)k_{n\alpha}} \mathbf{X}_{n\alpha}^\mp \right] + \frac{\partial \mathbf{W}_n^\pm(x)}{\partial B_k}. \tag{2.6.15}$$

Here the derivatives $\left\{ \frac{\partial L_{n\alpha}}{\partial B_k}, \frac{\partial M_{n\alpha}}{\partial B_k} \right\}$ emerge from the $\delta_k \mathbb{X}$ vector, while $\frac{\partial \mathbf{W}_n^\pm(x)}{\partial B_k}$ follows from the explicit differentiation. As noted above, derivatives $\frac{\partial \mathbf{W}_n^\pm(x)}{\partial B_k}$ exists only for $n = k$ or $n = k+1$, but $\left\{ \frac{\partial L_{n\alpha}}{\partial B_k}, \frac{\partial M_{n\alpha}}{\partial B_k} \right\}$ are defined for all values of n.

Surface emission Jacobian. We make one remark here about Jacobians with respect to the surface Planck function B_{sur}. Clearly none of the Green's function solutions depend on this quantity, so there are no derivatives there; we are left with a linearization of the boundary value problem. If we write $\delta_{sur} = \frac{\partial}{\partial B_{sur}}$, the linearized problem is $\delta_{sur}\mathbb{X} = \mathbb{M}^{-1}.\delta_{sur}\mathbb{Y}$, where from Eq. (2.6.11):

$$\delta_{sur} \mathbb{Y} = \begin{pmatrix} 0 \\ 0 \\ \dots \\ 0 \\ \mathbf{E} \end{pmatrix}. \tag{2.6.16}$$

Post-processing. Here, we do not go through the linearization of the post-processed solution, but this may be determined readily enough through differentiation of the components of the post-processed radiance solution.

Remark. Green's function methods are not used in VLIDORT; instead the thermal solution is determined by conventional substitution of an expression $\mathbf{W}_n^{\pm}(x) = \mathbf{G}_n^{\pm} + x\mathbf{H}_n^{\pm}$ into the RTE, where the vectors $\{\mathbf{G}_n^{\pm}, \mathbf{H}_n^{\pm}\}$ are then determined through linear algebra. However, these vectors can be expressed as linear combinations of the set $\{B_n\}, n = 0, \dots N$, so that the linearization outlined above can proceed - the key lies with the boundary-value problem.

2.6.4 Level boundary temperature Jacobians for thermal regimes

Here we consider temperature dependence for both Planck functions and layer optical properties. We assume at the start that we have already obtained (1) profile Jacobian output from VLIDORT with respect to optical properties in *layers*, and (2) profile Jacobian output with respect to the Planck functions at *levels* - the new feature described in the previous sections. In addition, we assume that Planck function temperature derivatives $\{\partial B_n/\partial T_n\}$ are defined for $n = 0,1,\cdots N$ for level boundary temperatures $\{T_n\}$.

To obtain relationships between layer optical properties and $\{T_n\}$, we follow the prescription in [*Spurr and Christi*, 2014], assuming that: (i) Atmospheric quantities (pressure $\{p_n\}$, temperature $\{T_n\}$, height $\{h_n\}$, trace gas volume mixing ratios $\{v_n\}$, aerosol particle concentrations $\{\epsilon_n\}$) are defined at level boundaries; (ii) for $\{p_n\}, \{T_n\}$ and $\{h_n\}$, the atmosphere is in hydrostatic equilibrium, but only one of these quantities is independently defined; (iii) integration between levels is be done using simple trapezoidal summation, with $\{T_n\}$ and $\{h_n\}$ interpolating log-linearly with $\{p_n\}$, and constituent quantities $\{v_n\}$ and $\{\epsilon_n\}$ interpolating linearly with height; and (iv) the air density at level n follows the rule $\rho_n = \Gamma \frac{p_n}{T_n}$, where Γ is a constant related to density at STP.

We assume for now that there is no fine-layering scheme involved in the integration of the hydrostatic equation (there is no loss of generality in this assumption). In this case, the *layer* trace gas absorption optical thickness $A_n(\lambda_s)$ at wavelength (λ_s) is then

$$G_n(\lambda_s) = \tfrac{1}{2}d_n \left\{ \frac{v_n p_n \sigma_{s,n}(T_n)}{T_n} + \frac{v_{n-1} p_{n-1} \sigma_{s,n-1}(T_{n-1})}{T_{n-1}} \right\} \Gamma. \tag{2.6.17}$$

Here Γ is a constant, $d_n = h_{n-1} - h_n$ is the height separation, and the cross-sections $\sigma_{s,n}(T_n)$ are assumed to be known functions of temperature. (In the thermal regime, cross-sections are determined through line-by-line (LBL) calculations based on line-spectroscopy data sets; the derivation of LBL temperature derivatives of $\sigma_{s,n}(T_n)$ was discussed in [*Spurr and Christi*, 2014]). We define the Rayleigh scattering optical thickness $R_n(\lambda_s)$ in a similar way, treating the Rayleigh cross-section ς_s as independent of temperature:

$$R_n(\lambda_s) = \tfrac{1}{2}d_n \left\{ \frac{p_n}{T_n} + \frac{p_{n-1}}{T_{n-1}} \right\} \varsigma_s \Gamma. \tag{2.6.18}$$

The aerosol optical thickness (AOT) can be written down similarly in terms of the aerosol loading and the extinction cross-section, but we assume that these quantities are independent of temperature and pressure.

From (2.6.17) and (2.6.18), we have the following partial derivatives for $k = n - 1$ or $k = n$:

$$\frac{\partial G_n(\lambda_s)}{\partial T_k} = \frac{1}{2}\Gamma d_n \frac{v_k p_k}{T_k}\left\{\frac{\partial \sigma_{s,k}}{\partial T_k} - \frac{\sigma_{s,k}}{T_k}\right\}; \quad \frac{\partial R_n(\lambda_s)}{\partial T_k} = -\frac{1}{2}\Gamma d_n \frac{p_k \varsigma_s}{T_k^2}. \tag{2.6.19}$$

From VLIDORT, we now have optical property _layer_ Jacobians $\frac{\partial I}{\partial G_n}$ and $\frac{\partial I}{\partial R_n}$ for $n = 1, \ldots N$, and Planck function _level_ Jacobians (from the previous section) $\frac{\partial I}{\partial B_n}$ for $n = 0,1, \ldots N$. Then application of chain-rule derivatives yields the complete level-boundary temperature Jacobians, where we have written $j = k - 1$:

$$\frac{\partial I}{\partial T_k} = \frac{\partial I}{\partial B_k}\frac{\partial B_k}{\partial T_k} + \left\{\frac{\partial I}{\partial G_k}\frac{\partial G_k}{\partial T_k} + \frac{\partial I}{\partial G_j}\frac{\partial G_j}{\partial T_k}\right\} + \left\{\frac{\partial I}{\partial R_k}\frac{\partial R_k}{\partial T_k} + \frac{\partial I}{\partial R_j}\frac{\partial R_j}{\partial T_k}\right\} \quad k = 1, \cdots N - 1; \tag{2.6.20}$$

$$\frac{\partial I}{\partial T_0} = \frac{\partial I}{\partial B_0}\frac{\partial B_0}{\partial T_0} + \frac{\partial I}{\partial G_1}\frac{\partial G_1}{\partial T_0} + \frac{\partial I}{\partial R_1}\frac{\partial R_1}{\partial T_0}; \tag{2.6.21}$$

$$\frac{\partial I}{\partial T_N} = \frac{\partial I}{\partial B_N}\frac{\partial B_N}{\partial T_N} + \frac{\partial I}{\partial G_N}\frac{\partial G_N}{\partial T_N} + \frac{\partial I}{\partial R_N}\frac{\partial R_N}{\partial T_N}. \tag{2.6.22}$$

These formulae may be readily generalized if there is fine-layer trapezoidal summation assumed in the hydrostatic equation integration - this was done for trace gas absorption and Rayleigh optical thickness values in [*Spurr and Christi*, 2014], and the results are equally applicable here.

The Planck Function temperature derivative is easy to write down. Indeed:

$$B(T) = \frac{2hv^3}{c^2}\frac{1}{e^{\frac{hv}{k_B T}}-1}, \tag{2.6.23}$$

in terms of the usual constants h, c and k_B (Boltzmann's constant), and frequency v. Thus

$$\frac{\partial B(T)}{\partial T} = \frac{2hv^3}{c^2}\frac{e^{\frac{hv}{k_B T}}}{\left(e^{\frac{hv}{k_B T}}-1\right)^2}\frac{hv}{k_B T^2}. \tag{2.6.24}$$

In practice, we use the integrated form $2\Delta v \bar{B}_v(T) = \int_{v-\Delta v}^{v+\Delta v} B_v(T)dv$ over a small interval $2\Delta v$. This is done numerically, and the integration uses several numerical approximations depending on the wave-number region. The same approximations can be applied for the integrated form of the temperature derivative in Eq. (2.6.5). Some software has been developed for VLIDORT to deliver Planck function T-derivatives, starting from the numerical routine supplied for the DISORT code [*Stamnes et al.*, 2000].

IMPORTANT NOTE. (10 OCTOBER 2014)

The Blackbody Jacobian option in VLIDORT is not currently enabled, pending further testing. There is currently no test for this option in the VLIDORT 2.7 release, and the user will not be able to choose this option. However, the LIDORT Version 3.7 release has a test program for this facility, and the code works for the scalar case.

3. The numerical VLIDORT model

3.1 Preparation of inputs

3.1.1 Basic optical property inputs

In this section, we give a brief introduction to the input requirements for VLIDORT, in particular the determination of optical property inputs (including linearized quantities). It is already clear that for a Stokes vector computation using VLIDORT, we require the input set $\{\Delta_n, \omega_n, \mathbf{B}_{nl}\}$ for each layer n, where Δ_n is the total optical thickness, ω_n the total single scatter albedo, and $\mathbf{B}_{nl}$ the set of Greek matrices specifying the total scattering law. The form for $\mathbf{B}_{nl}$ is given in Eq. (2.1.24) in terms of the six Greek constants $\{\alpha_l, \beta_l, \gamma_l, \delta_l, \varepsilon_l, \zeta_l\}$ which must be specified for each moment l of a Legendre function expansion in terms of the cosine of the scattering angle. The values β_l are the traditional phase function expansion coefficients, the ones that appear as inputs to the scalar version; they are normalized to 4π.

As an example, we consider a medium with Rayleigh scattering by air molecules, some trace gas absorption, and scattering and extinction by aerosols. Dropping the layer index, if the layer Rayleigh scattering optical depth is δ_{Ray} and trace gas absorption optical thickness α_{gas}, with the aerosol extinction and scattering optical depths τ_{aer} and δ_{aer} respectively, then the *total optical property inputs* are given by:

$$\Delta = \alpha_{gas} + \delta_{Ray} + \tau_{aer}; \quad \omega = \frac{\delta_{aer} + \delta_{Ray}}{\Delta}; \quad \mathbf{B}_l = \frac{\delta_{Ray}\mathbf{B}_{l,Ray} + \delta_{aer}\mathbf{B}_{l,aer}}{\delta_{Ray} + \delta_{aer}}. \quad (3.1.1)$$

The Greek matrix coefficients for Rayleigh scattering are given by the following table.

	α_l	β_l	γ_l	δ_l	ε_l	ζ_l
$l=0$	0	1	0	0	0	0
$l=1$	0	0	0	$\dfrac{3(1-2\rho)}{2+\rho}$	0	0
$l=2$	$\dfrac{6(1-\rho)}{2+\rho}$	$\dfrac{(1-\rho)}{2+\rho}$	$-\dfrac{\sqrt{6}(1-\rho)}{2+\rho}$	0	0	0

For zero depolarization ratios, the only surviving Greek constants are: $\beta_0 = 1.0$, $\beta_2 = 0.5$, $\alpha_2 = 3.0$, $\gamma_2 = -\sqrt{6}/2$ and $\delta_1 = 1.5$. Aerosol quantities must in general be derived from a suitable particle scattering model (Mie calculations, T-matrix methods, etc.).

We consider a 2-parameter bimodal aerosol optical model with the following *combined optical property definitions* in terms of the total aerosol number density N and the fractional weighting f between the two aerosol modes:

$$\Delta_{aer} = Ne_{aer} \equiv N[fe_1 + (1-f)e_2]; \quad (3.1.2a)$$

$$\omega_{aer} = \frac{\sigma_{aer}}{e_{aer}} \equiv \frac{fz_1e_1 + (1-f)z_2e_2}{e_{aer}}; \quad (3.1.2b)$$

$$\beta_{l,aer} = \frac{fz_1 e_1 \beta_l^{(1)} + (1-f)z_2 e_2 \beta_l^{(2)}}{\sigma_{aer}} . \tag{3.1.2c}$$

The quantity σ_{aer} is the combined scattering coefficient and e_{aer} the combined extinction coefficient. In Eq. (3.1.2c) we have given the combined expression for just one of the Greek constants; the other 5 are constructed in a similar fashion. Thus, the quantity $\beta_{l,Aer}$ is the l-th coefficient in the Legendre polynomial expansion of the phase function. Here, e_1, z_1 and $\beta_l^{(1)}$ are the extinction coefficient, single scatter albedo and Legendre expansion coefficient for aerosol type 1; similar definitions apply to aerosol type 2.

3.1.2 Linearized optical property inputs

For the linearized inputs with respect to a parameter ξ for which we require weighting functions, we define normalized quantities:

$$\phi_\xi = \frac{\xi}{\Delta}\frac{\partial \Delta}{\partial \xi}; \quad \varphi_\xi = \frac{\xi}{\omega}\frac{\partial \omega}{\partial \xi}; \quad \mathbf{\Psi}_{l,\xi} = \frac{\xi}{\mathbf{B}_l}\frac{\partial \mathbf{B}_l}{\partial \xi} . \tag{3.1.3}$$

These may be established by differentiating the definitions in Eq. (3.1.1). We give one example here: if there is a single absorbing gas (ozone, for example), with C the partial column of trace gas in any given layer, and σ_{gas} the absorption coefficient, then we have $\Delta = C\sigma_{gas} + \delta_{Ray} + \tau_{aer}$ in the above equations. For trace gas profile Jacobians, we require the derivatives in Eq. (3.1.3) as inputs, taken with respect to C. These are:

$$\phi_C \equiv \frac{C}{\Delta}\frac{\partial \Delta}{\partial C} = \frac{C\sigma_{gas}}{\Delta}; \quad \varphi_C \equiv \frac{C}{\omega}\frac{\partial \omega}{\partial C} = -\frac{C\sigma_{gas}}{\Delta}; \quad \mathbf{\Psi}_{l,\xi} \equiv \frac{\xi}{\mathbf{B}_l}\frac{\partial \mathbf{B}_l}{\partial \xi} = 0 . \tag{3.1.4}$$

Jacobian parameters may be elements of the retrieval state vector, or they may be sensitivity parameters which are not retrieved but will be sources of error in the retrieval. As another example (keeping to the notation used for the above bi-modal aerosol model), we will assume that the retrieval parameters are the total aerosol density N and the bimodal ratio f. All other quantities in the above definitions are sensitivity parameters.

For the _retrieval Jacobians_ (with respect to N and f) the relevant inputs are found by partial differentiation of the definitions in Eq. (3.1). After some algebra, one finds (we have just considered one for the Greek-matrix elements for simplicity):

$$N\frac{\partial \Delta}{\partial N} = N\frac{\partial \Delta_{aer}}{\partial N} = \Delta_{aer}; \qquad f\frac{\partial \Delta}{\partial f} = f\frac{\partial \Delta_{aer}}{\partial f} = fN(e_1 - e_2); \tag{3.1.5a}$$

$$N\frac{\partial \omega}{\partial N} = \frac{N\sigma_{aer} - \omega\Delta_{aer}}{\Delta}; \qquad f\frac{\partial \omega}{\partial f} = \frac{fN[(z_1 e_1 - z_2 e_2) - \omega(e_1 - e_2)]}{\Delta}; \tag{3.1.5b}$$

$$N\frac{\partial \beta_l}{\partial N} = \frac{N\sigma_{aer}(\beta_{l,aer} - \beta_l)}{N\sigma_{aer} + \sigma_{Ray}}; \qquad f\frac{\partial \beta_l}{\partial f} = \frac{fN(z_1 e_1 - z_2 e_2)(\beta_{l,aer} - \beta_l)}{N\sigma_{aer} + \sigma_{Ray}}. \tag{3.1.5c}$$

For _sensitivity Jacobians_, the quantities σ_{Ray}, α_{gas}, e_1, z_1, e_2 and z_2 are all _bulk property_ model parameters that are potentially sources of error. [We can also consider the phase function quantities γ_{Ray}, γ_1 and γ_2 as sensitivity parameters, but the results are not shown here]. After a lot

more algebra (chain rule differentiation, this time not normalizing), we find the following derivatives:

$$\frac{\partial \Delta}{\partial \sigma_{Ray}} = 1; \qquad \frac{\partial \omega}{\partial \sigma_{Ray}} = \frac{1-\omega}{\Delta}; \qquad \frac{\partial \beta_l}{\partial \sigma_{Ray}} = \frac{\beta_{l,Ray} - \beta_l}{N\sigma_{aer} + \sigma_{Ray}}; \qquad (3.1.6a)$$

$$\frac{\partial \Delta}{\partial \alpha_{Gas}} = 1; \qquad \frac{\partial \omega}{\partial \alpha_{Gas}} = -\frac{\omega}{\Delta}; \qquad \frac{\partial \beta_l}{\partial \alpha_{Gas}} = 0; \qquad (3.1.6b)$$

$$\frac{\partial \Delta}{\partial e_1} = Nf; \qquad \frac{\partial \omega}{\partial e_1} = \frac{Nf(z_1 - \omega)}{\Delta}; \qquad \frac{\partial \beta_l}{\partial e_1} = \frac{fz_1(\beta_{l,aer} - \beta_l)}{N\sigma_{aer} + \sigma_{Ray}}; \qquad (3.1.6c)$$

$$\frac{\partial \Delta}{\partial e_2} = N(1-f); \qquad \frac{\partial \omega}{\partial e_2} = \frac{N(1-f)(a_2 - \omega)}{\Delta}; \qquad \frac{\partial \beta_l}{\partial e_2} = \frac{(1-f)z_2(\beta_{l,aer} - \beta_l)}{N\sigma_{aer} + \sigma_{Ray}}; \qquad (3.1.6d)$$

$$\frac{\partial \Delta}{\partial z_1} = 0; \qquad \frac{\partial \omega}{\partial z_1} = \frac{Nfe_1}{\Delta}; \qquad \frac{\partial \beta_l}{\partial z_1} = \frac{fe_1(\beta_{l,aer} - \beta_l)}{N\sigma_{aer} + \sigma_{Ray}}; \qquad (3.1.6e)$$

$$\frac{\partial \Delta}{\partial z_2} = 0; \qquad \frac{\partial \omega}{\partial z_2} = \frac{N(1-f)e_2}{\Delta}; \qquad \frac{\partial \beta_l}{\partial z_2} = \frac{(1-f)e_2(\beta_{l,aer} - \beta_l)}{N\sigma_{aer} + \sigma_{Ray}}. \qquad (3.1.6f)$$

3.1.3 Additional atmospheric inputs

VLIDORT is a pseudo-spherical model dealing with the attenuation of the solar beam in a curved atmosphere, and it therefore requires some geometrical information. The user needs to supply the earth's radius R_{earth} and a height grid $\{z_n\}$ where $n = 0, 1, ...$ NLAYERS (the total number of layers); heights must be specified at layer boundaries with z_0 being the top of the atmosphere. This information is sufficient if the atmosphere is non-refracting.

If the atmosphere is refracting, it is necessary to specify pressure and temperature fields $\{p_n\}$ and $\{t_n\}$, also defined at layer boundaries. The refractive geometry calculation inside VLIDORT is based on the Born-Wolf approximation for refractive index $n(z)$ as a function of height: $n(z) = 1 + \alpha_0 p(z)/t(z)$. Factor α_0 depends slightly on wavelength, and this must be specified by the user if refractive bending of the solar beams is desired. To a very good approximation it is equal to 0.000288 multiplied by the air density at standard temperature and pressure. VLIDORT has an internal fine-layering structure to deal with repeated application of Snell's law. In this regard, the user must specify the number of fine layers to be used for each coarse layer (10 is usually sufficient).

3.1.4 Surface property inputs

The kernel-based BRDF treatment has now been separated from the main VLIDORT code. Calculation of the BRDF kernels and Fourier components of the BRDF is now performed in a dedicated BRDF supplement. Thus, VLIDORT now receives total BRDFs and their Fourier components (and if required, the surface-property linearizations of these quantities), without knowledge of the individual kernels used to construct these quantities. A brief description of the available BRDF kernels and their inputs are given here. For a fuller treatment, consult the BRDF supplement appendix (section 6.3).

For BRDF input, it is necessary for the user to specify up to three amplitude coefficients $\{R_k\}$ associated with the choice of kernel functions, and the corresponding vectors $\{\mathbf{b}_k\}$ of non-linear coefficients. For example, if the BRDF is a single Cox-Munk function, it is only necessary to specify the wind speed (in meters/second) and the relative refractive index between water and air. Table 3.1 lists the available kernel functions found in the VLIDORT BRDF supplement.

Table 3.1. Summary of BRDF kernels available in the VLIDORT BRDF supplement

#	Kernel Name	# parameters	type	Source
1	Lambertian	0	Scalar	----
2	Ross-thick	0	Scalar	Wanner et al., 1995
3	Ross-thin	0	Scalar	MODIS
4	Li-sparse	2	Scalar	MODIS
5	Li-dense	2	Scalar	MODIS
6	Roujean	0	Scalar	MODIS
7	Rahman(RPV)	3	Scalar	Rahman et al., 1993
8	Hapke	3	Scalar	Hapke, 1993
9	Cox-Munk	2	Scalar	Cox/Munk, 1954
10	GISS Cox-Munk	2	Vector	Mishchenko/Travis 1997
11	GISS Cox-Munk CRI	2	Vector	Natraj, 2012 (personal communication)
12	BPDF 2009	2	Vector	Maignan et al., 2009
13	New Cox-Munk	3	Scalar	[A. Sayer, 2014]

It will be noticed that most options are "scalar", that is reflectance is only applied to the total intensity part of the Stokes vector, and the BRDF matrix is non-zero only for the (1,1) component. The specular reflection of polarized light from randomly reflecting surfaces is well known, and the glitter BRDF ("GISS Cox-Munk") is based on software available from the NASA GISS site, with formalism described in the paper by [*Mishchenko and Travis*, 1997]. For land surfaces, there is a dearth of source material in the literature but some reasonably accurate empirically-based kernels have been developed in recent years from analyses of POLDER and MISR data [*Maignan et al.*, 2009]. The VLIDORT implementation (BPDF 2009 in Table 3.1) is included by kind permission of the authors.

Note we do not need to specify full tables of BRDF values for each Fourier component. The supplement has BRDF routines for calculating values of the kernel functions for all possible combinations of angles, and additional routines for delivering the Fourier components of the kernel functions. Fourier component specification is done numerically by integration over the azimuth angle, and for this, it is necessary to specify the number of BRDF azimuth quadrature abscissa N_{BRDF}. The choice $N_{\text{BRDF}} = 50$ is sufficient to obtain numerical accuracy of 10^{-4} in this Fourier component calculation. Nonetheless, the user is allowed to choose N_{BRDF}.

For surface property weighting functions, we need only specify whether we require weighting functions with respect to $\{R_k\}$ and/or to the components of vectors $\{\mathbf{b}_k\}$. Additional inputs are thus restricted to a number of Boolean flags; the BRDF supplement takes care of the rest.

3.1.5 Thermal emission inputs

For atmospheric thermal emission input, the current specification in VLIDORT requires the Planck Black-body function to be input at layer boundaries. The surface emission input requires a separate Planck function as input. A convenient routine for generating the integrated Planck

function in [W.m^{-2}] was developed as an internal routine for the DISORT code [*Stamnes et al.*, 2000]; this can be used outside the VLIDORT environment to generate the required Planck functions. This Planck function generator has been linearized with respect to temperature, so that all thermal source terms are differentiable for temperature retrievals.

For thermal emission alone, Planck functions are specified in physical units (the usual unit is [W.m^{-2}.v^{-1}]. For solar sources only, output is normalized to the input solar flux vector (which can be set to arbitrary units). For calculations with both sources, the solar flux must be specified in the same physical unit as that for the Plank function input.

3.2 Validation and benchmarking

3.2.1 Checking against the scalar code

VLIDORT is designed to work equally with Stokes 4-vectors {*I,Q,U,V*} and in the scalar mode (*I* only). The first validation task for the vector model is to run it in scalar mode and reproduce results generated independently from the scalar LIDORT model. A set of options can be used to test the major functions of the model (the *real* RT solutions, the boundary value problem and post processing) for the usual range of scenarios (single layer, multilayer, arbitrary level output and viewing angles, plane-parallel versus pseudo-spherical, etc.). This battery of tests is very useful, but of course it does not validate the Stokes-vector solutions and in particular the complex variable RTE formalism (absent in the scalar RT).

In this section, we make one important point concerning the verification of the multi-layer capability. This can easily be tested using the invariance principle: two optically identical layers of optical thickness values x_1 and x_2 will (at least for plane-parallel geometry) produce a field equivalent to that produced by an optically identical layer of thickness $x_1 + x_2$. This applies equally to the scalar and vector models. This technique is particularly useful for testing implementations of the boundary value linear algebra solution (section 2.3.1). We now turn to verification using slab problem results.

3.2.2 The Rayleigh slab problem

A first validation was carried out against the Rayleigh atmosphere results published in the tables of Coulson, Dave and Sekera [*Coulson et al.*, 1960]. These tables apply to a single-layer pure Rayleigh slab in plane parallel geometry; the single scattering albedo is 1.0 and there is no depolarization considered in the scattering matrix. Tables for Stokes parameters *I*, *Q* and *U* are given for three surface albedos (0.0, 0.25, 0.80), a range of optical thickness values from 0.01 to 1.0, for 7 azimuths from 0° to 180° at 30° intervals, some 16 view zenith angles with cosines from 0.1 to 1.0, and for 10 solar angles with cosines from 0.1 to 1.0. With the single scattering albedo set to 0.999999, VLIDORT was able to reproduce all these results to within the levels of accuracy specified in the introduction section of the CDS tables.

3.2.3 Benchmarking for aerosol slab problems

The benchmark results noted in [*Siewert*, 2000b] were used; all 8 output tables in this work were reproduced by VLIDORT. The slab problem used a solar angle 53.130° ($\mu_0 = 0.6$), with single scatter albedo $\omega = 0.973527$, surface albedo 0.0, total layer optical thickness of 1.0, and a set of Greek constants as noted in Table 1 of [*Siewert*, 2000b]. Output was specified at a number of

optical thickness values from 0 to 1, and at a number of output streams. 24 discrete ordinate streams were used in the half space for the computation. In Table 3.2, we present VLIDORT results for intensity at relative azimuth angle of 180°; the format is deliberately chosen to mimic that used in [*Siewert*, 2000b]. It is clear that the agreement with his Table 8 is almost perfect. The only point of issue is the downwelling output at $\mu = 0.6$: this is a limiting case because $\mu_0 = 0.6$ as well. Such a case requires l'Hopital's rule to avoid singularity, and this has been implemented in VLIDORT (as in LIDORT), but had not discussed in Siewert's paper. All tables in [*Siewert*, 2000a] were reproduced, with differences of 1 or 2 in the sixth decimal place (excepting the above limiting case).

Table 3.2 Replica of Table 8 from [*Siewert*, 2000b].

	0.000	0.125	0.250	0.500	0.750	0.875	1.000
-1.0	5.06872E-02	4.26588E-02	3.45652E-02	1.97273E-02	7.87441E-03	3.36768E-03	
-0.9	4.49363E-02	3.83950E-02	3.16314E-02	1.87386E-02	7.81148E-03	3.42290E-03	
-0.8	4.95588E-02	4.29605E-02	3.59226E-02	2.19649E-02	9.46817E-03	4.21487E-03	
-0.7	5.54913E-02	4.89255E-02	4.16034E-02	2.63509E-02	1.18019E-02	5.35783E-03	
-0.6	6.19201E-02	5.57090E-02	4.83057E-02	3.18640E-02	1.49296E-02	6.94694E-03	
-0.5	6.84108E-02	6.30656E-02	5.59610E-02	3.87231E-02	1.91563E-02	9.19468E-03	
-0.4	7.44303E-02	7.06903E-02	6.44950E-02	4.72940E-02	2.50375E-02	1.25100E-02	
-0.3	7.89823E-02	7.78698E-02	7.35194E-02	5.79874E-02	3.35858E-02	1.77429E-02	
-0.2	8.01523E-02	8.29108E-02	8.16526E-02	7.07286E-02	4.66688E-02	2.69450E-02	
-0.1	7.51772E-02	8.29356E-02	8.56729E-02	8.26216E-02	6.65726E-02	4.61143E-02	
-0.0	5.93785E-02	7.61085E-02	8.33482E-02	8.76235E-02	8.22105E-02	7.53201E-02	
0.0		7.61085E-02	8.33482E-02	8.76235E-02	8.22105E-02	7.53201E-02	6.04997E-02
0.1		4.81348E-02	7.00090E-02	8.63151E-02	8.80624E-02	8.49382E-02	7.76333E-02
0.2		2.95259E-02	5.13544E-02	7.72739E-02	8.77078E-02	8.84673E-02	8.55909E-02
0.3		2.07107E-02	3.91681E-02	6.67896E-02	8.29733E-02	8.70779E-02	8.79922E-02
0.4		1.58301E-02	3.14343E-02	5.81591E-02	7.72710E-02	8.36674E-02	8.74252E-02
0.5		1.28841E-02	2.64107E-02	5.17403E-02	7.22957E-02	8.01999E-02	8.60001E-02
0.6		1.10823E-02	2.32170E-02	4.74175E-02	6.88401E-02	7.78121E-02	8.51316E-02
0.7		1.01614E-02	2.15832E-02	4.53651E-02	6.77032E-02	7.75916E-02	8.61682E-02
0.8		1.03325E-02	2.19948E-02	4.67328E-02	7.07013E-02	8.16497E-02	9.14855E-02
0.9		1.31130E-02	2.72721E-02	5.64095E-02	8.41722E-02	9.68476E-02	1.08352E-01
1.0		4.54878E-02	8.60058E-02	1.53099E-01	2.03657E-01	2.23428E-01	2.39758E-01

An additional benchmarking for VLIDORT Version 2.0 and higher was done against the results of Garcia and Siewert [*Garcia and Siewert*, 1989] for another slab problem, this time with albedo 0.1. With VLIDORT set to calculate using only 20 discrete ordinate streams in the half space, tables 3-10 in [*Garcia and Siewert*, 1989] were reproduced to within 1 digit of six significant figures. This result is noteworthy because the radiative transfer computations in this paper were done using a completely different radiative transfer methodology (the so-called F_N method).

3.2.4 Weighting function verification

For the verification of analytically calculated Jacobians, it is only necessary to validate the derivative by using a finite difference estimate (ratio of the small change in the Stokes vector induced by a small change in a parameter in one layer):

$$\mathbf{K}_\xi \equiv \frac{\partial \mathbf{I}}{\partial \xi} \approx \frac{\delta \mathbf{I}}{\delta \xi} . \tag{3.2.1}$$

This applies equally to column and surface Jacobians. The VLIDORT version 2.4 tests contain analytic Jacobians that have been validated by finite differences, and the installation program contains software to carry out this validation for all types of Jacobians

Verification of each stage of the linearization may also be done in this way. A word is in order about the use of finite differences in general. It is of course always possible to attempt weighting function estimations using finite difference methods. However, there are pitfalls associated with this procedure (quite apart from the arbitrariness and time-consuming nature of the exercise). In certain situations, a small perturbation of one or more of the Greek constants can give rise to a set of eigensolutions which cannot be compared (in a finite-difference sense) with those generated with the original unperturbed inputs. This numerical difficulty sometimes occurs with lower-precision codes.

3.3 Performance considerations

3.3.1 The delta-M approximation

In the scalar model, sharply peaked phase functions are approximated as a combination of a delta-function and a smoother residual phase function. This is the delta-M approximation [*Wiscombe*, 1977], which is widely used in discrete ordinate and other RT models. The delta-M scaled optical property inputs (optical thickness, single scatter albedo, phase function Legendre expansion coefficients) are:

$$\bar{\tau} = \tau(1 - \omega f) \; ; \quad \overline{\omega} = \omega \frac{(1 - f)}{(1 - \omega f)} \; ; \quad \overline{\beta}_l = \frac{\beta_l - f(2l + 1)}{(1 - f)} \; . \tag{3.3.1}$$

The delta-M *truncation factor* is:

$$f = \frac{\beta_{2N}}{(2N + 1)} \; . \tag{3.3.2}$$

In VLIDORT, Legendre coefficients β_l appear as the (1,1) entry in matrix $\mathbf{B}_l$. In line with the scalar definition in terms of the phase function, we take in VLIDORT the truncation factor f as defined Eq. (3.3.2), and adopt the following scaling for the six entries in $\mathbf{B}_l$. Four coefficients (α_l, β_l, ζ_l and δ_l) will scale as β_l in Eq. (3.3.1), while the other two coefficients γ_l and ε_l scale as $\widetilde{\gamma}_l = \gamma_l / (1 - f)$. This specification can also be found in [*Chami et al.*, 2001] where a more detailed justification is presented. Scaling for the optical thickness and single scatter albedo in Eq. (3.3.1) is the same in the vector model. Linearizations of Eqs. (3.3.1) and (3.3.2) are straightforward, and these are discussed in [*Spurr*, 2002] for the scalar model.

3.3.2 Multiple solar zenith angle facility

In solving the RTE, the first step is always to determine solutions of the homogeneous equations in the absence of solar sources. This process does not need to be repeated for each solar beam source. In DISORT and earlier versions of LIDORT, only one solar zenith angle is specified, and the models must be called from scratch every time results are required for a new solar geometry. In the new code, the homogeneous solution is solved once before the loop over each solar configuration starts; for each solar beam geometry g, we generate a set of particular integral solutions P_g for our multi-layer atmosphere.

In solving the boundary value problem, we apply boundary conditions at all levels in the atmosphere, ending up with a large but sparse linear algebra system in the form $\mathbf{A}\mathbf{X}_g = \mathbf{B}_g$. Here, $\mathbf{X}_g$ is the vector of integration constants appropriate to solar beam with geometry g, $\mathbf{B}_g$ is the

source term vector consisting of contributions from the set of particular solutions P_g, and the banded tri-diagonal matrix **A** contains only contributions from the RTE homogeneous solutions. The inverse matrix **A⁻¹** can be determined once only, before the loop over solar geometry starts. This is the most time consuming step in the complete solution for the RT field, and once completed, it is straightforward and fast to set the integration constants $\mathbf{X}_g = \mathbf{A^{-1}B}_g$ by back substitution.

In summary then, two important operations on the homogeneous RT field are carried out before any reference to solar beam terms. Thus, the VLIDORT code has an internal loop over SZA angles. It is well known that convergence of the Fourier cosine azimuth series for the radiation field depends on the solar beam angle. We keep track of the convergence separately for each SZA; once the intensity field at our desired output angles and optical depths has converged for one particular SZA, we stop further calculation of Fourier terms for this SZA, even though solutions at other SZAs still require further computation of Fourier terms.

This multiple SZA feature was implemented at the outset in VLIDORT. This is a very substantial performance enhancement for VLIDORT, particularly in view of the increased time taken over the eigenproblem and the much larger BVP matrix inversion compared with the scalar code.

3.3.3 Eigensolver usage

We have already noted differences between the LAPACK solver DGEEV and the condensed version ASYMTX as used in LIDORT and DISORT. DGEEV must be used for any layers with scattering by aerosols or clouds, since there will be complex roots in this case. ASYMTX only deals with real symmetric eigenmatrices. Linearization of the homogeneous solutions from DGEEV uses adjoint theory and has some subtleties; adjoint solutions are not available for ASYMTX.

It turns out that, aside from additional elements down the diagonal, the eigenmatrix $\mathbf{\Gamma}_n$ in layer n consists of blocks of 4x4 matrices of the form $\mathbf{P}_{lm}(\mu_i)\mathbf{B}_{nl}\mathbf{P}_{lm}^{\mathrm{T}}(\mu_j)$, where the **P** and $\mathbf{B}_{nl}$ matrices were defined in Eq. (2.1.24) and (2.1.25) in section 2.1.1, μ_i are the discrete ordinates, and the 'T' superscript denotes matrix transpose. Since **P** and $\mathbf{P}^{\mathrm{T}}$ are symmetric, then $\mathbf{\Gamma}_n$ will be symmetric if $\mathbf{B}_{nl}$ is. Thus, $\mathbf{\Gamma}_n$ will be symmetric if the Greek constants ε_l in $\mathbf{B}_{nl}$ are zero for all values of l. This is a special case satisfied by the Rayleigh scattering law, but in general, this is not true for scattering with aerosols and clouds.

For aerosols and clouds, we require the complex eigensolver DGEEV from LAPACK, but for Rayleigh scattering we can use the faster "real-only" ASYMTX package. Our policy in VLIDORT will be to retain both eigensolvers and use them as appropriate – if any of the Greek constants ε_l in $\mathbf{B}_{nl}$ is non-zero for a given scattering layer, then we will choose the complex eigensolver in that layer. The use of ASYMTX and its linearization for Rayleigh layers will represent a considerable saving in processing time. For an application with a few particulate layers in an otherwise Rayleigh-scattering atmosphere, both eigensolvers will be required.

In version 2.4, VLIDORT can run with 2 or 3 Stokes vectors instead of the full 4. This is helpful for atmospheric simulations, since circular polarization in the Earth's atmosphere is typically three orders of magnitude smaller that its linear counterpart. For a 3-component calculation of I, Q, and U, the model uses 3x3 Mueller and scattering matrices, with the (3,4) and (4,4) elements of the Greek matrix omitted. In this case, the eigenproblem has real-valued solutions only, and this contributes to the substantial performance savings to be gained with this reduced problem.

3.3.4 Solution saving

In DISORT and earlier LIDORT versions, the models contained full computations of all RTE solutions in all layers and for all Fourier components contributing to radiance outputs. These solutions are computed regardless of the scattering properties of the layer: the RTE is solved by first calling an eigensolver routine for homogeneous solutions, and then by solving a linear algebra or Green's function problem to determine solar-forcing particular solutions. If there is no scattering for a given Fourier component m and layer n, then the RTE solution is trivial – it is just the extinction across the layer with transmittance factors $T_n(\mu) = \exp[-\Delta_n/\mu]$, where μ is any polar direction and Δ_n is the layer optical thickness.

The "solution saving" option is to skip numerical computations of homogeneous and particular solutions in the absence of scattering. In this case, if there are N discrete ordinates μ_j in the half-space, then the jth homogeneous solution vector $\mathbf{X}_j$ is trivial: it has components $\{\mathbf{X}_j\}_k = \delta_{jk}$. Separation constants are μ_j^{-1}, with whole-layer transmittances given by $T_n(\mu_j)$. Particular solution vectors are set to zero, since there is no solar beam scattering. Source function integration required for post-processing the solution at arbitrary polar direction is then a simple transmittance recursion using factors $T_n(\mu)$. Linearizations (optical parameter derivatives) of RTE solutions in any non-scattering layer are zero, and linearized solutions in adjacent scattering layers will be transmitted with factors $T_n(\mu)$. We note that if this transmittance propagation passes through layer n for which a linearization $\mathcal{L}[\Delta_n]$ exists, then the linearization will pick up an additional term $\mathcal{L}[T_n(\mu)] = -\mu^{-1} T_n(\mu) \mathcal{L}[\Delta_n]$.

Rayleigh scattering has a $P(\Theta) \sim \cos^2\Theta$ phase function dependency on scattering angle Θ. There is no scattering for Fourier components $m > 2$; solution saving then applies to "Rayleigh layers" for $m > 2$. For an atmosphere with Rayleigh scattering and a limited number of aerosol or cloud layers, there will be a substantial reduction in RTE solution computations is the solution saving option applies, and consequently a marked improvement in performance. In general, the phase function has a Legendre polynomial expansion $\Phi(\Theta) \sim \sum \beta_\lambda P_\lambda(\cos\Theta)$ in terms of moment coefficients β_λ. For a discrete ordinate solution with N streams, the phase function is truncated: β_{2N-1} is the last usable coefficient in the multiple scatter solution. In the delta-M approximation, β_{2N} is used to scale the problem and redefine the β_λ for $0 \leq \lambda \leq 2N-1$. Solution saving occurs when $\beta_\lambda = 0$ for $m \leq \lambda \leq 2N-1$; there is then no scattering for Fourier component m and higher.

3.3.5 BVP telescoping

For some Fourier component m, we consider a single active layer with non-trivial RTE solutions; all other atmospheric layers have no scattering (the extension to a number of *adjacent* active layers is easy). Integration constants $\mathbf{L}_n$ and $\mathbf{M}_n$ in layer n are given through

$$I^{\pm}(x, \mu_i) = \sum_{\alpha=1}^{N} \left[L_{n\alpha} X_{in\alpha}^{\pm} e^{-k_{n\alpha} x} + M_{n\alpha} X_{in\alpha}^{\mp} e^{-k_{n\alpha}(\Delta_n - x)} \right] + G_{in}^{\pm} e^{-x/\mu_0}. \tag{3.3.3}$$

Here, $\mathbf{X}_{n\alpha}$ and $k_{n\alpha}$ denote the homogeneous solution vectors and separation constants respectively, $\mathbf{G}_n$ are the solar source vectors (a plane-parallel solution has been assumed). The boundary value problem (BVP) for the entire atmosphere is posed by compiling boundary conditions *at all levels* to create a large sparse linear algebra system. The BVP matrix has size $2NS$, where S is the total number of layers, and although there are band compression algorithms

in place to aid with the LU-decomposition and inversion of this matrix, the BVP solution step is the most expensive CPU process in the LIDORT code.

Let us assume that n is an "active" layer with scattering particles in what is otherwise a non-scattering Rayleigh atmosphere. Then we have $X_{ip\alpha}^{\pm} = \delta_{i\alpha}$ and $G_{ip}^{\pm} = 0$ for all Fourier $m > 2$ and for all layers $p \neq n$. In this case the downwelling and upwelling solutions are:

$$I_{pj}^{\downarrow}(x) = L_{pj}\exp[-x/\mu_j];\tag{3.3.4a}$$

$$I_{pj}^{\uparrow}(x) = M_{pj}\exp[-(\Delta_p - x)/\mu_j].\tag{3.3.4b}$$

Boundary value constants will clearly propagate upwards and downwards through all these non-scattering layers via:

$$L_{p+1,j} = L_{pj}\exp[-\Delta_p/\mu_j];\tag{3.3.5a}$$

$$M_{p-1,j} = M_{pj}\exp[-\Delta_p/\mu_j].\tag{3.3.5b}$$

If we can find BVP coefficients $\mathbf{L}_n$ and $\mathbf{M}_n$ for the active layer n, then coefficients for all other layers will follow by propagation. We now write down the boundary conditions for layer n. At the top of the active layer, we have:

$$\sum_{\alpha=1}^{N}\left[L_{n\alpha}X_{in\alpha}^{+} + M_{n\alpha}X_{in\alpha}^{-}\Theta_{n\alpha}\right] + G_{in}^{+} = L_{n-1,i}C_{n-1,i};\tag{3.3.6a}$$

$$\sum_{\alpha=1}^{N}\left[L_{n\alpha}X_{in\alpha}^{-} + M_{n\alpha}X_{in\alpha}^{+}\Theta_{n\alpha}\right] + G_{in}^{-} = M_{n-1,i}.\tag{3.3.6b}$$

At the bottom of the active layer, we have

$$\sum_{\alpha=1}^{N}\left[L_{n\alpha}X_{in\alpha}^{+}\Theta_{n\alpha} + M_{n\alpha}X_{in\alpha}^{-}\right] + G_{in}^{+}\Lambda_{n\alpha} = L_{n+1,i};\tag{3.3.7a}$$

$$\sum_{\alpha=1}^{N}\left[L_{n\alpha}X_{in\alpha}^{-}\Theta_{n\alpha} + M_{n\alpha}X_{in\alpha}^{+}\right] + G_{in}^{-}\Lambda_{n\alpha} = M_{n+1,i}C_{n+1,j}.\tag{3.3.7b}$$

We have used the following abbreviations:

$$\Theta_{n\alpha} = \exp[-k_{n\alpha}\Delta_n], \ \ \Delta_n = \exp[-\eta_n\Delta_n], \ \ C_{nj} = \exp[-\Delta_n/\mu_j].\tag{3.3.8}$$

We now consider the top and bottom of atmosphere boundary conditions. At TOA, there is no diffuse radiation, so that $\mathbf{L}_p = 0$ for $p = 1$ and hence by Eq. (3.3.5a) also for all $p < n$. At BOA, the Lambertian reflection condition only applies to Fourier $m = 0$; for all other components there is no reflection, and so in our case $\mathbf{M}_p = 0$ for $p = S$ and hence by Eq. (3.3.5b) also for all $p > n$. With these conditions, Eqs. (3.3.6a) and (3.3.6b) become:

$$\sum_{\alpha=1}^{N}\left[L_{n\alpha}X_{in\alpha}^{+} + M_{n\alpha}X_{in\alpha}^{-}\Theta_{n\alpha}\right] = -G_{in}^{+};\tag{3.3.9a}$$

$$\sum_{\alpha=1}^{N}\left[L_{n\alpha}X_{in\alpha}^{-}\Theta_{n\alpha} + M_{n\alpha}X_{in\alpha}^{+}\right] = -G_{in}^{-}\Lambda_{n\alpha}.\tag{3.3.9b}$$

This is a 2N system for the desired unknowns $\mathbf{L}_n$ and $\mathbf{M}_n$ (there is actually no band-matrix compression for a single layer). For the layer immediately above n, we use (3.3.6b) to find $\mathbf{M}_{n-1}$ and for remaining layers to TOA we use (3.3.5b). Similarly for the layer immediately below n, we use (3.3.7a) to find $\mathbf{L}_{n+1}$ and for remaining layers to BOA, we use (3.3.5a). This completes the BVP telescoping process.

If the telescoped BVP is written as $\mathbf{AY}=\mathbf{B}$, then the corresponding linearized problem may be written $\mathbf{A}\mathcal{L}_k[\mathbf{Y}]=\mathbf{B^\star}=\mathcal{L}_k[\mathbf{B}]-\mathcal{L}_k[\mathbf{A}]\mathbf{Y}$; the k subscript refers to the layer for which weighting functions are required. The latter is essentially the same problem with a different source vector, and the solution may be found be back-substitution, since the matrix inverse $\mathbf{A}^{-1}$ is already known from the original BVP solution. Construction of the source vector $\mathbf{B^\star}$ depends on the RTE solution linearizations; clearly if $k = n$ there will be more contributions to consider than if $k < n$. However the linearized boundary conditions for $\mathbf{B^\star}$ are essentially the same as those noted for the full atmosphere problem – the only thing to remember is that the upper boundary is the same as TOA but with the first layer active, and the lower boundary is the same as BOA but with the last layer active.

**NOTE**: this treatment assumes a Lambertian surface, for which the $m = 0$ Fourier calculation provides the only non-vanishing surface contribution; in other words, there is no surface reflection for $m > 0$. The BVP telescoping theory has been extended to non-Lambertian surfaces, but has not been implemented in the present release. If the scattering layers show irregularity in any way, the telescoping option is turned off and the boundary value problem reverts to its full form. Verification of these performance enhancements is done by comparing results with the full calculation in the absence of solution saving and BVP telescoping: the results should be identical.

**NOTE (Version 2.4)**: The above treatment allowed for telescoping to be done for a single active scattering layer in the atmosphere; this has been generalized in VLIDORT Versions 2.4 and higher to allow telescoping for any contiguous set of active scattering layers. This generalization also applies to the scalar LIDORT code (Version 3.3 and higher).

3.3.6 Convergence with exact single scatter and direct-bounce contributions

The Nakajima-Tanaka TMS correction [*Nakajima and Tanaka*, 1988] has been a feature of LIDORT and VLIDORT from the outset. In essence, the correction involves an accurate or "exact" calculation of the single scatter contribution using the full non delta-M scaled phase function (which can be expressed as a non-truncated Legendre-polynomial expansion), and with certain scaling factors on the single scatter albedos and optical thickness values depending on the application of the delta-M scaling. This correction replaces the truncated single scatter terms that would emerge from the post-processed solution of the discrete ordinate field. In the DISORT code, TMS is implemented by first taking away the truncated SS term from the already-computed overall field, and replacing it with the exact term: $\mathbf{I}' = \mathbf{I} + \mathbf{I_{SSexact}} - \mathbf{I_{SStrunc}}$; Fourier convergence is applied to $\mathbf{I}$. In VLIDORT, the truncated SS term is simply omitted from the start, with only the diffuse field being computed: $\mathbf{I}' = \mathbf{I_{mult}} + \mathbf{I_{SSexact}}$, with Fourier convergence applied only to the diffuse term $\mathbf{I_{mult}}$. Convergence is faster with the smoother and less peaked diffuse field, and the number of separate Fourier terms can be reduced by up to a third in this manner.

In earlier versions of VLIDORT, the converged diffuse field was established first, with the TMS exact scatter term applied afterwards as a correction. Following discussions with Mick Christi, it is apparent that an improvement in Fourier convergence can be obtained by first applying the TMS correction $I_{SSexact}$ and including it right from the start in the convergence testing. The rationale here is that the overall field has a larger magnitude with the inclusion of the $I_{SSexact}$ offset, so that the addition of increasingly smaller Fourier terms will be less of an influence on the total. This feature has now been installed in VLIDORT (Versions 2.1 and higher).

It turns out that a similar consideration applies to the direct-bounce intensity field (the direct solar beam reflected off the surface, with no atmospheric scattering). For a non-Lambertian surface with known BRDF, VLIDORT will calculate the upwelling field for each Fourier component - this includes the post-processed direct-bounce reflection as well as the diffuse and single scatter contributions. The complete Fourier-summed direct-bounce contribution is necessarily truncated because of the discrete ordinate approximation. It is possible to compute an accurate direct-bounce BRDF contribution for the direct beam, using the original viewing angles, and this $I_{DBexact}$ term will then replace the truncated contribution $I_{DBtrunc}$.

This "direct-bounce" feature has now been installed in Versions 2.1 and higher and functions in the same way as the TMS correction: the truncated form $I_{DBtrunc}$ is simply not calculated, and the exact form $I_{DBexact}$ is computed right from the start as an initial correction, and included in the convergence testing along with the TMS correction $I_{SSexact}$. For sharply peaked strong BRDF surface contributions, this "DB correction" can be significant, and may give rise to a substantial saving in Fourier computations, particularly for situations where the atmospheric scattering may be quite well approximated by a low number of discrete ordinates.

3.3.7 Enhanced efficiency for observational geometry output

In "operational" environments such as satellite atmospheric or surface retrieval algorithms, there is a common requirement for radiative transfer output at specific "solar zenith angle, viewing angle, relative azimuth angle" *observational geometry triplets*. Although VLIDORT has long had the capability for multi-geometry output, this capability has not been efficient for generating output at observational geometry triplets. For example, if there are 4 such triplets, then previously VLIDORT was configured to generate 4x4x4=64 output radiances, that is, one RT output for each of the 4 solar zenith angles, each of the 4 viewing angles, and each of the 4 relative azimuth angles. One may view this as computing a "4x4x4 lattice cube of solutions", and this is fine for building a look-up table (LUT). However for triplet output with 4 SZA values, we require only those solutions along the diagonal of this "lattice cube of solutions" (i.e. 4 instead of 64, one for each triplet); the other 60 solutions are redundant.

To enhance computational performance, VLIDORT has been given an *observational geometry* facility to bypass this redundancy. This facility is configured with a Boolean flag, a specific number of geometry triplets and the triplet angles themselves. In this configuration, a single call to VLIDORT will generate the discrete-ordinate radiation fields for each SZA in the given triplet set, and then carry out post-processing only for those viewing zenith and relative azimuth angles uniquely associated with the triplet SZA. One of the big time savings here is with the internal geometry routines in VLIDORT - in our example, we require 4 calls instead of 64.

Tables 3.3-3.5 give an idea of the improved efficiency gained by using this observational geometry feature ("ObsGeo") for a set of geometries, in lieu of doing a "Lattice" computation for the same set of geometries. The efficiency in each entry is given as the ratio of two CPU times: (ObsGeo time / Lattice time)*100%. Tests were made for several values of NSTREAMS, the number of half-space discrete ordinates (computational streams). The atmosphere/surface

scenario in these tests is that used in the standard environment wrappers that come with the VLIDORT package: a 23-layer atmosphere with aerosol in the lowest 6 layers and a Lambertian surface (see section 6.2 for details). Table 3.3 refers to efficiency of intensity-only computations, whereas in Table 3.4 timings were compared for calculations with intensity, two column Jacobians and one surface Jacobian; in Table 3.5 calculation timings for intensity, three profile Jacobians and one surface Jacobian were compared.

Table 3.3 Efficiency of ObsGeo vs. Lattice computations for intensity (% ratio of CPU values).

# of geometries	*# of computational streams (half-space)*			
	2	4	6	8
1	101.2	99.7	100.2	100.4
2	95.4	96.0	98.2	88.6
3	85.6	88.8	94.0	95.1
4	72.0	75.9	83.7	87.0
5	60.0	69.8	84.8	85.6
6	52.1	68.4	80.7	83.7
7	41.2	62.8	77.3	82.4
8	34.3	57.4	73.5	80.8

Table 3.4 Efficiency of ObsGeo vs. Lattice computations for intensity, 2 atmospheric column Jacobians and 1 surface Jacobian (% ratio of CPU values).

# of geometries	*# of computational streams (half-space)*			
	2	4	6	8
1	101.5	100.0	100.1	100.4
2	90.8	92.3	96.7	88.4
3	75.7	81.6	86.1	89.4
4	59.9	66.0	79.4	86.6
5	46.5	66.0	76.5	81.3
6	36.0	54.9	70.2	77.9
7	28.7	47.6	64.6	74.8
8	23.0	41.5	59.0	71.7

Table 3.5 Efficiency of ObsGeo vs. Lattice computations for intensity, 3 atmospheric profile Jacobians and 1 surface Jacobian (% ratio of CPU values).

# of geometries	*# of computational streams (half-space)*			
	2	4	6	8
1	102.7	99.9	100.2	100.3
2	88.5	88.2	94.0	86.1
3	73.7	77.3	83.3	83.2
4	59.7	64.1	71.6	75.7
5	48.6	55.3	67.4	71.7
6	39.1	49.6	61.8	67.8
7	31.7	44.1	57.1	62.9
8	25.9	39.0	51.8	59.5

3.3.8 Model upgrades to ensure thread safety in OpenMP

VLIDORT Version 2.7 has the capability to run in the OpenMP distributed parallel-computing system. This development has necessitated a few important structural changes to the software. In VLIDORT, this required the removal of many "SAVE" statements as the arrays that have this attribute are shared among the parallel OpenMP threads and can be a source anomalous computational behavior.

The model has two additional inputs that are OpenMP related. These are purely for debug purposes and the user should ignore them - they are the actual thread number and TID (thread identification index). See section 6.2.2 for a description of the environment programs in the VLIDORT package used for testing VLIDORT in an OpenMP environment and additional comments.

3.4 Taylor series expansions in VLIDORT

We now turn to the use of Taylor series expansions for situations where numerical instability may be present in solutions of the RTE. In particular we consider several "multipliers" which occur in the RT derivation of whole-layer solutions to various components of the RT field. These multipliers arise from optical-thickness integrations of solutions of the homogeneous RTE, layer solutions of the single-scattering field, and layer solutions of the diffuse source term field (the latter derived through Green's function methods in LIDORT).

The first two multipliers occur in both LIDORT and VLIDORT, but at the present time, only the scalar model LIDORT has a complete Green's function RTE solution for diffuse-field source terms. Although in general, VLIDORT solves the particular integral RTE using a substitution method, the Green's function formulation has been applied to the solution of the azimuth-independent vector RTE in a medium with Rayleigh scattering only.

3.4.1 Multipliers for the intensity field

We first look at the homogeneous-solution and primary-scatter *downwelling* multipliers, and the Green's function (downwelling) multiplier for the diffuse source term at discrete-ordinate polar cosines. These are respectively:

$$H_j^{\downarrow} = \frac{(e^{-\Delta k_j} - e^{-\Delta \mu^{-1}})}{\mu^{-1} - k_j}; \ F^{\downarrow} = \frac{(e^{-\Delta \mu^{-1}} - e^{-\Delta \lambda})}{\lambda - \mu^{-1}}; \ G_j^{\downarrow} = \frac{(e^{-\Delta k_j} - e^{-\Delta \lambda})}{\lambda - k_j} \tag{3.4.1}$$

Here, λ is the layer average secant corresponding to spherical-atmosphere attenuation of a solar beam with solar zenith angle cosine μ_0 ($\lambda = 1/\mu_0$ in plane-parallel attenuation), k_j is the separation constant corresponding to discrete ordinate stream j arising from solution of the homogeneous RTE eigenproblem, μ is the cosine of the line-of sight viewing polar angle, and Δ is the layer total optical depth. [Layer index n is suppressed for clarity].

Remark. For polarized RT with VLIDORT, some of the separation constants k_j may be complex variables; λ and μ are always real-valued. Taylor series expansions involving k_j are only applicable for real values of this quantity.

We consider the limiting cases $|\lambda - k_j| \to 0$, $|\mu^{-1} - k_j| \to 0$ or $|\lambda - \mu^{-1}| \to 0$. Writing ϵ for any of these quantities, if the order of the Taylor-series expansion is M, then we neglect terms $O(\epsilon^{M+1})$. Considering first the multiplier $G_j^{\downarrow}$ from Eq. (3.4.1), then if $\epsilon = \lambda - k_j$, we have $e^{-\Delta k_j} = We^{\epsilon \Delta} \approx W\mathbf{z}_{M+1}(\Delta)$, where we have written $W = e^{-\Delta \lambda}$ for convenience, and we

have used the notation $\mathbf{z}_{M+1}(x) = \sum_{m=0}^{M+1} z_m(x)\epsilon^m$ to approximate the exponential $e^{\epsilon\Delta}$, so that the coefficients are $z_0(x) = 1$, $z_m(x) = x^m/m!$ for $0 < m \leq M + 1$. Applying the expansion, we find:

$$G_j^{\downarrow} \approx \frac{W(\mathbf{z}_{M+1}(\Delta)-1)}{\epsilon} = \Delta W \mathbf{z}_M^*(\Delta). \tag{3.4.2}$$

Here, the new coefficients are $z_0^* = 1$, $z_m^* = \Delta^m/(m + 1)!$ for $0 < m \leq M$. Note that we need to expand first to order $M + 1$ to ensure that the final expression is defined to order M.

We continue to use this notation in the sequel. Next we look at those two multipliers for the Green's function post-processed field which may be susceptible to instability through use of the difference $\lambda - k_j$ appearing in the denominator. From the theory [*Spurr*, 2002] (see also Chapter 2), these multipliers are:

$$M_j^{\uparrow\uparrow} = \frac{H_j^{\uparrow}-F^{\uparrow}}{\lambda-k_j}; \quad M_j^{\downarrow\downarrow} = \frac{H_j^{\downarrow}-F^{\downarrow}}{\lambda-k_j}, \tag{3.4.3}$$

where $H_j^{\downarrow}$ and $F^{\downarrow}$ have been defined in Eq. (3.4.1), and

$$H_j^{\uparrow} = \frac{(1-e^{-\Delta k_j}e^{-\Delta\mu^{-1}})}{\mu^{-1}+k_j}; \quad F^{\uparrow} = \frac{(1-e^{-\Delta\mu^{-1}}e^{-\Delta\lambda})}{\lambda+\mu^{-1}}; \tag{3.4.4}$$

Looking at $M_j^{\uparrow\uparrow}$ in Eq. (3.4.3), we set $k_j = \lambda - \epsilon$, and find

$$M_j^{\uparrow\uparrow} \approx \frac{Y}{\epsilon}\left[\left(1 - WU\mathbf{z}_{M+1}(\Delta)\right)\mathbf{a}_{M+1}(Y) - (1 - WU)\right], \tag{3.4.5}$$

where $U = e^{-\Delta\mu^{-1}}$, $Y = (\lambda + \mu^{-1})^{-1}$, and series $\mathbf{a}_{M+1}(x)$ has coefficients $a_0(x) = 1$, $a_m(x) = x^m$ for $0 < m \leq M + 1$. Since $a_0 = z_0 = 1$, it is apparent that the $(1 - WU)$ contributions will fall out, and the result can then be written:

$$M_j^{\uparrow\uparrow} \approx Y[Y\mathbf{a}_M(Y) - WU\mathbf{c}_M(\Delta, Y)], \tag{3.4.6}$$

Here $\mathbf{c}_M(\Delta, Y)$ has coefficients $c_0 = 1$, and $c_m(\Delta, Y) = \sum_{p=0}^{m} z_p(\Delta)a_{m-p}(Y)$ for $0 < m \leq M$. We have found that use of these series coefficients is convenient for computation, allowing us to generate expressions to any order of accuracy without complicated algebraic expressions.

The other multiplier $M_j^{\downarrow\downarrow}$ in Eq. (3.4.3) may be treated similarly. Note that multipliers in Eq. (3.4.4) are stable entities, along with the other Green's function multipliers $G_j^{\uparrow}$, $M_j^{\downarrow\uparrow}$ and $M_j^{\uparrow\downarrow}$, the latter three quantities being defined with denominator $\lambda + k_j$.

These multipliers are required for output of the upwelling and downwelling radiance fields at layer boundaries. VLIDORT has the ability to generate output at any level away from layer boundaries (the "partial-layer" option). In this case, source function integration to an optical thickness $\tau < \Delta$ in layer n will result in "partial-layer" multipliers similar to those already defined here. For example,

$$H_j^{\downarrow}(\tau) = \frac{(e^{-\tau k_j}-e^{-\tau\mu^{-1}})}{\mu^{-1}-k_j}; \quad F^{\downarrow}(\tau) = \frac{(e^{-\tau\mu^{-1}}-e^{-\tau\lambda})}{\lambda-\mu^{-1}}; \quad G_j^{\downarrow}(\tau) = \frac{(e^{-\tau k_j}-e^{-\tau\lambda})}{\lambda-k_j} \tag{3.4.7}$$

are the downwelling partial-layer multipliers equivalent to those in Eq. (3.4.1). Taylor series expansions for these and the corresponding Green's function multipliers have been generated in a similar fashion.

3.4.2 Linearized Multipliers

VLIDORT is a linearized RT model with the ability to generate analytically both profile and total-column Jacobians with respect to any atmospheric quantity. The entire discrete ordinate solution is analytically differentiable [*Spurr*, 2002] with respect to these variables, and this includes the multipliers discussed above. It is therefore necessary to develop Taylor series expansions for those *linearized* multipliers which are susceptible to instability as noted above.

We assume the choice of layer n is given. We start with partial derivatives $\dot{k}_j \equiv \partial k_j / \partial \xi$, $\dot{\lambda} \equiv \partial \lambda / \partial \xi$ and $\dot{\Delta} \equiv \partial \Delta / \partial \xi$ with respect to some atmospheric quantity ξ. Since the layer optical depth Δ is an intrinsic optical property (input to VLIDORT), its derivative is also an intrinsic input. The viewing angle cosine μ has no derivative. Note that $\dot{k}_j = 0$ and $\dot{\Delta} = 0$ for a *profile* atmospheric quantity ξ_m defined in layer $m \neq n$ (no cross-layer derivatives); on the other hand, the average secant λ will have cross-layer derivatives for layers $m < n$, thanks to solar beam attenuation through the atmosphere. Considering the third multiplier in Eq. (3.4.1), its derivative is

$$\frac{\partial G_j^{\downarrow}}{\partial \xi} = \frac{-e^{-\Delta k_j}(\dot{k}_j \Delta + k_j \dot{\Delta}) + e^{-\Delta \lambda}(\dot{\lambda}\Delta + \lambda \dot{\Delta}) - G_j^{\downarrow}(\dot{\lambda} - \dot{k}_j)}{\lambda - k_j}. \tag{3.4.8}$$

Expanding Eq. (3.4.8) as a Taylor series with $k_j = \lambda - \epsilon$, and using the series-coefficient notation developed in the previous section, we find:

$$\frac{\partial G_j^{\downarrow}}{\partial \xi} \approx \frac{-(\dot{k}_j \Delta + \lambda \dot{\Delta} + \epsilon \dot{\Delta})W\mathbf{z}_{M+1} + (\dot{\lambda}\Delta + \lambda \dot{\Delta})W - (\dot{\lambda} - \dot{k}_j)\Delta W\mathbf{z}_{M+1}^*}{\epsilon}. \tag{3.4.9}$$

Again, $W = e^{-\Delta \lambda}$, and the series $\mathbf{z}_{M+1}$ and $\mathbf{z}_{M+1}^*$ have argument Δ and were defined in the previous section. Since $z_0 = z_0^* = 1$, the lowest-order terms in the numerator of Eq. (3.4.9) will fall out, and we are left with:

$$\frac{\partial G_j^{\downarrow}}{\partial \xi} \approx -W\left[\dot{\Delta}\mathbf{z}_M + \left(\dot{k}_j \Delta + \lambda \dot{\Delta}\right)\Delta \mathbf{z}_M^* + (\dot{\lambda} - \dot{k}_j)\Delta^2 \mathbf{z}_M^{\dagger}\right], \tag{3.4.10}$$

where the third series $\mathbf{z}_M^{\dagger}$ has coefficients $z_0^{\dagger} = 1/2$ and $z_m^{\dagger} = \Delta^m/(m+2)!$ for $0 < m \leq M$. Note that the presence of the series $\mathbf{z}_M^{\dagger}$ to order M implies that the original series must be computed to order $M + 2$, that is, we require $\mathbf{z}_{M+2}$.

Linearization of the Green's function multipliers in Eq. (3.4.3) follows similar considerations. We give one example; explicit differentiation of $M_j^{\uparrow\uparrow}$ yields

$$\frac{\partial M_j^{\uparrow\uparrow}}{\partial \xi} = \frac{\dot{H}_j^{\uparrow} - \dot{F}^{\uparrow} - (\dot{\lambda} - \dot{k}_j)M_j^{\uparrow\uparrow}}{\lambda - k_j}; \tag{3.4.11}$$

with

$$\dot{H}_j^{\uparrow} = \frac{Ue^{-\Delta k_j}[\dot{\Delta}(\mu^{-1} + k_j) + \dot{k}_j\Delta] - \dot{k}_j H_j^{\uparrow}}{\mu^{-1} + k_j}; \tag{3.4.12}$$

$$\dot{F}^{\uparrow} = \frac{UW[\dot{\Delta}(\mu^{-1} + \lambda) + \dot{\lambda}\Delta] - \dot{\lambda}F^{\uparrow}}{\lambda + \mu^{-1}}. \tag{3.4.13}$$

Expanding in the usual manner and employing results established already, we find

$$\dot{H}_j^\uparrow \approx \left[UW\left(\dot{k}_j\Delta + \dot{\Delta}Y^{-1} - \epsilon\dot{\Delta} \right)\mathbf{z}_N - \dot{k}_jY(1 - WU\mathbf{z}_N)\mathbf{a}_N \right]Y\mathbf{a}_N; \tag{3.4.14}$$

$$\dot{F}^\uparrow = \left[UW\left(\dot{\lambda}\Delta + \dot{\Delta}Y^{-1} \right) - \dot{\lambda}Y(1 - WU) \right]Y; \tag{3.4.15}$$

where now $N = M + 2$ terms in the series have been retained in the expansions. Eq. (3.4.11) implies that we also require the original multiplier $M_j^{\uparrow\uparrow}$ expanded to order $M + 1$, that is,

$$M_j^{\uparrow\uparrow} \approx Y[Y\mathbf{a}_{M+1}(Y) - WU\mathbf{c}_{M+1}(\Delta, Y)], \tag{3.4.16}$$

should be used in Eq. (3.4.11). Putting together the above three equations, and canceling out the lowest-order terms in the expansions, we find after some algebra that

$$\frac{\partial M_j^{\uparrow\uparrow}}{\partial \xi} \approx Y\left[WU\mathbf{S}_M^{(1)} - \mathbf{S}_M^{(2)} - \mathbf{S}_M^{(3)} \right], \tag{3.4.17}$$

where the three series $\mathbf{S}_M^{(q)} = \sum_{m=0}^M s_m^{(q)}\epsilon^m$ have coefficients

$$s_m^{(1)} = \left(\dot{\Delta}Y^{-1} + \dot{k}_j\Delta \right)c_{m+1}(\Delta, Y) - \dot{\Delta}c_m(\Delta, Y); \tag{3.4.18}$$

$$s_m^{(2)} = Y\dot{k}_j[b_{m+1}(Y) + WUd_{m+1}(\Delta, Y)]; \tag{3.4.19}$$

$$s_m^{(3)} = \left(\dot{\lambda} - \dot{k}_j \right)[b_{m+2}(Y) + WUc_{m+2}(\Delta, Y)]. \tag{3.4.20}$$

Subsidiary coefficients for $0 < m \le M$ are given by $a_m = Y^m$ (series expansion of $(1 - \epsilon Y)^{-1}$), and $b_m = (m + 1)Y^m$ (series expansion of $(1 - \epsilon Y)^{-2}$). We also have the product coefficients $c_m(\Delta, Y) = \sum_{p=0}^m z_p(\Delta)a_{m-p}(Y)$ and $d_m(\Delta, Y) = \sum_{p=0}^m z_p(\Delta)b_{m-p}(Y)$ obtained from the first exponential series $\mathbf{z}(\Delta)$ which approximates $e^{\Delta\epsilon}$.

Derivatives of the other multipliers subject to possible instability may be obtained similarly, and there are also derivatives of the partial-layer multipliers to be considered.

Software for all these instability cases has been compiled in two new modules (`lidort_Taylor.f90` and `vlidort_Taylor.f90` in the respective RT models) - in each case, the order M controls the accuracy of the Taylor series expansions. The other parameter controlling the use of these limiting-case calculations is the "small-number" value ϵ. VLIDORT uses double-precision floating-point arithmetic. With this in mind, we have chosen $\epsilon = 10^{-3}$ as the default, after testing linearized multiplier accuracies obtained by running the model with and without the instability corrections. In practice $M = 3$ provides more than sufficient accuracy for this choice of ϵ.

4. The VLIDORT 2.7 package

4.1 Overview

The VLIDORT "tarball" package comes as a zipped Tar file. The package directory structure is summarized in Figure 4.1. From the parent directory, there are 10 upper level subdirectories, including one for the main source code (`vlidort_main`), one for VLIDORT's Fortran 90 input/output type structure definition files (`vlidort_def`), two for scalar and vector radiative transfer testing environments (`vlidort_s_test and vlidort_v_test`), one for first order source code (`fo_main`) and one for VLIDORT supplement files (`vsup`). Object code, Fortran 90 mod files, VLIDORT package utilities and VLIDORT documentation are also stored in separate directories. We note that the subdirectory names of `vlidort_main`, `fo_main`, `vbrdf`, and `vsleave` may contain a special version designation (e.g. `vlidort_main_FO_1p4`, `fo_main_1p4`, etc…) as capability is added to these source modules over time.
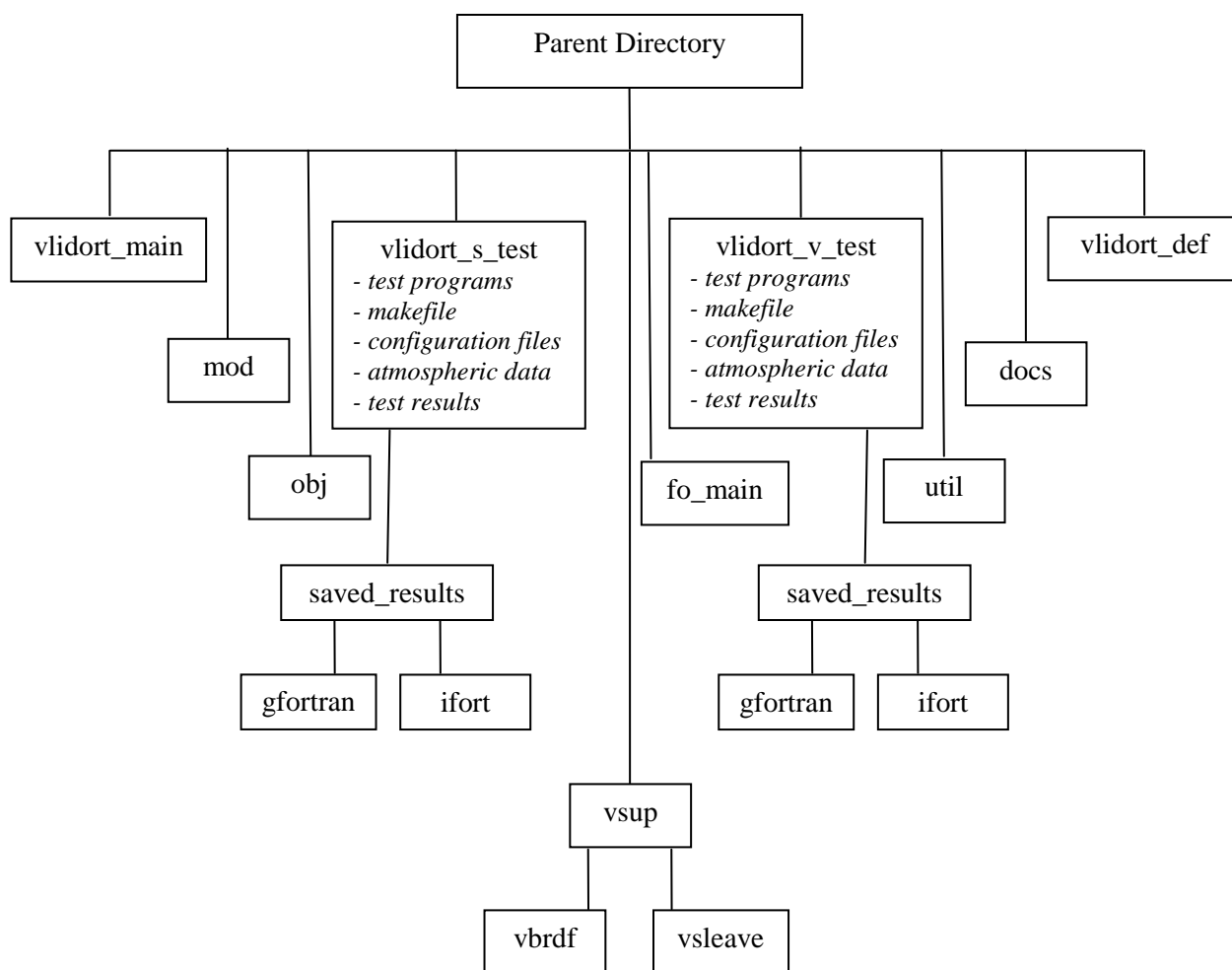
Figure 4.1. Directory structure for the VLIDORT installation package.

The test environment directories "vlidort_s_test" and "vlidort_v_test" contain several examples of calling programs for the VLIDORT code, along with associated makefiles, input configuration

files to read control options, and pre-prepared atmospheric setup data file(s) containing optical property inputs. There is also an archive of results files in both "vlidort_s_test" and "vlidort_v_test" in the subdirectory "saved_results", with which the user may compare after running the installation tests. Both the "gfortran" and "ifort" subdirectories of this subdirectory themselves contain subdirectories "obsgeo" and "nstokes3" which contain results for tests using the observational geometry feature and "NSTOKES=3" tests (not shown in the figure). Object and module files for the VLIDORT code are stored in the directories "obj" and "mod" (the "makefile" ensures this is done). As mentioned above, the VLIDORT source code is stored in subdirectories "vlidort_main" which contains the subroutines and "vlidort_def" which contains VLIDORT I/O type structure definitions along with the file "vlidort_pars.f90" of constants, dimensioning parameters and floating-point type definitions. The "docs" directory contains the VLIDORT user documentation, while directory "util" has VLIDORT package utilities. Finally, the "vsup" subdirectory contains the source code of VLIDORT supplements (currently the VBRDF and VSLEAVE supplements).

Accompanying these subdirectories are several bash shell scripts in the parent directory. These are used to run the installation tests and compare with archived results and are discussed in section 4.3.

4.2 Source code Directories

4.2.1 vlidort_def

This directory contains the following VLIDORT I/O type structure definition module files:

- vlidort_io_defs.f90
- vlidort_inputs_def.f90
- vlidort_outputs_def.f90
- vlidort_work_def.f90
- vlidort_sup_def.f90
- vlidort_sup_brdf_def.f90
- vlidort_sup_sleave_def.f90
- vlidort_sup_ss_def.f90
- vlidort_lin_io_defs.f90

- vlidort_lin_inputs_def.f90
- vlidort_lin_outputs_def.f90
- vlidort_lin_work_def.f90
- vlidort_lin_sup_def.f90
- vlidort_lin_sup_brdf_def.f90
- vlidort_lin_sup_sleave_def.f90
- vlidort_lin_sup_ss_def.f90
- vlidort_pars.f90*

*Note: there are currently four versions of this file in the "vlidort_def" subdirectory. The file labeled "vlidort_pars.f90" is the active file which contains general parameter settings; the other three such parameter files contain particular settings needed in running specific package tests.

4.2.1.1 File "vlidort_pars.f90"

Module *"vlidort_pars.f90"* contains all symbolic dimensioning parameters (integers), plus a number of fixed constants and numbers. This parameter file must be declared in every module (this includes all environment driver programs). There is a group of basic dimensioning numbers which are pre-set; this basic group is listed in Table 4.1. All other dimensioning parameters are combinations of this basic group, and are not described here (however, see the remark at the end of this section).

Table 4.1 Key parameters and dimensions in *"vlidort_pars.f90"*

Name	Type	Description
MAXSTREAMS	Dimension	Maximum number of half-space *quadrature* streams.
MAXLAYERS	Dimension	Maximum number of layers in the atmosphere.
MAXFINELAYERS	Dimension	Maximum number of fine layers per coarse layer, required for the exact single scatter ray-tracing
MAXMOMENTS_INPUT	Dimension	Maximum number of *input* scattering matrix expansion coefficients. Set to at least twice MAXSTREAMS.
MAX_THERMAL_COEFFS	Dimension	Maximum number of thermal coefficients (2)
MAX_SZANGLES	Dimension	Maximum number of solar zenith angles
MAX_USER_VZANGLES	Dimension	Maximum number of user-defined *off-quadrature* viewing zenith angles
MAX_USER_RELAZMS	Dimension	Maximum number of user-defined relative azimuth angles
MAX_USER_OBSGEOMS	Dimension	Maximum number of user-defined observational geometry angle triplets.
MAX_USER_LEVELS	Dimension	Maximum number of user-defined output levels
MAX_PARTLAYERS	Dimension	Maximum allowed number of *off-grid* (non-layer boundary) output levels. This number should always be less than MAX_USER_LEVELS.
MAX_TAYLOR_TERMS	Dimension	Maximum number of terms for Taylor series expansions.
MAXSTOKES	Dimension	Maximum number of Stokes parameters
MAX_DIRECTIONS	Dimension	Maximum number of directions (2), up/down
MAX_BRDF_KERNELS	Dimension	Maximum number of BRDF kernels
MAX_BRDF_PARAMETERS	Dimension	Maximum number of BRDF parameters allowed per kernel
MAXSTREAMS_BRDF	Dimension	Maximum number of azimuth-quadrature streams for BRDF Fourier
MAX_MSRS_MUQUAD	Dimension	Maximum number of zenith-quadrature streams for multiple scatter reflectance
MAX_MSRS_PHIQUAD	Dimension	Maximum number of azimuth-quadrature streams for multiple scatter reflectance
MAXSTREAMS_SCALING	Dimension	Maximum number of quadrature streams for internal WSA/BSA scaling.
MAX_ATMOSWFS	Dimension	Maximum number of atmospheric Jacobians
MAX_SURFACEWFS	Dimension	Maximum number of surface property Jacobians
MAX_SLEAVEWFS	Dimension	Maximum number of surface-leaving Jacobians.
MAX_MESSAGES	Dimension	Maximum number of error messages for error handling
HOPITAL_TOLERANCE	Constant	If the difference between any two polar angle cosines is less than ε, L'Hopital's Rule is invoked to avoid singularity.
OMEGA_SMALLNUM	Constant	If any total layer single scattering albedo is within ε of unity, then its value will be reset to 1-ε . Current value 10^{-15}

`MAX_TAU_SPATH,`
`MAX_TAU_UPATH,`
`MAX_TAU_QPATH`
	Constants	If the solar (S), viewing (U) or quadrature (Q) stream optical thickness exceeds the respective limit, then corresponding transmittances will be set to zero. Current values all 32.

These basic dimensioning numbers should be altered to suit memory requirements and/or a particular application. For example, if a calculation with clouds is required, allowance should be made for a large number of scattering matrix expansion coefficients and quadrature streams (discrete ordinates), so that dimensions MAXSTREAMS and MAXMOMENTS_INPUT should be increased as required. It is only necessary to go into the *"vlidort_pars.f90"* file in order to change the dimensioning parameters. Re-compilation with the makefile is then carried out to build the executable; whenever a change is made to one of the fundamental dimensions, it is recommended to use the "make clean" instruction to remove existing object, module and executable files before starting the compilation anew.

In addition to the basic dimensioning parameters, *"vlidort_pars.f90"* also contains fixed numbers such as π, 0.0, 1.0, some fixed character strings used for output formatting, and some file output numbers. A number of critical physics numbers are specified in this file. In particular, note the use of a toggle (OMEGA_SMALLNUM) to avoid the conservative scattering case when the total single scattering albedo is exactly unity.

The following five indices are also used to indicate error status for the package or any part of it:

- VLIDORT_SUCCESS = 0 (status index for a successful execution; no log-output).
- VLIDORT_DEBUG = 1 (status index for a debug execution; debug log-output).
- VLIDORT_INFO = 2 (status index for a successful execution; informational log-output).
- VLIDORT_WARNING = 3 (status index for a successful execution; warning log-output).
- VLIDORT_SERIOUS = 4 (status index for an aborted execution; failure log-output).

If the output is not completely successful in any way (status not equal to VLIDORT_SUCCESS), then the model's exception handling system will generate a number of error messages, divided into two types: (1) messages from the checking of input optical properties and control variables, and (2) messages and subroutine traces arising from a failed execution. The "Warning" status was introduced in Version 2.4 to deal with incorrect user-defined input values that can be re-set internally to allow the program to complete.

There is an additional set of indices for the BRDF kernels. Thus for a polarized Cox-Munk ocean glitter reflection, the index name is GISSCOXMUNK_IDX and has the value 10. These indices apply only to the BRDF supplement software; more details are found in section 6.

Remark (versions 2.6 and 2.7). There is a derived dimension `MAX_GEOMETRIES` which is used for the main VLIDORT output arrays in Table D2 and Tables H1-H3 in section 6.1.1. This has traditionally been set to:

`MAX_GEOMETRIES = MAX_SZANGLES*MAX_USER_VZANGLES*MAX_USER_RELAZMS`

This is suitable for lattice-option output and look-up table preparation. This number can be large and the output arrays using much memory. For the observational geometry option (section 3.3.7), `MAX_GEOMETRIES` can be set to `MAX_USER_OBSGEOMS`, and this will save memory.

4.2.1.2 Definition files – I/O type structures

In this section, we list the type structures that classify the input and output variables to the Fortran 90 code. An overview is presented in Table 4.2 below. Each type structure variable is specified by its type, assigned I/O intent, and an individual table detailing the components variables - these individual tables are found in Appendix 6.1. For the most part, the structures are based on the Fortran 77 "Include" files featured in older versions of VLIDORT. Note the structure levels cited in the second column of Table 4.2. Primary structures (level 1) are required for VLIDORT call statements whereas structures with level > 1 are embedded within their associated parent structures.

For the main VLIDORT program `vlidort_masters.f90`, the I/O variables are divided into four level-1 type structures: three input and one output. The three main input-structure variables and their component structures are specified in Tables A1-A8, B1-B7 and C1-C4 in Appendix 6.1. The main output structure and its components (which return intensities and fluxes) are listed in tables D1-D4. For calls to `vlidort_lcs_masters.f90` or `vlidort_lps_masters.f90`, we require these four structures plus four additional linearized ones: again, three input and one output. The three main linearized input structure variables and their component structures are specified in tables E1-E3, F1-F2 and G1-G4. The main linearized output structure and its components (which return the column atmospheric, profile atmospheric, general atmospheric and surface property Jacobians) are listed in tables H1-H5.

Most inputs are "Intent(in)", but a few may be modified during a VLIDORT call as the result of an internal input check - if the check fails in some manner, the code will generate a warning message that a particular input has been given a default value in order to proceed with code execution - such inputs are designated "Intent(in out)". The type structures which contain input variables which can be internally modified carry a "Modified" label in their name. All output type structure variables are "Intent(out)".

Table 4.2 Summary of VLIDORT I/O Type structures

VLIDORT I/O Type Structure	Structure Level	Intent	Table #
VLIDORT_Fixed_Inputs	1	Input	A1
VLIDORT_Fixed_Boolean	2	Input	A2
VLIDORT_Fixed_Control	2	Input	A3
VLIDORT_Fixed_Sunrays	2	Input	A4
VLIDORT_Fixed_UserValues	2	Input	A5
VLIDORT_Fixed_Chapman	2	Input	A6
VLIDORT_Fixed_Optical	2	Input	A7
VLIDORT_Fixed_Write	2	Input	A8
VLIDORT_Modified_Inputs	1	Input/Output	B1
VLIDORT_Modified_Boolean	2	Input/Output	B2
VLIDORT_Modified_Control	2	Input/Output	B3
VLIDORT_Modified_Sunrays	2	Input/Output	B4
VLIDORT_Modified_UserValues	2	Input/Output	B5
VLIDORT_Modified_Chapman	2	Input/Output	B6
VLIDORT_Modified_Optical	2	Input/Output	B7
VLIDORT_Sup_InOut	1	Input/Output	C1

VLIDORT_Sup_BRDF	2	Input	C2
VLIDORT_Sup_SS	2	Input/Output	C3
VLIDORT_Sup_SLEAVE	2	Input	C4
VLIDORT_Outputs	1	Output	D1
VLIDORT_Main_Outputs	2	Output	D2
VLIDORT_Exception_Handling	2	Output	D3
VLIDORT_Input_Exception_Handling	2	Output	D4
VLIDORT_Fixed_LinInputs	1	Input	E1
VLIDORT_Fixed_LinControl	2	Input	E2
VLIDORT_Fixed_LinOptical	2	Input	E3
VLIDORT_Modified_LinInputs	1	Input/Output	F1
VLIDORT_Modified_LinControl	2	Input/Output	F2
VLIDORT_LinSup_InOut	1	Input/Output	G1
VLIDORT_LinSup_BRDF	2	Input	G2
VLIDORT_LinSup_SS_InOut	2	Input/Output	G3
VLIDORT_LinSup_SS_Col	3	Input/Output	G3-1
VLIDORT_LinSup_SS_Prof	3	Input/Output	G3-2
VLIDORT_LinSup_SS_Surf	3	Input/Output	G3-3
VLIDORT_LinSup_SLEAVE	2	Input	G4
VLIDORT_LinOutputs	1	Output	H1
VLIDORT_LinCol	2	Output	H2
VLIDORT_LinProf	2	Output	H3
VLIDORT_LinAtmos	2	Output	H4
VLIDORT_LinSurf	2	Output	H5

Many input variables may be set by either writing explicitly coded statements in the calling program or reading entries from an ASCII-type input configuration file. In the latter case, one can use dedicated VLIDORT software to read this file. This file-read software looks for character strings which indicate the input variable or variables to be assigned. We discuss this in more detail in section 4.3.2 below. Where appropriate, all input variables are checked for consistency inside the VLIDORT package, before execution of the main radiative transfer modules.

4.2.2 vlidort_main

The main VLIDORT source code module files are listed in Table 4.3. Here, we make some notes on usage and connectivity. All subroutines start with the declaration of VLIDORT_PARS module.

The three top-level "master" module files are called from user-defined environments, and this is where the input and output are needed. All other subroutines are called from these masters. `vlidort_masters` is appropriate for the production of radiances and mean-value (flux/actinic flux) output. `vlidort_lcs_masters` is required for calculations of radiances, *column* (bulk property) atmospheric Jacobians, and surface property Jacobians. `vlidort_lps_masters` is required for calculations of radiances, *profile* atmospheric Jacobians, and surface property Jacobians. Each top-level master contains a loop over the Fourier cosine/sine azimuth series, plus the associated Fourier component subroutine.

For setting input variables for a non-Jacobian calculation, the user can invoke subroutine VLIDORT_INPUT_MASTER, which should be called in the user environment before the main call to VLIDORT_MASTERS (see below in section 4.3 for a pseudo-code example). This subroutine requires the use of a configuration file, which is read by a dedicated subroutine called in VLIDORT_INPUT_MASTER and based around the FINDPAR tool (discussed in section 4.3.2). For calculations with Jacobians using either the LCS or LPS Master module, the user can call the subroutine VLIDORT_L_INPUT_MASTER to generate inputs (including linearization control) by configuration file reading.

Module files required by all three masters.
We now give a description of the other module files in Table 4.3. All input functions are contained in `vlidort_inputs`. These are subroutines to initialize inputs and read them from file, to check the inputs for mistakes and inconsistencies, and to derive input variables for bookkeeping (for example, sorting the stream angles input, sorting and assigning masks for optical depth output).

Subroutines in `vlidort_miscsetups` are executed before the main Fourier component subroutine is called. Set-up routines include the Delta-M scaling and the preparation of all optical depth exponentials (transmittances). The solar beam Chapman function calculation for the curved atmosphere, and the ray tracing along the line of sight (required for exact single scatter corrections) are contained in `vlidort_geometry`.

Multipliers used in solving homogeneous and particular solutions of the radiative transfer problem are computed in `vlidort_multipliers`. In `vlidort_thermalsup`, subroutines for setting up thermal emission quantities and computing thermal emission solutions are found. `vlidort_taylor` contains the subroutines for applying the Taylor multipliers within VLIDORT where needed.

The module `vlidort_solutions` solves the discrete ordinate radiative transfer equation. There are subroutines for determining the eigensolutions and separation constants from the homogeneous equation, plus the particular integral for the solar source. `vlidort_bvproblem` applies the boundary value conditions in a multi-layer atmosphere with reflecting surface, and solves the boundary-value problem (constants of integration) using an accelerated band-compression linear algebra method; there are also subroutines dealing with the telescoped boundary value formulation.

Table 4.3. Module files in VLIDORT main source code directory.

vlidort_masters	Intensity Only	Called from user environment
vlidort_lcs_masters	Intensity + Column & Surface Jacobians	Called from user environment
vlidort_lps_masters	Intensity + Profile & Surface Jacobians	Called from user environment
vlidort_inputs	Reads (from file) variables in some Input type structures	Contains a master routine called optionally in user environments before calls to any of the 3 masters. Also contains input checking and other routines called by all 3 masters.

vlidort_miscsetups	Set-up pseudo-spherical and transmittances	Called by all 3 Masters
vlidort_geometry	Spherical geometry	Called by all 3 Masters
vlidort_multipliers	Homogeneous & particular solution multipliers	Called by all 3 Masters
vlidort_corrections	Exact single scatter computations	Called by all 3 Masters
vlidort_thermalsup	Thermal computations	Called by all 3 Masters
vlidort_solutions	Solves RT Equations in discrete ordinates	Called by all 3 Masters
vlidort_bvproblem	Creates and Solves Boundary Value problem	Called by all 3 Masters
vlidort_intensity	Post processing of RT solution	Called by all 3 Masters
vlidort_taylor	Taylor expansion multipliers	Called by all 3 Masters
vlidort_writemodules	Writes VLIDORT I/O to files	Called by all 3 Masters
vlidort_aux	Auxiliary code (Eigensolver, Findpar, etc.)	Called by all 3 Masters
vlidort_pack	Packs certain standard internal variables into internal type structures	Called by all 3 Masters
vlidort_unpack	Unpacks certain standard internal variables from internal type structures	Called by all 3 Masters
vlidort_l_inputs	Reads (from file) variables in some Input type structures	Contains a master routine called optionally in user environments before calls to LCS or LPS Master
vlidort_la_miscsetups	Linearized pseudo-spherical and transmittances	Called by LCS or LPS Master
vlidort_la_corrections	Linearization exact single scatter computations	Called by LCS or LPS Master
vlidort_l_thermalsup	Linearized thermal computations	Called by LCS or LPS Master
vlidort_lpc_solutions	Linearized RTE solutions	Called by LCS or LPS Master
vlidort_lpc_bvproblem	Solution Linearized boundary value problems	Called by LCS or LPS Master
vlidort_l_writemodules	Writes VLIDORT linearized I/O to files	Called by LCS or LPS Master
vlidort_lbbf_jacobians	Atmospheric and surface blackbody Jacobians	Called by LCS or LPS Master
vlidort_l_pack	Packs certain linearized internal variables into internal type structures	Called by LCS or LPS Master
vlidort_l_unpack	Unpacks certain linearized internal variables from internal type structures	Called by LCS or LPS Master
vlidort_ls_corrections	Linearization of exact direct bounce reflection	Called by LCS or LPS Master
vlidort_ls_wfsurface	Post-processing of surface property Jacobians	Called by LCS or LPS Master
vlidort_ls_wfsleave	Post-processing of surface-leaving Jacobians	Called by LCS or LPS Master
vlidort_lc_miscsetups	Set-up linearization of column transmittances	Called by LCS Master
vlidort_lc_corrections	Linearization of exact single scatter solutions	Called by LCS Master
vlidort_lc_solutions	Linearized RTE solutions	Called by LCS Master

vlidort_lc_bvproblem	Solution of Linearized boundary value problems	Called by LCS Master
vlidort_lc_wfatmos	Post-processing of atmospheric Jacobians	Called by LCS Master
vlidort_lc_pack	Packs certain linearized column internal variables into internal type structures	Called by LCS Master
vlidort_lc_unpack	Unpacks certain linearized column internal variables from internal type structures	Called by LCS Master
vlidort_lp_miscsetups	Set-up linearization of profile transmittances	Called by LPS Master
vlidort_lp_corrections	Linearization of exact single scatter solutions	Called by LPS Master
vlidort_lp_solutions	Linearized RTE solutions	Called by LPS Master
vlidort_lp_bvproblem	Solution of Linearized boundary value problems	Called by LPS Master
vlidort_lp_wfatmos	Post-processing of atmospheric Jacobians	Called by LPS Master
vlidort_lp_pack	Packs certain linearized profile internal variables into internal type structures	Called by LPS Master
vlidort_lp_unpack	Unpacks certain linearized profile internal variables from internal type structures	Called by LPS Master
vlidort_get_planck	Generates Planck intensities and Jacobians	Called from user environment

In `vlidort_intensity`, we compute intensities at user-defined optical depths and stream angles; this is the post-processing (source function integration). This module also contains computations of the mean-value output (actinic and regular fluxes). The exact Nakajima-Tanaka single scatter intensity and the exact direct beam intensity are found in `vlidort_corrections`, while `vlidort_writemodules` contains subroutines to write control inputs and scene inputs received by VLIDORT and outputs generated by VLIDORT.

The `vlidort_pack` and `vlidort_unpack` modules are utility modules for packing/unpacking certain standard internal variables to/from internal type structures. Finally, in `vlidort_aux`, there are standard numerical routines for the eigenproblem solution (based on ASYMTX as used in DISORT) and Gauss quadrature evaluation. This module also contains the input file-read tool FINDPAR, and some exception handling software.

The module `lapack_tools` is a compilation of LAPACK subroutines used in VLIDORT (eigensolver DGEEV, linear algebra modules DGETRF/DGETRS and DGBTRF/DGBTRS, plus other routines). More details in Section 4.5

Module files required for Jacobian calculations.
The module `vlidort_lcs_masters` calculates column atmospheric Jacobians and surface property Jacobians in addition to the radiance and mean-value fields, while the module `vlidort_lps_masters` returns profile atmospheric Jacobians and surface property Jacobians in addition to the radiance and mean-value fields. All linearized input functions are contained in `vlidort_l_inputs`.

Module `vlidort_la_miscsetups` and `vlidort_lpc_solutions` are shared by the two linearization masters and apply to both types of atmospheric property Jacobian. The first computes linearizations of the delta-M, single-scatter albedo, and transmittance setups for each layer optical property and are called early in the main Fourier loop, while the second gives linearizations of the eigenvalue and particular integral RTE solutions. Setups and source terms required for linearized thermal computations are located in the module `vlidort_l_thermalsup`. Along with this, the module `vlidort_lbbf_jacobians` computes linearized quantities required for atmospheric and surface blackbody Jacobians.

Module `vlidort_la_corrections` and `vlidort_lpc_bvproblem` are also shared by the 2 linearization masters and apply to both types of atmospheric property Jacobian. The first computes linearizations of Z-matrices for the linearization of the single-scatter correction in different scenarios, while the second contains help subroutines used in solving the linearized boundary-value problem. Subroutines to write linearized inputs received by VLIDORT are located in `vlidort_l_writemodules`.

The complete generation of column weighting functions is governed by the following five module files, namely: `vlidort_lc_bvproblem`, `vlidort_lc_wfatmos`, `vlidort_lc_miscsetups`, `vlidort_lc_corrections`, and `vlidort_lc_solutions`. The first solves the linearized boundary-value problem (constants of integration) in a multi-layer atmosphere; this requires only the setup of linearized vectors for the L-U back-substitution (also contains modules dealing with linearization boundary value telescoping). The second is the post processing solution - generation of column Jacobians at arbitrary optical depths and user line-of-sight angles, the Fourier cosine-series convergence for these Jacobians, and the derivation of weighting functions for the mean-value fields. The third generates transmission-related quantities for the column weighting functions while the fourth generates weighting functions for the exact single scatter components of the radiation fields. The last develops the linearized beam solutions for the linearized column RT problem. These routines are only called by the master subroutine `vlidort_lcs_master`.

The complete generation of profile atmospheric weighting functions is similarly determined by five module files: `vlidort_lp_bvproblem`, `vlidort_lp_wfatmos`, `vlidort_lp_miscsetups` `vlidort_lp_corrections`, and `vlidort_lp_solutions`. These routines are only called by the master subroutine `vlidort_lps_master`.

Finally, modules `vlidort_ls_corrections`, `vlidort_ls_wfsurface`, `vlidort_ls_wfsleave` are called by either linearization master. The first linearizes the exact direct-bounce component of the reflected surface field with respect to any surface property Jacobian. The second solves the linearized boundary-value problem (constants of integration) for these surface-property Jacobians and develops the associated post-processing solution, while the last linearizes the surface-leaving radiation source (if present) with respect to selected properties(such as fluorescence magnitude) characterizing this source.

In a manner similar to the previously mentioned `vlidort_pack` and `vlidort_unpack` modules, the modules `vlidort_l_pack`, `vlidort_l_unpack`, `vlidort_lc_pack`, `vlidort_lc_unpack`, `vlidort_lp_pack`, and `vlidort_lp_unpack` are utility modules for packing/unpacking certain linearized internal variables into/from internal type structures during linearized calculations.

Module files for input preparation.

In addition to the source-code modules in directory `vlidort_main`, VLIDORT also makes use of some additional modules for performing various tasks for the preparation of input. Currently, there is just one: `vlidort_getplanck`. For a given temperature, this will generate the associated Planck black-body function and its temperature derivative. Stand-alone supplements used by VLIDORT (such as the BRDF supplement) may be found in the appendices.

4.3 Calling VLIDORT, Configuration files, Makefiles, Installation

Next, an example of a calling environment for VLIDORT is discussed in section 4.3.1, followed by some comments in section 4.3.2 regarding input configurations files. Section 4.3.3 contains some information concerning the Makefiles that come with the VLIDORT package - these are for use in a Unix/Linux operating environment. In section 4.3.4, some information regarding the installation tests that come with the VLIDORT package is supplied. In addition, we present descriptions of some simple scripts to run the installation tests in a Unix/Linux operating environment. Makefiles for handling these installation tests in the Microsoft® Windows® environment is planned. Finally, section 4.3.5 contains some helpful tips for setting VLIDORT inputs.

4.3.1 Calling environment – an example

We show how the master VLIDORT module is used within a calling environment by means of a simple example in the form of a schematic computational sequence (pseudo-code). Comment lines are prefaced by the symbol "!". This is a calling environment for a basic calculation of intensity (no Jacobians) for a number of different scenarios.

VLIDORT execution is controlled by a single subroutine `VLIDORT_MASTER`, which is called once for each scenario. In the example here, the main loop is preceded by a call to the subroutine `VLIDORT_INPUT_MASTER` in order to read the appropriate input from the configuration file `VLIDORT.inp` (passed as a subroutine argument). If the `STATUS_INPUTREAD` integer output is not equal to `VLIDORT_SUCCESS`, the program stops and the user should examine the exception-handling errors by calling the `VLIDORT_WRITE_STATUS` subroutine. *It is possible for the user to dispense with this kind of file-read input set-up and assignment, and simply assign input variables explicitly in hard-wired statements. However, this requires a certain level of confidence in the model!* In the next section, we discuss a typical configuration file.

The subroutine output `STATUS_INPUTCHECK` is available for the checking of the input data once the file-read is complete. Checking is internal to VLIDORT and is done first before any radiative transfer. If this integer output is equal to `VLIDORT_SERIOUS`, VLIDORT will exit without performing any calculations; if it equals `VLIDORT_WARNING`, the model will execute but it means that some of the input is incorrect and that VLIDORT has reverted to a default input and carried on with this default. If there is a fatal error during the execution of VLIDORT, then the model will bypass any further calculation and exit with an error message and 3 error traces to indicate the source of the error. In this case, the `STATUS_CALCULATION` integer output will have the value `VLIDORT_SERIOUS`. There are no warnings here; all errors in execution are fatal. More details on the exception handling are in section 4.5.

```
program main_VLIDORT

!  Module files for VLIDORT
   USE VLIDORT_PARS
   USE VLIDORT_IO_DEFS

   USE VLIDORT_INPUTS
   USE VLIDORT_MASTERS

!  Negate implicit typing
   implicit none

!  Status declarations
   INTEGER :: STATUS_INPUTCHECK, STATUS_CALCULATION

!  Initialize status variables to 0
   STATUS_INPUTCHECK=0; STATUS_CALCULATION=0

!  Determine File-read Control variables in Input Structures
   call VLIDORT_INPUT_MASTER &
      ('VLIDORT.inp',       & ! Input
        VLIDORT_FixIn,      & ! Outputs
        VLIDORT_ModIn,      & ! Outputs
        VLIDORT_InputStatus ) ! Outputs

!  Set number of threads (e.g. number of wavelengths)
   nthreads = 8

!  Assign Physical (Optical property) input variables for all threads:
   call USER_VLIDORT_PREPARE

!  Start thread loop; this can be put in OPEN_MP environments
   do i = 1, nthreads

!  VLIDORT master call and error check
     call VLIDORT_MASTER ( &
         VLIDORT_FixIn, &
         VLIDORT_ModIn, &
         VLIDORT_Sup,   &
         VLIDORT_Out )

     call VLIDORT_WRITE_STATUS ( &
         STATUS_FILE_NAME, &
         STATUS_FILE_UNIT, &
         STATUS_FILE_FLAG, &
         VLIDORT_Out%Status )

!  End thread or wavelength loop
   end do

!  finish
   write user-defined output arrays
   stop

end program main_VLIDORT
```

4.3.2 Configuration file discussion

In the previous section, we noted that a call to subroutine VLIDORT_INPUT_MASTER enables variables to be assigned from a configuration file of inputs. This process assigns values to most

(but not all) of the variables in the input Type Structures. The file-read is done using the FINDPAR tool in the source-code module `vlidort_aux` ; FINDPAR looks for a character string made up of a prefix (in this case the word "VLIDORT") and a text description of the variable(s) to be assigned and then reads the variable(s) specified underneath the character string. All strings ending with a question mark indicate the assignment of Boolean variables. If the character string is not present, or if the file-read itself is corrupted by bad input, then an error message is generated and a status flag set.

Tables J, K, L, and M in Appendix 6.1 contain the lists of dedicated character strings and the associated input variables. These character string tables and their associated VLIDORT input type structures are listed in Table 4.4 to give the reader an initial overview.

Examples of configuration files are found in the test directories. In the string tables, we present variables from the input tables (i.e. the A, B, E, and F Tables in Table 4.2 above) that are assigned using this file-read procedure, along with their associated character strings. It should be noted that some input variables are not file-reads (array inputs mostly). These include some of the variables in Input Tables A6, A7, B6, E2, and F2. Some variables in the control inputs are normally assigned by the user, depending on the application. It is also possible to overwrite file-read assignments, in particular for applications where the number of layers (variable "NLAYERS" in VLIDORT) will be pre-set by a call to generate atmospheric optical properties.

Table 4.4 Summary of VLIDORT configuration file tables of input strings

VLIDORT I/O Type Structure	String Table #
VLIDORT_Fixed_Boolean	J1
VLIDORT_Fixed_Control	J2
VLIDORT_Fixed_Sunrays	J3
VLIDORT_Fixed_UserValues	J4
VLIDORT_Fixed_Chapman	J5
VLIDORT_Fixed_Optical	J6
VLIDORT_Fixed_Write	J7
VLIDORT_Modified_Boolean	K1
VLIDORT_Modified_Control	K2
VLIDORT_Modified_Sunrays	K3
VLIDORT_Modified_UserValues	K4
VLIDORT_Modified_Chapman	K5
VLIDORT_Fixed_LinControl	L1
VLIDORT_Modified_LinControl	M1

4.3.3 Makefile discussion

As an example, we now describe the Makefile in the "vlidort_s_test" directory (other Makefiles are similar). The software was compiled and tested at RT Solutions using the Intel® and GNU FORTRAN compilers. The software has also been tested successfully using the Portland Group® FORTRAN 90/95 compiler (courtesy V. Natraj). The Makefile begins by defining path variables for the active directories in the installation package:

```
UTIL_PATH = util
```

```
VSUP_PATH = vsup
VLID_DEF_PATH = vlidort_def
VLID_MAIN_PATH = vlidort_main
VLID_TEST_PATH = vlidort_s_test
FO_MAIN_PATH = fo_main

MOD_PATH = mod
OBJ_PATH = obj
```

These are followed by two file variables used by the "clean" command at the bottom of the Makefile ("make clean" will empty the "mod" and "obj" directories):

```
MOD_FILES = $(MOD_PATH)/*.mod
OBJ_FILES = $(OBJ_PATH)/*.o
```

Note that all Fortran module files and compiled object files are collected in the above "mod" and "obj" subdirectories to avoid cluttering up the environment directory.

Next, a default shell variable is defined to avoid unnecessary problems that might arise if the GNU Makefile were to be run under a different command shell other than the "bash" shell

```
SHELL = /bin/bash
```

Following this, Fortran compiler variables are defined. They are actually commented out, since the current setup calls for the Fortran compiler to be supplied on the command line when the installation test script is invoked. These variables are then followed by compiler flags for several compilers used to test the VLIDORT code. For example, for the Intel® "ifort" compiler, the compiler flags are:

```
# Additional flags for Intel
ifeq ($(FC), ifort)
      FFLAGS := $(FFLAGS) -I$(MOD_PATH) -module $(MOD_PATH)
      FFLAGS_DEBUG = -g -warn all -check all -traceback
      FFLAGS_OPENMP = -openmp
      FFLAGS_OPT = -O3
endif
```

Source files are then defined for both intensity and Jacobian tests:

```
BASE_SOURCES =
SOURCES =
L_SOURCES =
LPS_SOURCES =
LCS_SOURCES =

BASE_SOURCES +=   \
   $(VLID_DEF_PATH)/vlidort_pars.f90

SOURCES +=   \
   $(BASE_SOURCES) \
   $(VLID_MAIN_PATH)/lapack_tools.f90            \
   $(VLID_DEF_PATH)/vlidort_inputs_def.f90       \
   $(VLID_DEF_PATH)/vlidort_sup_brdf_def.f90     \
```

```
    $(VLID_DEF_PATH)/vlidort_sup_ss_def.f90          \
    $(VLID_DEF_PATH)/vlidort_sup_sleave_def.f90      \
    $(VLID_DEF_PATH)/vlidort_sup_def.f90             \
    $(VLID_DEF_PATH)/vlidort_outputs_def.f90         \
    $(VLID_DEF_PATH)/vlidort_io_defs.f90             \
    $(VLID_MAIN_PATH)/vlidort_aux.f90                \
    $(VLID_MAIN_PATH)/vlidort_getplanck.f90          \
    $(VLID_MAIN_PATH)/vlidort_geometry.f90           \
    $(VLID_MAIN_PATH)/vlidort_Taylor.f90             \
    $(VLID_MAIN_PATH)/vlidort_inputs.f90             \
    $(VLID_MAIN_PATH)/vlidort_miscsetups.f90         \
    $(VLID_MAIN_PATH)/vlidort_multipliers.f90        \
    $(VLID_MAIN_PATH)/vlidort_corrections.f90        \
    $(VLID_MAIN_PATH)/vlidort_thermalsup.f90         \
    $(VLID_MAIN_PATH)/vlidort_solutions.f90          \
    $(VLID_MAIN_PATH)/vlidort_bvproblem.f90          \
    $(VLID_MAIN_PATH)/vlidort_intensity.f90          \
    $(VLID_MAIN_PATH)/vlidort_writemodules.f90       \
    $(VLID_MAIN_PATH)/vlidort_pack.f90               \
    $(VLID_MAIN_PATH)/vlidort_unpack.f90             \
    $(VLID_MAIN_PATH)/vlidort_masters.f90

L_SOURCES += \
    $(VLID_DEF_PATH)/vlidort_lin_inputs_def.f90      \
    $(VLID_DEF_PATH)/vlidort_lin_sup_brdf_def.f90    \
    $(VLID_DEF_PATH)/vlidort_lin_sup_ss_def.f90      \
    $(VLID_DEF_PATH)/vlidort_lin_sup_sleave_def.f90\
    $(VLID_DEF_PATH)/vlidort_lin_sup_def.f90         \
    $(VLID_DEF_PATH)/vlidort_lin_outputs_def.f90     \
    $(VLID_DEF_PATH)/vlidort_lin_io_defs.f90         \
    $(VLID_DEF_PATH)/vlidort_lin_work_def.f90        \
    $(VLID_MAIN_PATH)/vlidort_l_inputs.f90           \
    $(VLID_MAIN_PATH)/vlidort_la_miscsetups.f90      \
    $(VLID_MAIN_PATH)/vlidort_la_corrections.f90     \
    $(VLID_MAIN_PATH)/vlidort_ls_corrections.f90     \
    $(VLID_MAIN_PATH)/vlidort_l_thermalsup.f90       \
    $(VLID_MAIN_PATH)/vlidort_lpc_solutions.f90      \
    $(VLID_MAIN_PATH)/vlidort_lpc_bvproblem.f90      \
    $(VLID_MAIN_PATH)/vlidort_lbbf_jacobians.f90     \
    $(VLID_MAIN_PATH)/vlidort_ls_wfsurface.f90       \
    $(VLID_MAIN_PATH)/vlidort_ls_wfsleave.f90        \
    $(VLID_MAIN_PATH)/vlidort_l_writemodules.f90     \
    $(VLID_MAIN_PATH)/vlidort_l_pack.f90             \
    $(VLID_MAIN_PATH)/vlidort_l_unpack.f90

LPS_SOURCES += \
    $(VLID_MAIN_PATH)/vlidort_lp_miscsetups.f90      \
    $(VLID_MAIN_PATH)/vlidort_lp_corrections.f90     \
    $(VLID_MAIN_PATH)/vlidort_lp_solutions.f90       \
    $(VLID_MAIN_PATH)/vlidort_lp_bvproblem.f90       \
    $(VLID_MAIN_PATH)/vlidort_lp_wfatmos.f90         \
    $(VLID_MAIN_PATH)/vlidort_lp_pack.f90            \
    $(VLID_MAIN_PATH)/vlidort_lp_unpack.f90          \
    $(VLID_MAIN_PATH)/vlidort_lps_masters.f90

LCS_SOURCES += \
    $(VLID_MAIN_PATH)/vlidort_lc_miscsetups.f90      \
```

```
        $(VLID_MAIN_PATH)/vlidort_lc_corrections.f90    \
        $(VLID_MAIN_PATH)/vlidort_lc_solutions.f90      \
        $(VLID_MAIN_PATH)/vlidort_lc_bvproblem.f90      \
        $(VLID_MAIN_PATH)/vlidort_lc_wfatmos.f90        \
        $(VLID_MAIN_PATH)/vlidort_lc_pack.f90           \
        $(VLID_MAIN_PATH)/vlidort_lc_unpack.f90         \
        $(VLID_MAIN_PATH)/vlidort_lcs_masters.f90

# (Include vector supplement source files)
include $(VSUP_PATH)/makefile.vsup

# (Include first-order source files)
include $(FO_MAIN_PATH)/makefile.fo

# Main scalar tests

SOURCES_SOLAR = $(FO_SOURCES_Vector) + \
        $(SOURCES) \
        $(VLID_TEST_PATH)/2p7_solar_tester.f90

SOURCES_THERMAL = $(FO_SOURCES_Vector) + \
        $(SOURCES) \
        $(VBRDF_SUP_SOURCES) \
        $(VLID_TEST_PATH)/2p7_thermal_tester.f90

SOURCES_SOLAR_LPCS = $(FO_SOURCES_L_Vector) + \
        $(SOURCES) \
        $(L_SOURCES) \
        $(LPS_SOURCES) \
        $(LCS_SOURCES) \
        $(VLID_TEST_PATH)/2p7_solar_lpcs_tester.f90

SOURCES_THERMAL_LPCS = $(FO_SOURCES_L_Vector) + \
        $(SOURCES) \
        $(L_SOURCES) \
        $(LPS_SOURCES) \
        $(LCS_SOURCES) \
        $(VLID_TEST_PATH)/2p7_thermal_lpcs_tester.f90

SOURCES_BRDFPLUS = $(FO_SOURCES_L_Vector) + \
        $(SOURCES) \
        $(L_SOURCES) \
        $(LCS_SOURCES) \
        $(VSLEAVE_SUP_SOURCES) \
        $(VBRDF_LINSUP_SOURCES) \
        $(VLID_TEST_PATH)/vlidort_sup_accessories.f90 \
        $(VLID_TEST_PATH)/2p7_brdfplus_tester.f90
```

We also define utility programs:

```
# Utilities

SOURCES_UTIL =
SOURCES_UTIL +=    \
        $(UTIL_PATH)/vlidort_diff.f90
```

Next comes the pattern rules for creating object files:

```
# For vlidort main source files
$(OBJ_PATH)/%.o : $(VLID_DEF_PATH)/%.f90
        $(FC) $(FFLAGS) $< -o $@
$(OBJ_PATH)/%.o : $(VLID_MAIN_PATH)/%.f90
        $(FC) $(FFLAGS) $< -o $@
$(OBJ_PATH)/%.o : $(VLID_TEST_PATH)/%.f90
        $(FC) $(FFLAGS) $< -o $@

# For utility source files
$(OBJ_PATH)/%.o : $(UTIL_PATH)/%.f90
        $(FC) $(FFLAGS) $< -o $@
```

Then we have variables for defining source and object file lists. For example:

```
F90SOURCES_SOLAR := $(notdir $(filter %.f90, $(SOURCES_SOLAR)))
F90OBJECTS_SOLAR := $(patsubst %.f90, %.o, $(addprefix $(OBJ_PATH)/,
$(F90SOURCES_SOLAR)))
```

Finally, the command to build the desired target executable(s) for the installation tests is:

```
main: solar \
      thermal \
      brdf

solar: s2p7_solar_tester.exe \
       s2p7_solar_lpcs_tester.exe

thermal: s2p7_thermal_tester.exe \
         s2p7_thermal_lpcs_tester.exe

brdf: s2p7_brdfplus_tester.exe

s2p7_solar_tester.exe: $(F90OBJECTS_SOLAR)
        $(FC) $^ -o $@
s2p7_thermal_tester.exe: $(F90OBJECTS_THERMAL)
        $(FC) $^ -o $@
s2p7_solar_lpcs_tester.exe: $(F90OBJECTS_SOLAR_LPCS)
        $(FC) $^ -o $@
s2p7_thermal_lpcs_tester.exe: $(F90OBJECTS_THERMAL_LPCS)
        $(FC) $^ -o $@
s2p7_brdfplus_tester.exe: $(F90OBJECTS_BRDFPLUS)
        $(FC) $^ -o $@

vlidort_diff: $(F90OBJECTS_UTIL)
        $(FC) $^ -o $@
```

lastly, the Makefile "clean" target command is defined:

```
.PHONY: clean
clean:
        rm -f *.o $(OBJ_FILES) *.mod $(MOD_FILES) *.log *.exe
```

4.3.4 Installation and testing

To install the VLIDORT package, create a new "home" directory and unzip the VLIDORT tarball to view the list of subdirectories outlined in section 1 and Figure 4.1

Go into the "vlidort_s_test" subdirectory. There, one will find the Makefile discussed in section 4.3.3. This Makefile can build the executables for the "tester" environment programs listed in Table 4.5. There are two tests each for solar scattering and thermal emission sources; two of the tests will generate only radiances and mean-value output, while the other two will generate additionally a series of Jacobian outputs. An additional test (row 5 in Table 4.5) is designed to run VLIDORT along with standard and linearized BRDF supplement modules to obtain intensities and Jacobians in the presence of a bidirectionally reflecting lower boundary.
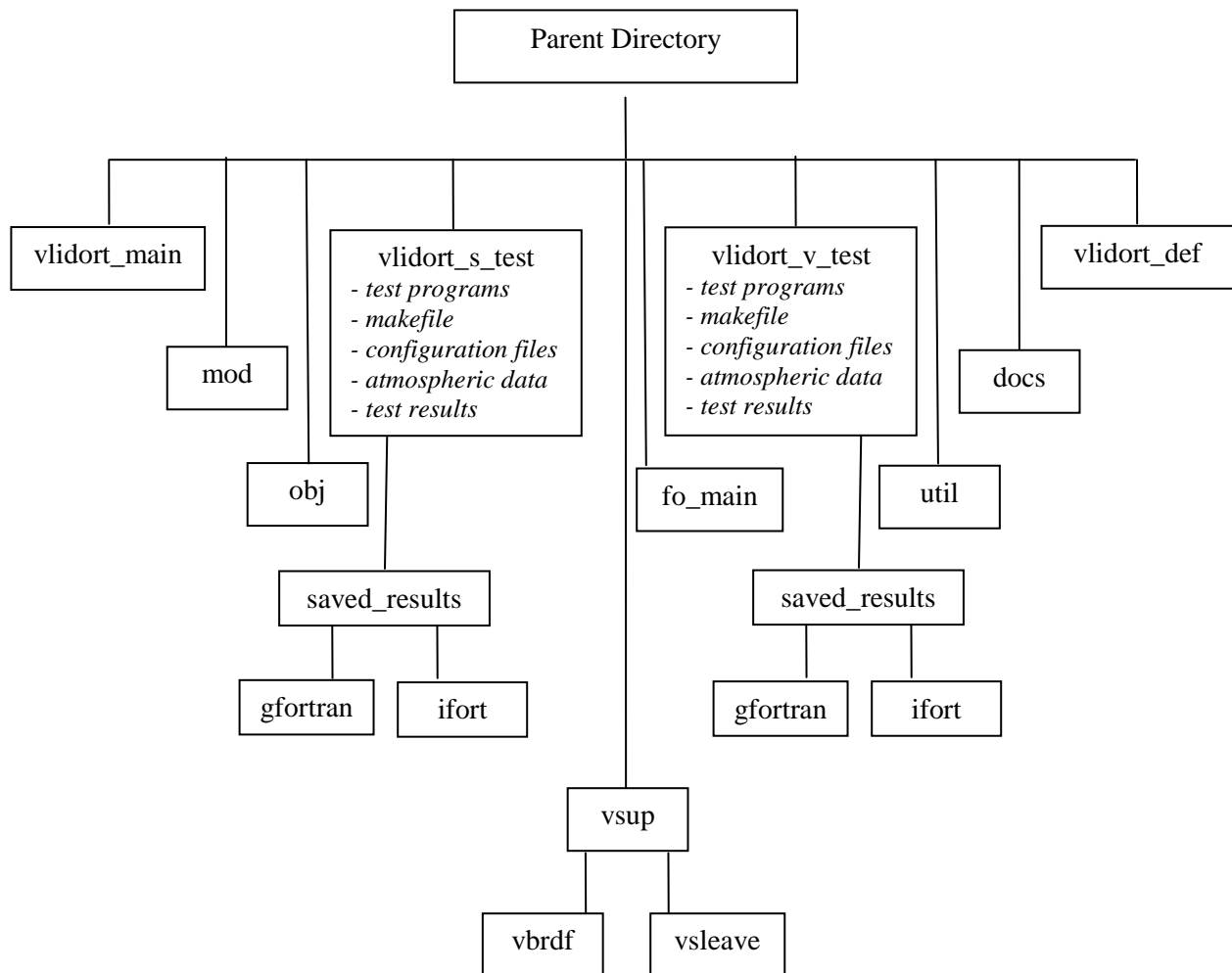
To run the programs in the scalar test directory ("vlidort_s_test"), return to the home directory in which you have installed the VLIDORT package and invoke the bash script "vlidort_run" from the command line as (using "$" as the command prompt):

    $ vlidort_run s <your_compiler>

Here, "s" indicates you want to run tests from the *scalar* test directory and <your_compiler> is the standard name used to invoke the Fortran compiler you are using (e.g. "gfortran" when using the GNU Fortran compiler). This will cause the "vlidort_run" script to generate and run each of the environment program executables in Table 4.5 below in sequence and generate the corresponding result file(s). Similarly, polarized Stokes vector tests may be run by invoking the bash script "vlidort_run" using the command:

$ vlidort_run v <your_compiler>

The only difference from the previous command is the use of the "v" parameter. In this case, a similar set of tests will be run using VLIDORT, but now with inputs more appropriate to polarized calculations (Table 4.5A, rows 1-5). In addition, there is a test ("Siewert2000") to compare with benchmark results from the radiative transfer literature [see section 3.2.3 for a discussion on this validation], a test to build an environment program that demonstrates how to use the VSLEAVE supplement code in conjunction with calls to VLIDORT, and two tests to demonstrate how VLIDORT might be used in a parallel computing context using the OpenMP environment.

Please draw attention to the "I000" designation in some of the result files in Table 4.5A. These filenames are appropriate for the five vector tests in which the NSTOKES variable is set to 1 (Q = U = 0, and only the unpolarized intensity is calculated). However, if NSTOKES is set to 3 for example, VLIDORT would calculate the I, Q, and U components of the Stokes vector and the resulting file names would indicate they contain the results related to these additional components by having an "IQU0" designation.

We note that a subset of these installation tests may be run by using the bash script "vlidort_run_subset" instead of the script "vlidort_run". To do this, go inside "vlidort_run_subset" and choose the desired test(s) to run by setting the desired test variable to "1" and insuring the others are set to "0". The "vlidort_run_subset" script is then run in a manner identical to "vlidort_run".

Upon completing execution, one may compare the contents of the result files (located in the "s" or "v" test directory) with benchmark results generated at RT solutions using the Intel® or GNU Fortran compilers. The latter results are located in the respective "saved_results" subdirectory. This comparison is performed with the "vlidort_check2" script. However, to use this script, the "vlidort_diff" utility must first be compiled. First, check that a copy of the Makefile from either the "vlidort_s_test" or "vlidort_v_test" subdirectory is present in the parent directory, then compile this utility via the command:

$ make vlidort_diff FC=<your_compiler>

Then, the comparison is done by executing the script "vlidort_check2" (for example, the results from the scalar tests located in the "vlidort_s_test" subdirectory),

$ vlidort_check2 s <check_compiler>

Here, <check_compiler> is either "ifort" or "gfortran".

Table 4.5. Files for VLIDORT Scalar Tests

Program #	Environment file	Input configuration files	Output result files
1	2p7_solar_tester.f90	2p7_VLIDORT_ReadInput.cfg	results_solar_tester.all

2	2p7_ solar_lpcs_tester.f90	2p7_VLIDORT_ReadInput.cfg	results_solar_lcs_tester.all
			results_solar_lcs_
			    tester.all_FO
			results_solar_lps_tester.all
			results_solar_lps_
			    tester.all_FO

3	2p7_thermal_tester.f90	2p7_VLIDORT_ReadInput.cfg	results_thermal_tester.all results_brdf_thermcheck.res

4	2p7_ thermal_lpcs_tester.f90	2p7_VLIDORT_ReadInput.cfg	results_thermal_lcs_tester.all
			results_thermal_lcs_
			    tester.all _FO
			results_thermal_lps_tester.all
			results_thermal_lps_
			    tester.all _FO

5	2p7_brdfplus_tester.f90	2p7_VLIDORT_ReadInput.cfg VBRDF_ReadInput.cfg	results_brdfplus_tester.all results_brdf_supcheck.res results_brdf_supcheck.wfs

Table 4.5A. Files for VLIDORT Vector Tests

Program #	Environment file	Input configuration files	Output result files
1	V2p7_solar_tester.f90	V2p7_VLIDORT_ReadInput.cfg	results_solar_tester_I000.all

2	V2p7_ solar_lpcs_tester.f90	V2p7_VLIDORT_ReadInput.cfg	results_solar_lcs_tester_
			    normal_ I000.all
			results_solar_lcs_tester_
			    normal_ I000.all_FO
			results_solar_lps_tester_
			    normal_I000.all
			results_solar_lps_tester_
			    normal_I000.all_FO

3	V2p7_thermal_tester.f90	V2p7_VLIDORT_ReadInput.cfg	results_thermal_tester_ I000.all

4	V2p7_ thermal_lpcs_tester.f90	V2p7_VLIDORT_ReadInput.cfg	results_thermal_lcs_ tester_
			    I000.all
			results_thermal_lcs_ tester_
			    I000.all_FO
			results_thermal_lps_tester_
			    I000.all
			results_thermal_lps_tester_
			    I000.all_FO

5	V2p7_brdfplus_tester.f90	V2p7_VLIDORT_ReadInput.cfg
		VBRDF_ReadInput.cfg
			results_brdfplus_tester_
			    normal_I000 .all
			results_brdf_supcheck_
			    normal.res
			results_brdf_supcheck_
			    normal.wfs

6	V2p7_vsleavefplus_tester.f90	V2p7_VLIDORT_ReadInput.cfg	results_vsleaveplus_tester_

		VSLEAVE_ReadInput.cfg	normal_fluor.all
7	V2p7_Siewert2000_validation.f90	V2p7_Siewert2000_validation.cfg	results_Siewert2000_ validation.all
8	V2p7_solar_OMP_tester.f90	V2p7_VLIDORT_ReadInput.cfg	results_solar_tester_I000.all_ nt1_nwn00010 results_solar_tester_I000.all_ nt2_nwn00010
9	V2p7_solar_lpcs_OMP_tester.f90	V2p7_VLIDORT_ReadInput.cfg	results_solar_lcs_tester_ normal_I000.all_nt1_nwn00010 results_solar_lcs_tester_ normal_I000.all_nt2_nwn00010 results_solar_lps_tester_ normal_I000.all_nt1_nwn00010 results_solar_lps_tester_ normal_I000.all_nt2_nwn00010

For the scalar test example here, any differences will be placed in difference files starting with "diff_" and will be located in the subdirectory "vlidort_s_test". Often there will be trivial differences between results run on different machines with different compilers, so these difference files may not be of the same size, but should only contain sets of lines differing in trivial ways. Currently the difference files will be of ~224 bytes if there are no differences between freshly generated results and those archived in the "saved_results" subdirectories. This is due to the fact that the "vlidort_diff" utility (used inside "vlidort_check2") returns some basic information about each file analyzed and the thresholds used to distinguish trivial from nontrivial differences between freshly generated results output and older archived data. We will not discuss these thresholds further here. Currently, difference files may be generated for vector tests run with NSTOKES set to 1 or 3.

The results from the vector tests may be checked in a similar way by executing the script "vlidort_check" as

$ vlidort_check v <check_compiler>

The main difference between "vlidort_check" and "vlidort_check2" is that the first runs the basic Unix "diff" utility and the second the more tailored "vlidort_diff" utility.

Note: If the results in the result file results_vsleaveplus_tester_normal_fluor.all do not match those in the archived results file after initially running the scripts "vlidort_run" and "vlidort_check2", go into the file "vsleave_sup_routines.f90" inside the subdirectory "vsup/vsleave" and change the logical variable "use_nag_compiler" in the subroutine "get_fluorescence_755" from ".false." to ".true.". The issue is usually related to the reading of a binary fluorescence data file used in this test and can usually be corrected by switching the sense of this logical variable which allows the binary file to be read slightly differently.

Additional tests for checking Observational Geometry Mode

In addition to the tests above, one may run several of the above vector tests to run VLIDORT in observational geometry mode and check the results of those tests. To do this, do the following:

(1) Modify the reply to the question "Do Observation Geometry?" from "f" to "t" in the following four input configuration files:

   V2p7_VLIDORT_ReadInput.cfg

V2p7_vsleaveplus_tester.cfg
VBRDF_ReadInput.cfg
VSLEAVE_ReadInput.cfg

(2) From the command prompt, do

$ vlidort_run_extras.bash v `<your_compiler>` obsgeo

To check the results against saved result files, do

$ vlidort_check2.bash v `<check_compiler>`/obsgeo

(3) Return the reply to the question "Do Observation Geometry?" from "t" back to "f" in the four input configuration files to their default state.

Additional tests for checking the case NSTOKES=3

Again, in addition to the standard tests above, one may run several of the above vector tests to run VLIDORT in vector mode with NSTOKES = 3 and check the results of those tests. To do this, do the following:

(1) Modify the reply to the entry "Number of Stokes vector components" from "1" to "3" in the following two input configuration files:

V2p7_VLIDORT_ReadInput.cfg
VBRDF_ReadInput.cfg

(2) From the command prompt, do

$ vlidort_run_extras.bash v `<your_compiler>` nstokes3

To check the results against saved result files, do

$ vlidort_check2.bash v `<check_compiler>`/nstokes3

(3) Return the reply to the entry "Number of Stokes vector components" from "3" back to "1" in the two input configuration files to their default state.

Additional tests for checking VLIDORT calculations with OpenMP

It is possible to run some of the above tests to observe the VLIDORT performance enhancement in a parallel computing environment using OpenMP. To run these OpenMP test programs, return to the home directory in which you have installed the VLIDORT package and run the bash script "vlidort_run_OMP" from the command line as:

$ vlidort_run_OMP.bash gfortran

Note that although a number of FORTRAN 90/95 compilers support OpenMP Version 3.1 (or later) parallel computing environment, the VLIDORT Version 2.7 package has been tested only with the "gfortran" and "ifort" compilers using OpenMP. The user is advised to consult requirements for using OpenMP with compilers other than "gfortran" and "ifort". The compiler flag for "ifort' is `–openmp`, while for "gfortran" we are using `–fopenmp –frecursive` flags. Earlier versions of the "gfortran" compiler do not have the `–frecursive` flag.

In addition, before running the "vlidort_run_OMP" script, the user is referred to section 6.2.3 for more information regarding proper preparation for OpenMP tests: parallel programming tests such as these can result in memory segmentation faults if steps are not taken to ensure enough

memory is set aside for both main and OpenMP-spawned computational threads. Results of the OpenMP tests may be checked using

$ vlidort_check2.bash v <check_compiler>

We turn now to some of the contents of the scalar test environment programs. The programs will produce VLIDORT output for one particular atmospheric scenario, a 23-layer atmosphere with molecular absorption and scattering in all layers, and with aerosols in the lowest 6 layers. The prepared atmosphere is partly contained in the file `input_atmos.dat`, and the aerosols are inserted by hand. Down-welling and up-welling output is generated for 36 geometries (3 solar zenith angles, 4 relative azimuth angles, 3 viewing zenith angles) and for 5 vertical levels. In all cases, azimuth-averaged outputs (actinic and regular fluxes + linearizations) are generated as well as radiances and Jacobians of intensities.

The testers (or "drivers") are used to perform several tasks. For example, for the driver `2p7_solar_lpcs_tester.f90`, the first task is a baseline calculation of radiances, two total column Jacobians (with respect to the total gas absorption optical depth G and the total aerosol optical depth Y) and one surface Jacobian with respect to Lambertian albedo A. The remaining tasks are designed to validate these analytic Jacobians by finite difference (FD) estimates. For the FD tasks, all linearization options are turned off, and for threads 2-4 respectively, intensity-only calculations are done with G, Y and A perturbed by 0.1% of their original values. The final output file contains the baseline intensity followed by 6 columns giving the normalized Jacobians, featuring the 3 analytic computations from thread 1 and the 3 finite difference estimates from threads 2-4.

Programs 1-4 are controlled by the configuration file `2p7_VLIDORT_ReadInput.cfg`, which is first read by the VLIDORT input read routine, then checked for errors before the main call to VLIDORT is undertaken. Program 1 generates radiances and fluxes only. Program 2 generates radiances and fluxes, but also their linearizations with respect to 2 total column weighting functions (the total amount of trace gas in the atmosphere, and the total aerosol loading in the lowest 6 layers), 2 profile weighting functions (trace gas absorber and aerosol extinction profile), and surface property weighting functions with respect to the Lambertian albedo. Programs 3 and 4 perform similar computations, but where thermal sources are also present.

Program 5 provides an example using VLIDORT with the VBRDF supplements. Here the scenario is a 3-kernel BRDF surface (Ross-thin, Li-dense, Cox-Munk). In addition to the configuration file `2p7_VLIDORT_ReadInput.cfg`, program 5 is also controlled by the configuration file `VBRDF_ReadInput.cfg`, which is first read by the BRDF input read routine. Certain input variables from the two configuration files are then checked for consistency before the BRDF Fourier components are calculated and passed to VLIDORT by the subroutine VLIDORT_BRDF_INPUT_CHECKER in the module `vlidort_sup_accessories`. Program 5 generates 6 surface property weighting functions for this 3-kernel BRDF - one for each of the three kernel amplitude factors, two more with respect to Li-dense kernel parameters, and a final one for the Cox-Munk wind speed.

In the vector tests, program 6 provides an example of using the standard and linearized VSLEAVE supplement code (both VSLEAVE input read and VSLEAVE computational subroutines) in conjunction with associated calls to VLIDORT (to VLIDORT_MASTER and VLIDORT_LCS_MASTER). A special subroutine VLIDORT_VSLEAVE_INPUT_CHECKER

(in module `vlidort_sup_accessories`) is called to check the consistency of related input fed to both VLIDORT and the given VSLEAVE computational subroutine. This surface-leaving test simulates the effect of fluorescence in the spectral band 640-820nm.

Program 7 performs a VLIDORT validation check against results found in Siewert (2000c). See section 6.2.2 for more on the details of this test.

Finally, programs 8 and 9 provide examples of VLIDORT usage in an OpenMP parallel programming environment. Programs 8 and 9 have the same solar-source set-ups as those for programs 1 and 2, but now each test driver comes with a series of OpenMP parallel programming directives. CPU timing information is also generated to give the user an idea of the computational acceleration that may be obtained. Although the OpenMP tests are currently set up for one or two computational threads, four or more threads can easily be implemented by a few simple driver changes. The drivers were set up this way, since computational speed-up is limited by the number of available processing cores, and two is the minimum number of cores required to demonstrate the computational speed-up using OpenMP. For a machine with multiple cores, the performance "scalability" is excellent. For example, with four cores a speed-up of almost 4-fold is achieved: 24.78 second (1 core), 12.40 seconds (2 cores) and 6.62 seconds (4 cores).

Section 6.2 has additional notes on the scalar and vector test cases in this installation. Appendices 6.3 and 6.4 have descriptions of the VBRDF and VSLEAVE supplements.

4.3.5 Helpful Tips for input settings

In this section, we compile some useful tips for setting the inputs.

All geometrical angles are given in degrees. Solar angles must lie in the range [0°,90°); this version of VLIDORT is not a twilight code. Viewing zenith angles are by convention positive in the range [0°,90°], and relative azimuth angles are in the range [0°,360°]. These inputs are checked; invalid values will cause the model to abort and generate error messages.

Output at various vertical levels is essentially specified according to geometrical height (not optical depth as in DISORT and earlier versions of VLIDORT). The reason for this is that the geometrical height specification is independent of wavelength. We illustrate the convention for vertical output with some examples. USER_LEVELS(1) = 2.0 means that the first level for output will be at the bottom of the second layer in the atmosphere. USER_LEVELS(2) = 2.5 means that the second level of output will be halfway down the third layer. Thus if you want TOA output only, then you need to set USER_LEVELS(1) = 0.0. If there are 24 layers in your atmosphere and you want BOA output only, then you set USER_LEVELS(1) = 24.0. The ordering is not important; VLIDORT will make an internal "sort" of the output levels into ascending order, and the final intensities and Jacobians will be generated in the sorted order. Out-of-range levels are rejected (this is a fatal input check error).

The number of scattering matrix expansion coefficients (NGREEK_MOMENTS_INPUT) should be at least $2N-1$, where N is the number of discrete ordinates in the polar angle half space (the variable NSTREAMS). If you are using the delta-M scaling, then NGREEK_MOMENTS_INPUT should be at least $2N$ (otherwise the scaling will not work). By definition, the multiple scattering fields are calculated using at most $2N-1$ (possibly scaled)

expansion coefficients, whereas the exact single scatter calculations will use all coefficients from 0 to NGREEK_MOMENTS_INPUT.

4.4 Exception handling and utilities

4.4.1 Exception handling

There are two types of exception handling in VLIDORT, one for checking the model input, the other for dealing with calculation failures. Main subroutines VLIDORT_MASTER, VLIDORT_LCS_MASTER and VLIDORT_LPS_MASTER have the exception handling outputs listed in Table 4.6.

Table 4.6. Exception handling for the VLIDORT 2.7 code
(✞; 0=VLIDORT_SUCCESS, 3=VLIDORT_WARNING, 4=VLIDORT_SERIOUS)

Name	Type	Values	Purpose
STATUS_INPUTCHECK	INTEGER	0, 3 or 4 ✞	Overall Status of Input-check
NCHECKMESSAGES	INTEGER	0 to 25	Number of Input-check Error Messages
CHECKMESSAGES	CHARACTER	ASCII String	Array of Input-check Error Messages
ACTIONS	CHARACTER	ASCII String	Array of Input-check Actions to take
STATUS_CALCULATION	INTEGER	0 or 4 ✞	Overall Status of Calculation
MESSAGE	CHARACTER	ASCII String	Calculation Failure, Message
TRACE_1	CHARACTER	ASCII String	First Subroutine Trace for Place of Failure
TRACE_2	CHARACTER	ASCII String	Second Subroutine Trace for Place of Failure
TRACE_3	CHARACTER	ASCII String	Third Subroutine Trace for Place of Failure

The integers STATUS_INPUTCHECK and STATUS_CALCULATION can take one of several values indicated in the VLIDORT_pars module (see section 4.2.1.1 above). Input checking is done first, before any calculation takes place. If STATUS_CHECKINPUT equals the parameter VLIDORT_SUCCESS (value 0), then the input check is successful. If there is an error with this procedure, then a message string is generated and stored in the array CHECKMESSAGES and the number of such messages (NCHECKMESSAGES) is increased by 1. At the same time, a second character string is generated and stored in the array ACTIONS - these strings give the user hints as to how to fix the inconsistent or incorrect input specified. If there is a fatal error in the input checking (STATUS_INPUTCHECK = VLIDORT_SERIOUS), VLIDORT will exit immediately. Not all checking errors are fatal. If there is a warning error (STATUS_INPUTCHECK = VLIDORT_WARNING), VLIDORT will continue execution, but warning messages and actions concerning the input will be generated and stored in CHECKMESSAGES and ACTIONS. If warnings occur, VLIDORT will correct the input internally and proceed with the execution.

STATUS_CALCULATION refers to the status of the radiative transfer calculation. If an error has been returned from one of the internal calculation routines, then the overall flag STATUS_CALCULATION will be set to VLIDORT_SERIOUS. All calculation errors are fatal. Apart from the use of standard numerical routines to solve the eigensystem and a number of linear algebra problems, VLIDORT is entirely analytical. Only in exceptional circumstances should an error condition be returned from the one of the eigenroutines (ASYMTX or DGEEV from LAPACK) or one of the LAPACK linear algebra modules. One possibility to watch out for is degeneracy caused by two layers having identical optical properties. Experience has shown that

such errors are invariably produced by bad optical property input that has escaped the input check.

A message about the calculation error is generated along with 3 traces for that error (as noted above in the table). Provided inputs are correctly generated, there should be little opportunity for the software to generate such an error. If you have persistent calculation errors, please send a message to the author at rtsolutions@verizon.net.

The VLIDORT package also contains an optional subroutine (VLIDORT_WRITE_STATUS) that should be called immediately after either of the two main master routines. If there are any errors or warnings, this routine will generate an "Output Log" file (with prescribed name and unit number) containing relevant error messages and traces as listed in Table 4.6. The opening of the "log" file is controlled by a flag which will be set when the first error is obtained. If there are no errors, you will get the message "VLIDORT has executed successfully". The author recommends usage of this routine, or at the very least, the two main output status integers should be examined upon exiting any of the master calling routines.

Table 4.7. Exception handling for the File-reads
(✟; 0=VLIDORT_SUCCESS, 3=VLIDORT_WARNING, 4=VLIDORT_SERIOUS)

Name	Type	Values	Purpose
STATUS_INPUTREAD	INTEGER	0 or 4 ✟	Overall Status of Input-read
NINPUTMESSAGES	INTEGER	0 to 25	Number of Input-read Error Messages
INPUTMESSAGES	CHARACTER	ASCII String	Array of Input-read Error Messages
INPUTACTIONS	CHARACTER	ASCII String	Array of Input-read Actions to take

If you are using the input routine VLIDORT_INPUT_MASTER to open a configuration file and read in inputs (see example in section 4.3.1), then exception handling for this procedure has a similar form (Table 4.7). If there are any errors from a call to VLIDORT_INPUT_MASTER, then you should examine the output by printing out the above messages in Table 4.7 whenever STATUS_INPUTREAD is equal to 4 (VLIDORT_SERIOUS). The BRDF supplemental programs also have input-read routines with the same exception handling procedures as noted in Table 4.7.

4.4.2 Utilities

All software in VLIDORT was written by R. Spurr, with the exception of a number of utility routines taken from standard sources. Most VLIDORT utility routines are collected together in the module file "vlidort_aux.f90". They include a number of standard numerical routines, and some file-read and error handling routines.

Numerical routines are: ASYMTX (eigensolver module from DISORT); GAULEG (Gauss-Legendre quadrature determination, adapted from Numerical Recipes); CFPLGARR (Legendre-polynomial generator). The FINDPAR tool for reading the initialization file was developed by J. Lavagnino and is found here. Note that for scalar calculations, ASYMTX is preferred over the LAPACK eigensolver DGEEV for performance reasons (the latter looks for complex solutions and is approximately twice as slow). However, DGEEV is required for the complex calculations in VLIDORT.

A selection of routines from the LAPACK library is used in VLIDORT and is contained in the file "lapack_tools.f90". The most important routines in the LAPACK selection are DGEEV, DGBTRF, DGBTRS, DGETRF, DGETRS, DGBTF2, DLASWP, XERBLA, DGETF2, DGEMM, DGEMV, DGER, DTBSV, and DTRSM. These LAPACK routines are not performance-optimized for the VLIDORT package (there is in particular a lot of redundancy in the linear algebra problems). The LAPACK routines were given literal translations into Fortran 90 equivalents. Eventually, it is expected that the LAPACK routines will be upgraded with enhanced performance in terms of run-time and efficiency.

4.5 Copyright issues: GNU License

R. Spurr developed the original VLIDORT model at the Smithsonian Astrophysical Observatory (SAO) in 2004. All software generated at SAO is in the public domain. All subsequent versions of the code have remained in the public domain; sponsorship has come from a number of US and European Government Institutions. The following copyright remarks apply to this Version 2.7 release of VLIDORT software distributed by RT Solutions, Inc.

PERMISSION TO USE, COPY, MODIFY, AND DISTRIBUTE ANY VLIDORT SOFTWARE DEVELOPED BY RT SOLUTIONS INC., ANY DOCUMENTATION APPERTAINING TO THE VLIDORT SOFTWARE AND ANY RESULTS OBTAINED USING VLIDORT SOFTWARE IS HEREBY GRANTED WITHOUT FEE AND WITHOUT WRITTEN AGREEMENT, PROVIDED THAT BOTH THE NOTICE OF COPYRIGHT AS EXPRESSED IN THIS PARAGRAPH AND THE FOLLOWING TWO DISCLAIMER PARAGRAPHS APPEAR IN ALL COPIES OF THE SOFTWARE.

IN NO EVENT SHALL RT SOLUTIONS BE LIABLE TO ANY PARTY FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OF THE VLIDORT SOFTWARE IN VERSION 2.7 AND ITS DOCUMENTATION, EVEN IF RT SOLUTIONS INC. HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE SOFTWARE IS WITH THE USER.

BECAUSE THE VLIDORT PROGRAM VERSION 2.7 IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM TO THE EXTENT PERMITTED BY APPLICABLE LAW.EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. RT SOLUTIONS HAS NO OBLIGATION TO PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS OR MODIFICATIONS TO THE VLIDORT SOFTWARE IN VERSION 2.7.

4.6 Acknowledgments

In the present document, Chapters 4 and 6 have been extensively revised in the light of the Fortran 90 translation and associated use of newly defined input and output Type structures. The author is indebted to Mick Christi for assistance with this revision.

5. References

Anderson, E., Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen, LAPACK User's Guide, 2nd Edition, Philadephia, Society for Industrial and Applied Mathematics, 1995.

Barichello, L., R. Garcia, and C. Siewert, Particular solutions for the discrete-ordinates method, *J. Quant. Spectrosc. Radiat. Transfer*, **64**, 219-226, 2000.

Caudill T.R., D.E. Flittner, B.M. Herman, O. Torres, and R.D. McPeters, Evaluation of the pseudo-spherical approximation for backscattered ultraviolet radiances and ozone retrieval, *J. Geophys. Res.*, **102**, 3881-3890, 1997.

Chami M., R. Santer, and E. Dilligeard, Radiative transfer model for the computation of radiance and polarization in an ocean-atmosphere system: polarization properties of suspended matter for remote sensing, *Applied Optics*, **40**, 2398-2416, 2001.

Chandrasekhar, S., Radiative Transfer, Dover Publications Inc., New York, 1960.

Christi M.J., and G.L. Stephens, Retrieving profiles of atmospheric CO_2 in clear sky and in the presence of thin cloud using spectroscopy from the near and thermal infrared: a preliminary case study, *J. Geophys. Res.*, **109**, D04316, doi: 10.1029/2003JD004058, 2004.

Coulson K., J. Dave, and D. Sekera., Tables related to radiation emerging from planetary atmosphere with Rayleigh scattering, University of California Press, Berkeley, 1960.

Crisp D., R.M. Atlas, F-M. Bréon, L.R. Brown, J.P. Burrows, P. Ciais, B.J. Connor, S.C. Doney, I.Y. Fung, D.J. Jacob, C.E. Miller, D. O'Brien, S. Pawson, J.T. Randerson, P. Rayner, R.J. Salawitch, S.P. Sander, B. Sen, G.L. Stephens, P.P. Tans, G.C. Toon, P.O. Wennberg, S.C. Wofsy, Y.L. Yung , Z. Kuang, B. Chudasama, G. Sprague, B. Weiss, R. Pollock, D. Kenyon, and S. Schroll, The Orbiting Carbon Observatory (OCO) Mission, *Adv. Space Res.,* **34**, 700, 2004.

Cox, C., and W. Munk. Statistics of the sea surface derived from sun glitter, *J. Mar. Res*., **13,** 198-227, 1954.

Cox, C., and W. Munk, Measurement of the roughness of the sea surface from photographs of the sun's glitter, *J. Opt. Soc. Am*., **44**, 838-850, 1954.

Dahlback A., and K. Stamnes, A new spherical model for computing the radiation field available for photolysis and heating at twilight, *Planet. Space Sci.*, **39**, 671, 1991.

Dave, J.V., Intensity and polarization of the radiation emerging from a plane-parallel atmosphere containing monodispersed aerosols, *Applied Optics*, **9**, 2673-2684, 1970.

de Haan, J.F., P.B. Bosma, and J.W. Hovenier. The adding method for multiple scattering of polarized light, *Astron. Astrophys.*, **183**, 371-391, 1987.

de Rooij, W.A., and C.C.A.H. van der Stap Expansion of Mie scattering matrices in generalized spherical functions, *Astron. Astrophys.,* **131**, 237-248, 1984.

Deuze, J.L., P. Goloub, M. Herman, A. Marchand, G. Perry, S. Susana, and D. Tanre, Estimate of the aerosol properties over the ocean with POLDER, *J. Geophys. Res.*, **105,** 15329, 2000.

EPS/METOP System – Single Space Segment – GOME-2 requirements Specification. ESA/EUMETSAT, MO-RS-ESA-GO-0071, 1999: Issue 2.

Frankenberg, C., C. O'Dell, L. Guanter, and J. McDuffie, Remote sensing of near-infrared chlorophyll fluorescence from space in scattering atmospheres: implications for its retrieval and interferences with atmospheric CO_2 retrievals, *Atmos. Meas. Tech*., **5,** 2081-2094, 2012.

Garcia, R.D.M., and C.E. Siewert, A Generalized Spherical Harmonics Solution for Radiative Transfer Models that Include Polarization Effects, *JQSRT*, **36**, 401-423, 1986.

Garcia, R.D.M, and C.E. Siewert, The F_N method for radiative transfer models that include polarization, *JQSRT,* **41**, 117-145, 1989.

Hansen, J.E., and L.D. Travis, Light scattering in planetary atmospheres, *Space Sci. Rev.*, **16**, 527-610, 1974.

Hapke, B., Theory of Reflectance and Emittance Spectroscopy (Cambridge University Press, Cambridge, UK., 1993).

Hasekamp, O.P., and J. Landgraf, A linearized vector radiative transfer model for atmospheric trace gas retrieval. *JQSRT*, **75**, 221-238, 2002.

Hasekamp, O., J. Landgraf, and R. van Oss, The need of polarization monitoring for ozone profile retrieval from backscattered sunlight. *J. Geophys. Res.,* **107**, 4692, 2002.

Heintzenberg, J., H-F. Graf, R.J. Charlson, and P. Warneck, Climate forcing and physico-chemical life cycle of the atmospheric aerosol - why do we need an integrated, interdisciplinary global research programme?, *Contr. Atmos. Phys.*, **69**, 261-271, 1996.

Hovenier, J.W., Multiple scattering of polarized light in planetary atmospheres, *Astron. Astrophys.*, **13**, 7- 29, 1971.

Hovenier, J.W., and C.V.M. van der Mee, Fundamental relationships relevant to the transfer of polarized light in a scattering atmosphere, *Astron. Astrophys.,* **128**, 1-16, 1983.

Hovenier, J.W., C. van der Mee, and H. Domke, Transfer of Polarized Light in Planetary Atmospheres Basic Concepts and Practical Methods, Kluwer, Dordrecht, 2004.

Jiang, Y., X. Jiang, R-L. Shia, S.P. Sander, and Y.L. Yung, Polarization study of the O_2 A-band and its application to the retrieval of O_2 column abundance, EOS *Trans. Am Geophys Union,* **84**, 255, 2003.

Jin, Z., T. Charlock, K. Rutledge, K. Stamnes, and Y. Wang, Analytic solution of radiative transfer in the coupled atmosphere-ocean system with a rough surface. *Applied Optics*, **45**, 7433-7455, 2006.

Kotchenova S. Y., E. F. Vermote, R. Matarrese, and F. J. Klemm. Validation of a vector version of the 6S radiative transfer code for atmospheric correction of satellite data. Part I: Path radiance. *Applied Optics*, **45**, 6762-74, 2006.

Lacis, A., J. Chowdhary, M. Mishchenko, and B. Cairns, Modeling errors in diffuse sky radiance: vector vs. scalar treatment, *Geophys. Res. Lett.*, **25**, 135-8, 1998.

Landgraf, J., O. Hasekamp, T. Trautmann, and M. Box, A linearized radiative transfer model for ozone profile retrieval using the analytical forward-adjoint perturbation theory approach, *J. Geophys. Res.,* **106**, 27291-27306, 2001.

Lucht W., C. Schaaf, and A. Strahler. An Algorithm for the Retrieval of Albedo from Space Using Semi-empirical BRDF Models, IEEE *Trans. Geosci Remote Sens.*, **38**, 977, 2000.

Lucht W., and J.-L. Roujean. Considerations in the Parametric Modeling of BRDF and Albedo from Multiangular Satellite Sensor Observations, *Remote Sensing Reviews,* Vol. 18, pp. 343-379, 2000.

Mackowski, D.W., and M.I. Mishchenko, Calculation of the T matrix and the scattering matrix for ensembles of spheres, *J. Opt. Soc. Am. A*, **13**, 2266-2278, 1996.

Maignan, F., F.-M. Bréon, E. Fédèle, and M. Bouvier, Polarized reflectance of natural surfaces: Spaceborne measurements and analytical modeling, *Rem. Sens Env.*, **113**, 2642-2650 (2009).

Mishchenko, M., A. Lacis, and L. Travis, Errors induced by the neglect of polarization in radiance calculations for Rayleigh scattering atmospheres, *JQSRT*, **51**, 491-510, 1994.

Mishchenko, M.I., and L.D. Travis, Satellite retrieval of aerosol properties over the ocean using polarization as well as intensity of reflected sunlight, *J. Geophys. Res.*, **102**, 16989, 1997.

Mishchenko, M.I., and L.D. Travis, Capabilities and limitations of a current FORTRAN implementation of the T-matrix method for randomly oriented, rotationally symmetric scatterers, *JQSRT*, **60**, 309- 324, 1998.

Mishchenko, M., J. Hovenier, and L. Travis, Eds., Light Scattering by non-Spherical Particles, Academic Press, San Diego, 2000.

Mishchenko, M.I., Microphysical approach to polarized radiative transfer: extension to the case of an external observation point, *Applied Optics*, **42**, 4963- 4967, 2003.

Mishchenko, M.I., B. Cairns, J.E. Hansen, L.D. Travis, R. Burg, Y.J. Kaufman, J.V. Martins, and E.P. Shettle, Monitoring of aerosol forcing of climate from space: Analysis of measurement requirements, *JQSRT*, **88**, 149-161, 2004.

Morel, A., and B. Gentili. A simple band ratio technique to quantify the colored dissolved and detrital organic material from ocean color remotely sensed data. *Remote Sens. Env.*, **113**, 2009.

Nadal, F., and F.-M. Bréon, Parameterization of surface polarized reflectance derived from POLDER spaceborne measurements, IEEE Transactions on Geoscience and Remote Sensing, **37**, 1709-1718 (1999).

Nakajima, T., and M. Tanaka, Algorithms for radiative intensity calculations in moderately thick atmospheres using a truncation approximation, *J. Quant. Spectrosc. Radiat. Transfer*, **40**, 51-69, 1988.

Natraj, V., R. Spurr, H. Boesch, Y. Jiang, and Y.L. Yung, Evaluation of Errors from Neglecting Polarization in the Forward Modeling of O_2 A Band measurements from Space, with Relevance to the CO_2 Column Retrieval from Polarization-Sensitive Instruments. *J. Quant. Spectrosc. Radiat. Transfer*, **103**, 245-259, 2007.

Rahman, H., B. Pinty, and M. Verstrate, Coupled surface-atmospheric reflectance (CSAR) model. 2. Semi-empirical surface model usable with NOAA advanced very high resolution radiometer data, *J. Geophys. Res.,* **98**, 20791, 1993.

Rodgers, C.D., Inverse Methods for Atmospheric Sounding: Theory and Practice, World Scientific Publishing Co. Pte. Ltd., Singapore, 2000.

Rozanov, V, T. Kurosu, and J. Burrows, Retrieval of atmospheric constituents in the UV-visible: a new quasi-analytical approach for the calculation of weighting functions, *JQSRT*, **60**, 277-299, 1998.

Rozanov A.V., V.V. Rozanov, and J.P. Burrows, Combined differential-integral approach for the radiation field computation in a spherical shell atmosphere: Non-limb geometry, *J. Geophys. Res.*, **105**, 22937-22942, 2000.

Sancer, M., Shadow-corrected electromagnetic scattering from a randomly-rough ocean surface, *IEEE Trans Antennas Propag*, **AP-17**, 557-585, 1969.

Schulz, F.M., K. Stamnes, and F. Weng, VDISORT: an improved and generalized discrete ordinate method for polarized (vector) radiative transfer, *JQSRT*, **61**, 105-122, 1999.

Schulz, F.M., and K. Stamnes, Angular distribution of the Stokes vector in a plane-parallel vertically inhomogeneous medium in the vector discrete ordinate radiative transfer (VDISORT) model, *JQSRT*, **65**, 609-620, 2000.

Schutgens, N., and P. Stammes, A novel approach to the polarization correction of spaceborne spectrometers, *J. Geophys. Res.*, **108**, 4229, 2003. doi:10.1029/2002JD002736.

Siewert, C.E., On the equation of transfer relevant to the scattering of polarized light, *Astrophysics J.,* **245**, 1080-1086, 1981.

Siewert, C.E., On the phase matrix basic to the scattering of polarized light, *Astron. Astrophys.*, **109**, 195-200, 1982.

Siewert, C.E., A concise and accurate solution to Chandrasekhar's basic problem in radiative transfer, *J. Quant. Spectrosc. Radiat. Transfer*, **64**, 109-130, 2000.

Siewert, C.E., A discrete-ordinates solution for radiative transfer models that include polarization effects, *JQSRT*, **64**, 227-254, 2000.

Spurr, R., T. Kurosu, and K. Chance, A linearized discrete ordinate radiative transfer model for atmospheric remote sensing retrieval, *J. Quant. Spectrosc. Radiat. Transfer*, **68**, 689–735, 2001.

Spurr, R., Simultaneous derivation of intensities and weighting functions in a general pseudo-spherical discrete ordinate radiative transfer treatment, *J. Quant. Spectrosc. Radiat. Transfer*, **75**, 129–175, 2002.

Spurr, R.J.D., LIDORT V2PLUS: A comprehensive radiative transfer package for UV/VIS/NIR nadir remote sensing; a General Quasi-Analytic Solution. *Proc. S.P.I.E. International Symposium, Remote Sensing 2003*, Barcelona, Spain, September 2003.

Spurr, R.J.D., A New Approach to the Retrieval of Surface Properties from Earthshine Measurements, *J. Quant. Spectrosc. Radiat. Transfer*, **83**, 15-46, 2004.

Spurr, R. J. D., 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. Transfer*, **102(2)**, 316-342, doi:10.1016/j.jqsrt.2006.05.005 (2006).

Spurr, R., and M. J. Christi, Linearization of the Interaction Principle: Analytic Jacobians in the Radiant Model, *JQSRT*, **103/3**, 431-446, doi 10.1016/j.jqsrt.2006.05.001, 2006.

Spurr, R., LIDORT and VLIDORT: Linearized pseudo-spherical scalar and vector discrete ordinate radiative transfer models for use in remote sensing retrieval problems. *Light Scattering Reviews*, Volume 3, ed. A. Kokhanovsky, Springer, 2008.

Spurr, R., and V. Natraj, A linearized two-stream radiative transfer code for fast approximation of multiple-scatter fields, *J. Quant. Spectrosc. Radiat. Transfer,* **112**, 2630-2637, 2011.

Spurr, R., J. Wang, J. Zeng, and M. Mishchenko, Linearized T-Matrix and Mie scattering computations, *J. Quant. Spectrosc. Radiat. Transfer,* **113**, 425-439, 2012.

Spurr R, and M. Christi., On the generation of atmospheric property Jacobians from the (V)LIDORT linearized radiative transfer models, *J. Quant. Spectrosc. Radiat. Transfer*, DOI: 10.1016/j.jqsrt.2014.03.011, 2014.

Sromovsky, L.A., Effects of Rayleigh-scattering polarization on reflected intensity: a fast and accurate approximation method for atmospheres with aerosols. *Icarus*, **173**, 284, 2005.

Stam, D.M., J.F. de Haan, J.W. Hovenier, and P. Stammes, Degree of linear polarization of light emerging from the cloudless atmosphere in the oxygen A band, *J. Geophys. Res.*, **104**, 16843, 1999.

Stamnes, K., and P. Conklin, A new multi-layer discrete ordinate approach to radiative transfer in vertically inhomogeneous atmospheres, *J. Quant. Spectrosc. Radiat. Transfer*, **31**, 273, 1984.

Stamnes, K., S.-C. Tsay, W. Wiscombe, and K. Jayaweera, Numerically stable algorithm for discrete ordinate method radiative transfer in multiple scattering and emitting layered media, *Applied Optics*, **27**, 2502-2509, 1988.

Stamnes K., S-C. Tsay, W. Wiscombe, and I. Laszlo, DISORT: A general purpose Fortran program for discrete-ordinate-method radiative transfer in scattering and emitting media. Documentation of Methodology Report, available from 82H82H83Hftp://climate.gsfc.nasa.gov/wiscombe/Multiple_scatt/, 2000.

Stammes, P., J.F. de Haan, and J.W. Hovenier, The polarized internal radiation field of a planetary atmosphere, *Astron. Astrophys.*, **225**, 239-259, 1989.

Stammes, P., P. Levelt, J. de Vries, H. Visser, B. Kruizinga, C. Smorenburg, G. Leppelmeier, and E. Hilsenrath, Scientific requirements and optical design of the Ozone Monitoring Instrument on EOS-CHEM. *Proceedings of the SPIE Conference on Earth Observing Systems IV*, July 1999, Denver, Colorado, USA, vol. SPIE 3750, 221-232, 1999.

Thomas, G. E., and K. Stamnes, Radiative Transfer in the Atmosphere and Ocean, Cambridge University Press, 1999.

Ustinov, E.A., Analytic evaluation of the weighting functions for remote sensing of blackbody planetary atmospheres: A general linearization approach, *JQSRT*, **74**, 683-686, 2002.

Ustinov, E.A., Atmospheric weighting functions and surface partial derivatives for remote sensing of scattering planetary atmospheres in thermal spectral region: General adjoint approach, *JQSRT,* **92,** 351-371, 2005.

Van Oss R.F., R.H.M. Voors, and R.J.D. Spurr, Ozone Profile Algorithm, OMI Algorithm Theoretical Basis Document. Volume II, OMI Ozone products (Bhartia PK, ed.), ATBD-OMI-02, Version 1.0, September 2001.

Van Oss, R.F., and R.J.D. Spurr, Fast and accurate 4 and 6 stream linearized discrete ordinate radiative transfer models for ozone profile retrieval, *J. Quant. Spectrosc. Radiat. Transfer*, **75**, 177-220, 2002.

Vermote, E. F., D. Tanré, J. L. Deuzé, M. Herman, and J. J. Morcrette. Second simulation of the satellite signal in the solar spectrum, 6S: an overview, IEEE *Trans. Geosci. Remote Sens.*, **35**, 675– 686, 1997.

Vestrucci, M., and C.E. Siewert, A numerical evaluation of an analytical representation of the components in a Fourier decomposition of the phase matrix for the scattering of polarized light, *JQSRT*, **31,** 177-183, 1984.

Wanner, W., X. Li, and A. Strahler, On the derivation of kernels for kernel-driven models of bidirectional reflectance, *J. Geophys. Res.*, **100**, 21077, 1995.

Wauben W.M.F., and J.W. Hovenier, Polarized radiation of an atmosphere containing randomly-oriented spheroids, *JQSRT*, **47**, 491-500, 1992.

Wiscombe, W. The delta-M method: rapid yet accurate radiative flux calculations for strongly asymmetric phase functions, *J. Atmos. Sci.*, **34**, 1408-1422, 1977.

Zhao, D., and Y. Toba, A spectral approach for determining altimeter wind speed model functions, *J. Ocean.*, **59**, 235-244, 2003.

6. Appendices

6.1 Tables

This section contains tables regarding: (1) input and output type structures; and (2) file-read character strings found in the input configuration file 2p7_VLIDORT_ReadInput.cfg.

Note. The user may notice a few variables that appear in the test drivers accompanying VLIDORT, but which are not found in the following input and output type structure tables. Such variables should be assigned default values (that is, .FALSE. for logical variables and zero for integer and floating-point variables) during VLIDORT's normal use. These variables are part of ongoing development work with VLIDORT, and are flagged as such in VLIDORT's input and output type structure files (in subdirectory "vlidort_def") by the phrase "RT Solutions use only".

6.1.1 VLIDORT I/O type structures

This section contains tables for VLIDORT input and output (I/O) type structures. Table 6.1 gives an overview of the categories of these I/O tables.

Table 6.1: VLIDORT I/O type structure table guide

Table Prefix	Input/Output Category
A	Basic fixed inputs
B	Basic modified inputs
C	Basic supplement inputs
D	Basic outputs
E	Linearized fixed inputs
F	Linearized modified inputs
G	Linearized supplement inputs
H	Linearized outputs

6.1.1.1 VLIDORT basic fixed inputs

Table A1: Type Structure VLIDORT_Fixed_Inputs

Name	Kind/Intent	Description
Bool	VLIDORT_Fixed_Boolean (I)	Type structure for fixed Boolean inputs (see Table A2).
Cont	VLIDORT_Fixed_Control (I)	Type structure for fixed control inputs (see Table A3).
Sunrays	VLIDORT_Fixed_Sunrays (I)	Type structure for fixed solar inputs (see Table A4).
UserVal	VLIDORT_Fixed_UserValues (I)	Type structure for fixed user value inputs (see Table A5).
Chapman	VLIDORT_Fixed_Chapman (I)	Type structure for fixed pseudo-spherical and refractive geometry inputs (see Table A6).
Optical	VLIDORT_Fixed_Optical (I)	Type structure for fixed atmospheric optical property inputs (see Table A7).
Write	VLIDORT_Fixed_Write (I)	Type structure for fixed write control inputs (see Table A8).

Table A2: Type Structure VLIDORT_Fixed_Boolean

Name	Kind/Intent	Description
DO_FULLRAD_MODE	Logical (I)	If set, VLIDORT will do a full radiance calculation.
DO_SSCORR_TRUNCATION	Logical (I)	If set, VLIDORT performs additional delta-M scaling on the single scatter RTE, applicable to either the nadir-only or the outgoing sphericity SS calculations.
DO_SSEXTERNAL	Logical (I)	If set, VLIDORT will use single scatter results computed externally in computations requiring single-scatter input.
DO_SSFULL	Logical (I)	If set, VLIDORT operates in single scatter mode, with no diffuse field. In this case, results are taken from SSCORR nadir or outgoing modules and, for the upwelling field, the direct-bounce correction (BRDF/Lambertian) is added. Note: DO_SSFULL overrides DO_FULLRAD_MODE.
DO_THERMAL_EMISSION	Logical (I)	If set, VLIDORT will compute atmospheric thermal emission with possible scattering.
DO_SURFACE_EMISSION	Logical (I)	If set, VLIDORT will compute surface thermal emission
DO_PLANE_PARALLEL	Logical (I)	Flag for use of the plane-parallel approximation for the direct beam attenuation. If not set, the atmosphere will be pseudo-spherical.
DO_UPWELLING	Logical (I)	If set, VLIDORT will compute upwelling output.
DO_DNWELLING	Logical (I)	If set, VLIDORT will compute downwelling output.
DO_QUAD_OUTPUT	Logical (I)	If set, VLIDORT will return radiances at quadrature stream angles.
DO_LAMBERTIAN_SURFACE	Logical (I)	Flag for choosing Lambertian surface properties.
DO_SURFACE_LEAVING	Logical (I)	Flag for choosing implementation of a surface leaving Stokes vector contribution.
DO_SL_ISOTROPIC	Logical (I)	Flag for choosing isotropic surface leaving contributions.

Table A3: Type Structure VLIDORT_Fixed_Control

Name	Kind/Intent	Description
TAYLOR_ORDER	Integer (I)	Order of Taylor polynomial used for computational smoothing in situations where numerical instability could lead to spurious results.
NSTOKES	Integer (I)	Number of Stokes vector parameters for which computations will be done.
NSTREAMS	Integer (I)	Number of quadrature streams in the cosine half space [0,1]. Must be ≤ symbolic dimension MAXSTREAMS.
NLAYERS	Integer (I)	Number of layers in atmosphere (NLAYERS = 1 is allowed). Must be ≤ symbolic dimension MAXLAYERS.
NFINELAYERS	Integer (I)	Number of fine layers subdividing coarse layering. Only for DO_SSCORR_OUTGOING, ≤ dimension MAXFINELAYERS.
N_THERMAL_COEFFS	Integer (I)	Number of coefficients used in treatment of blackbody emission in a layer. N_THERMAL_COEFFS = 1 implies constant within a layer; N_THERMAL_COEFFS = 2 implies a linear treatment. Maximum value allowed is currently 2.
VLIDORT_ACCURACY	Real*8 (I)	Accuracy criterion for convergence of Fourier series in relative azimuth. If for each output stream, addition of the mth Fourier term changes the total (Fourier-summed) intensity by a relative amount less than this value, then we pass the convergence test. For each solar angle, convergence is tested for intensities at all output stream

		angles, levels and azimuth angles. Once one solar beam result has converged, there is no further point in calculating any more Fourier terms for this beam,.

Table A4: Type Structure VLIDORT_Fixed_Sunrays

Name	Kind/Intent	Description
FLUX_FACTOR	Real*8 (I)	Beam source flux, the same value to be used for all solar angles. Normally set equal to 1 for "sun-normalized" output.

Table A5: Type Structure VLIDORT_Fixed_UserValues

Name	Kind/Intent	Description
N_USER_LEVELS	Integer (I)	Number of vertical output levels.

Table A6: Type Structure VLIDORT_Fixed_Chapman

Name	Kind/Intent	Description
HEIGHT_GRID	Real*8 (I)	Heights in [km] at layer boundaries, measured from TOA. Only required when Chapman function calculation of DELTA_SLANT_INPUT is done internally. Must be monotonically decreasing from TOA (this is checked).
PRESSURE_GRID	Real*8 (I)	Pressure in [mb] from TOA to BOA. Only required for internal Chapman factor calculation with refractive geometry.
TEMPERATURE_GRID	Real*8 (I)	Temperature in [K] from TOA to BOA. Only required for internal Chapman factor calculation with refractive geometry.
FINEGRID	Integer (I)	Integer array indicating number of fine layer divisions to be used in Snell's Law bending in the Chapman factor calculation with refraction. Recommended to set FINEGRID(N)=10. Refraction only.
RFINDEX_PARAMETER	Real*8 (I)	Only required for DO_REFRACTIVE_GEOMETRY option.

Table A7: Type structure VLIDORT_Fixed_Optical

Name	Kind/Intent	Description
DELTAU_VERT_INPUT (n)	Real*8 (I)	Vertical optical depth thickness values for all layers *n*.
GREEKMAT_TOTAL_INPUT (L,n,S)	Real*8 (I)	For all layers *n* and Stokes vector components *S*, Legendre moments of the phase function expansion multiplied by (2*L*+1); initial value (L=0) should always be 1 (checked).
THERMAL_BB_INPUT (n)	Real*8 (I)	Atmospheric thermal blackbody functions, levels *n*
LAMBERTIAN_ALBEDO	Real*8 (I)	Lambertian albedo values (between 0 and 1).
SURFACE_BB_INPUT	Real*8 (I)	Thermal input for surface.
ATMOS_WAVELENGTH	Real*8 (I)	Wavelength [nm] for atmospheric optical property inputs. This is a diagnostic number, playing no part in the RT calculation. However it is vital to set this value when using VLIDORT with wavelength-dependent BRDF and/or SLEAVE supplements - supplemental optical properties must be prepared at the same wavelength as used for VLIDORT optical input.

Table A8: Type structure VLIDORT_Fixed_Write

Name	Kind/Intent	Description
DO_DEBUG_WRITE	Logical (I)	Flag for writing VLIDORT debug output. (RT Solution use only)
DO_WRITE_INPUT	Logical (I)	Flag for sending certain VLIDORT general inputs to file.
INPUT_WRITE_FILENAME	Character (I)	File name for certain VLIDORT general inputs (up to 60 characters).
DO_WRITE_SCENARIO	Logical (I)	Flag for sending certain VLIDORT scenario inputs to file.
SCENARIO_WRITE_FILENAME	Character (I)	File name for certain VLIDORT scenario inputs (up to 60 characters).
DO_WRITE_FOURIER	Logical (I)	Flag for sending VLIDORT Fourier output to file. (not active)
FOURIER_WRITE_FILENAME	Character (I)	File name for certain VLIDORT Fourier output (up to 60 characters). (not active)
DO_WRITE_RESULTS	Logical (I)	Flag for sending VLIDORT general output to file.
RESULTS_WRITE_FILENAME	Character (I)	File name for VLIDORT general output (up to 60 characters).

6.1.1.2 VLIDORT basic modified inputs

Table B1: Type Structure VLIDORT_Modified_Inputs

Name	Kind/Intent	Description
MBool	VLIDORT_Modified_Boolean (IO)	Type structure for modified Boolean inputs (see Table B2).
MCont	VLIDORT_Modified_Control (IO)	Type structure for modified control inputs (see Table B3).
MSunrays	VLIDORT_Modified_Sunrays (IO)	Type structure for modified solar inputs (see Table B4).
MUserVal	VLIDORT_Modified_UserValues (IO)	Type structure for modified user value inputs (see Table B5).
MChapman	VLIDORT_Modified_Chapman (IO)	Type structure for modified pseudo-spherical and refractive geometry inputs (see Table B6).
MOptical	VLIDORT_Modified_Optical (IO)	Type structure for modified atmospheric optical property inputs (see Table B7).

Table B2: Type Structure VLIDORT_Modified_Boolean

Name	Kind/Intent	Description
DO_SSCORR_NADIR	Logical (IO)	If set, VLIDORT performs Nakajima-Tanaka single scatter correction, based on a regular pseudo-spherical geometry calculation (no outgoing correction). This flag applies equally to the stand-alone FO code and the VLIDORT native single-scatter correction code.
DO_SSCORR_OUTGOING	Logical (IO)	If set, VLIDORT performs Nakajima-Tanaka single scatter correction, based on a line-of-sight pseudo-spherical geometry calculation. This flag applies equally to the stand-alone FO code and the VLIDORT native single-scatter correction code.
DO_FO_CALC	Logical (IO)	If set, VLIDORT performs Nakajima-Tanaka single scatter correction using the RT solutions stand-alone single scatter

Name	Kind/Intent	Description
		(i.e. first order) code instead of VLIDORT's native single-scatter correction subroutines. Note - this flag is currently a hard-wired input that must be set by the user (if this FO option is desired). There is no character-string file-read input for this variable (hence no entry in Table K1 below).
DO_DOUBLE_CONVTEST	Logical (IO)	If set, the Fourier azimuth series is examined twice for convergence. If not set, a single test is made (saves an additional Fourier computation).
DO_SOLAR_SOURCES	Logical (IO)	Flag for solar beam source of light. Always TRUE for atmospheric scattering of sunlight,, but may be either TRUE or FALSE in thermal regime (not yet implemented)
DO_REFRACTIVE_GEOMETRY	Logical (IO)	Flag for using refractive geometry input in the pseudo-spherical approximation. Need Pressure/Temperature.
DO_CHAPMAN_FUNCTION	Logical (IO)	Flag for making an internal calculation of the slant path optical depths DELTA_SLANT_INPUT. If called, must specify height grid and earth radius.
DO_RAYLEIGH_ONLY	Logical (IO)	Flag for simulations in a Rayleigh atmosphere (molecules + trace gas absorptions). If set, only Fourier terms $m = 0, 1$ and 2 are calculated.
DO_DELTAM_SCALING	Logical (IO)	Flag for controlling use of the Delta-M scaling option. In most circumstances, this flag will be set.
DO_SOLUTION_SAVING	Logical (IO)	If set, then the RTE will not be solved if there is no scattering in certain layers for certain Fourier components (this is checked internally). Usage for example in Rayleigh atmosphere with one cloud layer.
DO_BVP_TELESCOPING	Logical (IO)	If set, then a reduced boundary value problem is solved for a set of contiguous scattering layers inside an otherwise transmittance-only atmosphere. Usage for example in Rayleigh atmosphere with one cloud layer.
DO_USER_VZANGLES	Logical (IO)	If set, there will be output at a number of off-quadrature zenith angles specified by user. This is the normal case.
DO_ADDITIONAL_MVOUT	Logical (IO)	Flag to produce integrated (mean-value) output *in addition* to radiance.
DO_MVOUT_ONLY	Logical (IO)	Flag to generate mean-value output only. Since such outputs are hemisphere-integrated, there is no need for user-defined angles, and only Fourier m=0 contributes.
DO_THERMAL_TRANSONLY	Logical (IO)	If set, VLIDORT will compute atmospheric thermal emission without scattering (transmission only).
DO_OBSERVATION_GEOMETRY	Logical (IO)	If set, VLIDORT will compute RT solutions only at observational geometry triplets specified by the user when computing RT solutions for multiple geometries. Used in conjunction with input variables N_USER_OBSGEOMS and USER_OBSGEOM_INPUT.

Table B3: Type Structure VLIDORT_Modified_Control

Name	Kind/Intent	Description
NGREEK_MOMENTS_INPUT	Integer (IO)	Number of Legendre expansion coefficients for the phase function. In the delta-M approximation, this must be at least 2*NSTREAMS to ensure delta-M truncation factor exists. NGREEK_MOMENTS_INPUT is used in exact single scatter, so should be > 2*NSTREAMS−1. Must be ≤ MAXMOMENTS_INPUT.

Table B4: Type Structure VLIDORT_Modified_Sunrays

Name	Kind/Intent	Description
N_SZANGLES	Integer (IO)	Number solar angles. Must ≤ symbolic dimension MAX_SZANGLES.
SZANGLES (b)	Real*8 (IO)	Array of *b* solar zenith angles (degrees). Checked internally range [0, 90).

Table B5: Type Structure VLIDORT_Modified_UserValues

Name	Kind/Intent	Description
N_USER_RELAZMS	Integer (IO)	Number of user-defined relative azimuth angles. Must not be greater than symbolic dimension MAX_USER_RELAZMS.
USER_RELAZMS (r)	Real*8 (IO)	Array of *r* user-defined relative azimuth angles (in degrees) for off-quadrature output. Ordering is not important. Must be between 0 and 180.
N_USER_VZANGLES	Integer (IO)	Number of user-defined viewing zenith angles. Must be not greater than symbolic dimension MAX_USER_VZANGLES.
USER_VZANGLES_INPUT (v)	Real*8 (IO)	Array of *v* user-defined viewing zenith angles (in degrees) for off-quadrature output. The ordering is not important (VLIDORT orders and checks this input internally). Must be between 0 and 90 degrees.
USER_LEVELS (o)	Real*8 (IO)	Array of *o* output level values. These can be in any order (VLIDORT sorts them in ascending order internally). Repetition of input values is also checked. See text for details.
GEOMETRY_SPECHEIGHT	Real*8 (IO)	This is the height in [km] above the Earth's surface at which input geometrical variables are specified. This may differ from the lowest value of the input height grid. Thus, for example, we may have geometrical angles at sea level, but we could be performing calculations down to cloud-top only – then, the input geometry needs to be adjusted to the lowest grid height whenever the outgoing single scatter option is set.
N_USER_OBSGEOMS	Integer (IO)	Number of user-defined observational geometry triplets. Must not be greater than the symbolic dimension MAX_USER_OBSGEOMS.
USER_OBSGEOM_INPUT (g,3)	Real*8 (IO)	Array of *g* user-defined observational geometry triplets (in degrees) for off-quadrature output. It consists of the geometry triplets (solar zenith angle, viewing angle, relative azimuth angle) for which RT solutions are desired.

Table B6: Type Structure VLIDORT_Modified_Chapman

Name	Kind/Intent	Description
CHAPMAN_FACTORS	Real*8 (IO)	Real array for Chapman factors computed externally.
EARTH_RADIUS	Real*8 (IO)	Earth's radius in [km]. Only required when DO_CHAPMAN_FUNCTION has been set. Checked internally to be in range [6320, 6420].

Table B7: Type structure VLIDORT_Modified_Optical

Name	Kind/Intent	Description
OMEGA_TOTAL_INPUT (n)	Real*8 (IO)	Single scattering albedos for all layers *n*. Should not be too close to 1.0; this is checked internally – OMEGA_SMALLNUM toggle generates a warning.

6.1.1.3 VLIDORT basic supplement I/O

Table C1: Type Structure VLIDORT_Sup_InOut

Name	Kind/Intent	Description
BRDF	VLIDORT_Sup_BRDF (I)	Type structure for BRDF supplement inputs (see Table C2).
SS	VLIDORT_Sup_SS (IO)	Type structure for single-scatter (SS) supplement (see Table C3).
SLEAVE	VLIDORT_Sup_SLEAVE (I)	Type structure for water-surface ("surface leaving") SLEAVE supplement (see Table C4).

Table C2: Type structure VLIDORT_Sup_BRDF

Name	Kind/Intent	Description
EXACTDB_BRDFUNC (S,a,b,s)	Real*8 (I)	Direct-bounce BRDF for Stokes vector component *S*, incident solar angle *s*, reflected line-of-sight angle *a*, and relative azimuth *b*.
BRDF_F_0 (M,S,k,s)	Real*8 (I)	Fourier components *M* of total BRDF for Stokes vector component *S*, incident solar angle *s* and reflected discrete ordinate *k*.
BRDF_F (M,S,k,j)	Real*8 (I)	Fourier components *M* of total BRDF for Stokes vector component *S*, incident discrete ordinate *j* and reflected discrete ordinate *k*.
USER_BRDF_F_0 (M,S,a,s)	Real*8 (I)	Fourier components *M* of total BRDF for Stokes vector component *S*, incident solar angle *s* and reflected line-of-sight zenith angle *a*.
USER_BRDF_F (M,S,a,j)	Real*8 (I)	Fourier components *M* of total BRDF for Stokes vector component *S*, incident discrete ordinate *j* and reflected line-of-sight zenith angle *a*.
EMISSIVITY (S,k)	Real*8 (I)	Surface emissivity for Stokes vector component *S* and emitted discrete ordinate *k*.
USER_EMISSIVITY (S,a)	Real*8 (I)	Surface emissivity for Stokes vector component *S* and emitted line-of-sight zenith angle *a*.

Table C3: Type structure VLIDORT_Sup_SS

Name	Kind/Intent	Description
STOKES_SS (t,v,S,d)	Real*8 (IO)	Stokes single scatter vector at output level *t*, output geometry *v*, Stokes parameter *S*, and direction *d*.
STOKES_DB (t,v,S)	Real*8 (IO)	Stokes direct-bounce vector at output level *t*, output geometry *v*, and Stokes parameter *S*.

Table C4: Type structure VLIDORT_Sup_SLEAVE

Name	Kind/Intent	Description
SLTERM_ISOTROPIC (S,s)	Real*8 (I)	Isotropic surface leaving radiance for Stokes vector component *S* and incident solar angle *s*.

SLTERM_USERANGLES (S,a,b,s)	Real*8 (I)	Surface-leaving radiance for Stokes vector component *S*, incident solar angle *s*, reflected line-of-sight angle *a*, and relative azimuth *b*.
SLTERM_F_0 (M,S,k,s)	Real*8 (I)	Fourier components *M* of diffuse-term surface-leaving radiance for Stokes vector component *S*, incident solar angle *s* and reflected discrete ordinate *k*.
USER_SLTERM_F_0 (M,S,a,s)	Real*8 (I)	Fourier components *M* of diffuse-term surface-leaving for Stokes vector component *S*, incident solar angle *s* and reflected line-of-sight zenith angle *a*.

6.1.1.4 VLIDORT basic outputs

Table D1: Type Structure VLIDORT_Outputs

Name	*Kind/Intent*	*Description*
Main	VLIDORT_Main_Outputs (O)	Type structure for main outputs (see Table D2).
Status	VLIDORT_Exception_Handling (O)	Type structure for exception-handling outputs (see Table D3).

Table D2: Type Structure VLIDORT_Main_Outputs

Name	*Kind/Intent*	*Description*
STOKES (t,v,s,S,d)	Real*8 (O)	Stokes vector at output level *t*, output geometry *v*, solar angle *s*, Stokes parameter *S*, and direction *d*.
MEAN_STOKES (t,s,S,d)	Real*8 (O)	Stokes mean vector (actinic flux) for output level *t*, solar angle *s*, Stokes parameter *S*, and direction *d*.
FLUX_STOKES (t,s,S,d)	Real*8 (O)	Stokes flux vector (regular flux) for output level *t*, solar angle *s*, Stokes parameter *S*, and direction *d*.
MEAN_DIRECT (t,s,S)	Real*8 (O)	Stokes direct mean vector (actinic flux) for output level *t*, solar angle *s*, and Stokes parameter *S*.
FLUX_DIRECT (t,s,S)	Real*8 (O)	Stokes direct flux vector (regular flux) for output level *t*, solar angle *s*, and Stokes parameter *S*.
FOURIER_SAVED (s)	Integer (O)	Number of Fourier moments required to calculate Stokes outputs for solar angle *s* to required degree of accuracy.
N_GEOMETRIES	Integer (O)	Number of scene geometries for which VLIDORT has calculated outputs.
SZA_OFFSETS (s)	Integer (O)	Solar zenith angle offsets for solar angle *s*.
VZA_OFFSETS (s,v)	Integer (O)	Viewing zenith angle offsets for solar angle *s* and output geometry *v*.
SOLARBEAM_BOATRANS(s)	Real*8 (O)	Solar beam transmittance to the bottom of the atmosphere, for solar angle *s*. This is a useful diagnostic output.

Table D3: Type Structure VLIDORT_Exception_Handling

Name	*Kind/Intent*	*Description*
STATUS_INPUTCHECK	Integer (O)	Overall status of input check.
NCHECKMESSAGES	Integer (O)	Number of input-check error messages.
CHECKMESSAGES	Character (O)	Array of input-check error messages.
ACTIONS	Character (O)	Array of input-check actions to take.
STATUS_CALCULATION	Integer (O)	Overall status of calculation.
MESSAGE	Character (O)	Calculation failure message.
TRACE_1	Character (O)	First subroutine trace for place of failure.
TRACE_2	Character (O)	Second subroutine trace for place of failure.
TRACE_3	Character (O)	Third subroutine trace for place of failure.

Table D4: Type Structure VLIDORT_Input_Exception_Handling

Name	Kind/Intent	Description
STATUS_INPUTREAD	Integer (O)	Overall status of input read.
NINPUTMESSAGES	Integer (O)	Number of input read error messages.
INPUTMESSAGES	Character (O)	Array of input-read error messages.
INPUTACTIONS	Character (O)	Array of input-read actions to take.

6.1.1.5 VLIDORT linearized fixed inputs

Table E1: Type Structure VLIDORT_Fixed_LinInputs

Name	Kind/Intent	Description
Cont	VLIDORT_Fixed_LinControl (I)	Type structure for fixed linearized control inputs (see Table E2).
Optical	VLIDORT_Fixed_LinOptical (I)	Type structure for fixed linearized atmospheric optical property inputs (see Table E3).

Table E2: Type structure VLIDORT_Fixed_LinControl

Name	Kind/Intent	Description
LAYER_VARY_FLAG (n)	Logical (I)	Flag for calculating profile Jacobians in layer *n.*
LAYER_VARY_NUMBER (n)	Integer (I)	Number of profile weighting functions in layer *n.*
N_TOTALCOLUMN_WFS	Integer (I)	Number of total column weighting functions. Should not exceed dimension MAX_ATMOSWFS.
N_TOTALPROFILE_WFS	Integer (I)	Number of profile weighting functions = Maximum value of LAYER_VARY_NUMBER. Should not exceed dimension MAX_ATMOSWFS.
N_SURFACE_WFS	Integer (I)	Equal to 1 if Lambertian calculation and surface linearization flag set. For linearized BRDF option, should be set equal to N_SURFACE_WFS in the BRDF structure. Should not exceed dimension MAX_SURFACEWFS.
N_SLEAVE_WFS	Integer (I)	Number of surface-leaving Jacobians.
COLUMNWF_NAMES	Character (I)	Names of column Jacobians (up to 31 characters).
PROFILEWF_NAMES	Character (I)	Names of profile Jacobians (up to 31 characters).

Table E3: Type structure VLIDORT_Fixed_LinOptical

Name	Kind/Intent	Description
L_DELTAU_VERT_INPUT (q,n)	Real*8 (I)	Relative variation in optical thickness for layer *n* with respect to varying parameter *q* in that layer.
L_OMEGA_TOTAL_INPUT (q,n)	Real*8 (I)	Relative variation in total single scattering albedo in layer *n*, with respect to parameter *q* in that layer.
L_GREEKMAT_TOTAL_INPUT (q,L,n,S)	Real*8 (I)	Relative variation in phase function moment coefficients. For Stokes vector component *S*, Legendre moment *L* in layer *n* with respect to parameter *q* in that layer.

6.1.1.6 VLIDORT linearized modified inputs

Table F1: Type Structure VLIDORT_Modified_LinInputs

Name	Kind/Intent	Description
MCont	VLIDORT_Modified_LinControl (IO)	Type structure for modified linearized control inputs (see Table F2).

Table F2: Type structure VLIDORT_Modified_LinControl

Name	Kind/Intent	Description
DO_SIMULATION_ONLY	Logical (IO)	Flag for output of standard radiative transfer quantities only (e.g. radiances and fluxes). If set, no Jacobians will be computed.
DO_COLUMN_LINEARIZATION	Logical (IO)	Flag for output of total column Jacobians.
DO_PROFILE_LINEARIZATION	Logical (IO)	Flag for output of profile Jacobians.
DO_ATMOS_LINEARIZATION	Logical (IO)	Flag for output of atmospheric Jacobians (the logical AND of the above COLUMN and PROFILE flags and the LTE flag from Table A11). If using subroutine `VLIDORT_L_INPUT_MASTER`, this is defined automatically.
DO_SURFACE_LINEARIZATION	Logical (IO)	Flag for output of surface Jacobians.
DO_LINEARIZATION	Logical (IO)	Flag for output of any Jacobians (the logical AND of the above ATMOS and SURFACE flags and the SURFBB flag from Table A11). If using subroutine `VLIDORT_L_INPUT_MASTER`, this is defined automatically.
DO_SLEAVE_WFS	Logical (IO)	Flag for output of surface-leaving Jacobians.
DO_ATMOS_LBBF	Logical (IO)	Flag for output of atmospheric blackbody Jacobians.
DO_SURFACE_LBBF	Logical (IO)	Flag for output of surface blackbody Jacobians.

6.1.1.7 VLIDORT linearized supplement I/O

Table G1: Type Structure VLIDORT_LinSup_InOut

Name	Kind/Intent	Description
BRDF	VLIDORT_LinSup_BRDF (I)	Type structure for linearized BRDF supplement inputs (see Table G2).
SS	VLIDORT_LinSup_SS (IO)	Type structure for linearized single-scatter (SS) supplement (see Table G3).
SLEAVE	VLIDORT_LinSup_SLEAVE (I)	Type structure for linearized surface leaving SLEAVE supplement (see Table G4).

Table G2: Type structure VLIDORT_LinSup_BRDF

Name	Kind/Intent	Description
LS_EXACTDB_BRDFUNC (q,S,a,b,s)	Real*8 (I)	Linearized direct-bounce BRDF for Stokes vector component S, incident solar angle s, reflected line-of-sight angle a, and relative azimuth b, w.r.t. surface property q.
LS_BRDF_F_0 (q,M,S,k,s)	Real*8 (I)	Linearized Fourier components M of total BRDF for Stokes vector component S, incident solar

Name	Kind/Intent	Description
		angle *s* and reflected discrete ordinate *k,* w.r.t. surface property *q.*
LS_BRDF_F (q,M,S,k,j)	Real*8 (I)	Linearized Fourier components *M* of total BRDF for Stokes vector component *S*, incident discrete ordinate *j* and reflected discrete ordinate *k,* w.r.t. surface property *q.*
LS_USER_BRDF_F_0 (q,M,S,a,s)	Real*8 (I)	Linearized Fourier components *M* of total BRDF for Stokes vector component *S*, incident solar angle *s* and reflected line-of-sight zenith angle *a*, w.r.t. surface property *q.*
LS_USER_BRDF_F (q,M,S,a,j)	Real*8 (I)	Linearized Fourier components *M* of total BRDF for Stokes vector component *S*, incident discrete ordinate *j* and reflected line-of-sight zenith angle *a,* w.r.t. surface property *q.*
LS_EMISSIVITY (q,S,k)	Real*8 (I)	Linearized surface emissivity for Stokes vector component *S* and emitted discrete ordinate *k,* w.r.t. surface property *q.*
LS_USER_EMISSIVITY (q,S,a)	Real*8 (I)	Linearized surface emissivity for Stokes vector component *S* and emitted line-of-sight zenith angle *a*, w.r.t. surface property *q.*

Table G3: Type Structure VLIDORT_LinSup_SS

Name	Kind/Intent	Description
Col	VLIDORT_LinSup_SS_Col (IO)	Type structure for linearized single-scatter atmospheric column Jacobians (see Table G3-1).
Prof	VLIDORT_LinSup_SS_Prof (IO)	Type structure for linearized single-scatter atmospheric profile Jacobians (see Table G3-2).
Surf	VLIDORT_LinSup_SS_Surf (IO)	Type structure for linearized single-scatter surface Jacobians (see Table G3-3).

Table G3-1: Type structure VLIDORT_LinSup_SS_Col

Name	Kind/Intent	Description
COLUMNWF_SS (q,t,v,S,d)	Real*8 (IO)	Column Jacobians of single-scatter Stokes vector w.r.t. variable *q*, at output level *t*, geometry *v*, Stokes parameter *S*, and direction *d*.
COLUMNWF_DB (q,t,v,S)	Real*8 (IO)	Column Jacobians of direct-bounce Stokes vector w.r.t. variable *q*, at output level *t*, geometry *v*, and Stokes parameter *S*.

Table G3-2: Type structure VLIDORT_LinSup_SS_Prof

Name	Kind/Intent	Description
PROFILEWF_SS (q,n,t,v,S,d)	Real*8 (IO)	Profile Jacobians of single-scatter Stokes vector w.r.t. variable *q* in layer *n*, at output level *t*, geometry *v*, Stokes parameter *S*, and direction *d*.
PROFILEWF_DB (q,n,t,v,S)	Real*8 (IO)	Profile Jacobians of direct-bounce Stokes vector w.r.t. variable *q* in layer *n*, at output level *t*, geometry *v*, and Stokes parameter *S*.

Table G3-3: Type structure VLIDORT_LinSup_SS_Surf

Name	Kind/Intent	Description
SURFACEWF_DB (r,t,v,S)	Real*8 (IO)	Surface Jacobians of direct-bounce Stokes vector w.r.t. variable r, at output level t, geometry v, and Stokes parameter S.

Table G4: Type structure VLIDORT_LinSup_SLEAVE

Name	Kind/Intent	Description
LSSL_SLTERM_ISOTROPIC (q,S,s)	Real*8 (I)	Linearized Isotropic surface-leaving radiance for Stokes vector component S and incident solar angle s, w.r.t. surface property q.
LSSL_SLTERM_USERANGLES (q,S,a,b,s)	Real*8 (I)	Linearized surface-leaving radiance for Stokes vector component S, incident solar angle s, reflected line-of-sight angle a, and relative azimuth b, w.r.t. surface property q.
LSSL_SLTERM_F_0 (q,M,S,k,s)	Real*8 (I)	Linearized Fourier components M of surface-leaving radiance for Stokes vector component S, incident solar angle s and reflected discrete ordinate k, w.r.t. surface property q.
LSSL_USER_SLTERM_F_0 (q,M,S,a,s)	Real*8 (I)	Linearized Fourier components M of surface-leaving radiance for Stokes vector component S, incident solar angle s and reflected line-of-sight zenith angle a, w.r.t. surface property q.

6.1.1.8 VLIDORT linearized outputs

Table H1: Type Structure VLIDORT_LinOutputs

Name	Kind/Intent	Description
Col	VLIDORT_LinCol (O)	Type structure for linearized atmospheric column outputs (see Table H2).
Prof	VLIDORT_LinProf (O)	Type structure for linearized atmospheric profile outputs (see Table H3).
Atmos	VLIDORT_LinAtmos(O)	Type structure for linearized atmospheric general outputs (see Table H4).
Surf	VLIDORT_LinSurf (O)	Type structure for linearized surface outputs (see Table H5).

Table H2: Type Structure VLIDORT_LinCol

Name	Kind/Intent	Description
COLUMNWF (q,t,v,S,d)	Real*8 (O)	Column Jacobians of Stokes vector with respect to **_total_** atmospheric variable q, at output level t, geometry v, Stokes parameter S, and direction d.
MINT_COLUMNWF (q,t,s,S,d)	Real*8 (O)	Atmospheric Jacobians of Stokes mean vector (actinic flux) w.r.t. atmospheric variable q, at output level t, solar beam s, Stokes parameter S, and direction d.
FLUX_COLUMNWF (q,t,s,S,d)	Real*8 (O)	Atmospheric Jacobians of Stokes flux vector (regular flux) w.r.t. atmospheric variable q, at output level t, solar beam s, Stokes parameter S, and direction d.
MINT_COLUMNWF_DIRECT (q,t,s,S)	Real*8 (O)	Atmospheric Jacobians of Stokes direct mean vector (actinic flux) w.r.t. atmospheric variable q, at output level t,

		solar beam *s*, and Stokes parameter *S*.
FLUX_COLUMNWF_DIRECT (q,t,s,S)	Real*8 (O)	Atmospheric Jacobians of Stokes direct flux vector (regular flux) w.r.t. atmospheric variable *q*, at output level *t*, solar beam *s*, and Stokes parameter *S*.

Table H3: Type Structure VLIDORT_LinProf

Name	Kind/Intent	Description
PROFILEWF (q,n,t,v,S,d)	Real*8 (O)	Profile Jacobians of Stokes vector with respect to ***profile*** atmospheric variable *q* in layer *n,* at output level *n,* geometry *v*, Stokes parameter *S*, and direction *d*.
MINT_PROFILEWF (q,n,t,s,S,d)	Real*8 (O)	Atmospheric Jacobians of Stokes mean vector (actinic flux) w.r.t. variable *q* in layer *n*, at output level *t*, solar beam *s*, Stokes parameter *S*, and direction *d*.
FLUX_PROFILEWF (q,n,t,s,S,d)	Real*8 (O)	Atmospheric Jacobians of Stokes flux vector (regular flux) w.r.t. atmospheric variable *q* in layer *n*, at output level *t*, solar beam *s*, Stokes parameter *S*, and direction *d*.
MINT_PROFILEWF_DIRECT (q,n,t,s,S)	Real*8 (O)	Atmospheric Jacobians of Stokes direct mean vector (actinic flux) w.r.t. atmospheric variable *q* in layer *n*, at output level *t*, solar beam *s*, Stokes parameter *S*, and direction *d*.
FLUX_PROFILEWF_DIRECT (q,n,t,s,S)	Real*8 (O)	Atmospheric Jacobians of Stokes direct flux vector (regular flux) w.r.t. atmospheric variable *q* in layer *n*, at output level *t*, solar beam *s*, Stokes parameter *S*, and direction *d*.

Table H4: Type Structure VLIDORT_LinAtmos

Name	Kind/Intent	Description
ABBWFS_JACOBIANS (o,v,n,S,d)	Real*8 (O)	Atmospheric-blackbody radiance Jacobians at output level *o*, geometry *v*, level *n*, Stokes parameter *S*, and direction *d*.
ABBWFS_FLUXES (o,f,n,S,d)	Real*8 (O)	Atmospheric-blackbody flux Jacobians at output level *o*, level *n*, flux type *f*, Stokes parameter *S*, and direction *d*. (Note: f=1 is for actinic flux, f=2 regular flux)

Table H5: Type Structure VLIDORT_LinSurface

Name	Kind/Intent	Description
SURFACEWF (r,t,v,S,d)	Real*8 (O)	Surface Jacobians of Stokes vector with respect to ***surface*** variable *r,* at output level *t*, geometry *v*, Stokes parameter *S*, and direction *d*.
MINT_SURFACEWF (r,t,s,S,d)	Real*8 (O)	Surface Jacobians of Stokes mean vector (actinic flux) w.r.t. variable *r*, at output level *t*, solar beam *s*, Stokes parameter *S*, and direction *d*.
FLUX_SURFACEWF (r,t,s,S,d)	Real*8 (O)	Surface Jacobians of Stokes flux vector (regular flux) w.r.t. variable r, at output level *t*, solar beam *s*, Stokes parameter *S*, and direction *d*.
SBBWFS_JACOBIANS (o,v,S,d)	Real*8 (O)	Surface-blackbody radiance Jacobians at output level *o*, geometry *v*, Stokes parameter *S*, and direction *d*.
SBBWFS_FLUXES (o,f,S,d)	Real*8 (O)	Surface-blackbody flux Jacobians at output level *o*, flux type *f*, Stokes parameter *S*, and direction *d*. (Note: f=1 is for actinic flux, f=2 regular flux)

6.1.2 VLIDORT file-read character strings

This section contains tables for file-read character strings found in the input configuration file 2p7_VLIDORT_ReadInput.cfg and their associated VLIDORT I/O type structure variables. Table 6.2 gives an overview of the categories of these tables.

Table 6.2: VLIDORT input configuration file table guide

Table Prefix	Input/Output Category
J	Basic fixed inputs
K	Basic modified inputs
L	Linearized fixed inputs
M	Linearized modified inputs

6.1.2.1 VLIDORT basic fixed inputs

Table J1: File-read Character strings for Fixed Boolean variables (Table A2)

Name	Kind	Character string in Configuration file
DO_FULLRAD_MODE	Logical	Do full Stokes vector calculation?
DO_SSCORR_TRUNCATION	Logical	Do delta-M scaling on single scatter corrections?
DO_SSEXTERNAL	Logical	Do external single scatter calculation?
DO_SSFULL	Logical	Do full-up single scatter calculation?
DO_THERMAL_EMISSION	Logical	Do thermal emission?
DO_SURFACE_EMISSION	Logical	Do surface emission?
DO_PLANE_PARALLEL	Logical	Do plane-parallel treatment of direct beam?
DO_UPWELLING	Logical	Do upwelling output?
DO_DNWELLING	Logical	Do downwelling output?
DO_LAMBERTIAN_SURFACE	Logical	Do Lambertian surface?
DO_SURFACE_LEAVING	Logical	Do surface-leaving term?
DO_SL_ISOTROPIC	Logical	Do isotropic surface-leaving term?

Table J2: File-read Character strings for Fixed Control variables (Table A3)

Name	Kind	Character string in Configuration file
TAYLOR_ORDER	Integer	Number of small-number terms in Taylor series expansions
NSTOKES	Integer	Number of Stokes vector components
NSTREAMS	Integer	Number of half-space streams
NLAYERS	Integer	Number of atmospheric layers
NFINELAYERS	Integer	Number of fine layers (outgoing sphericity option only)
N_THERMAL_COEFFS	Integer	Number of thermal coefficients
VLIDORT_ACCURACY	Real*8	Fourier series convergence

Table J3: File-read Character strings for Fixed Sunrays variables (Table A4)

Name	Kind	Character string in Configuration file
FLUX_FACTOR	Real*8	Solar flux constant

Table J4: File-read Character strings for Fixed UserValues variables (Table A5)

Name	Kind	Character string in Configuration file
N_USER_LEVELS	Integer	Number of user-defined output levels

Table J5: File-read Character strings for *some* Fixed Chapman variables (Table A6)

Name	Kind	Character string in Configuration file
RFINDEX_PARAMETER	Real*8	Refractive index parameter

Table J6: File-read Character strings for *some* Fixed Optical variables (Table A7)

Name	Kind	Character string in Configuration file
LAMBERTIAN_ALBEDO	Real*8	Lambertian albedo

Table J7: File-read Character strings for Fixed Write variables (Table A8)

Name	Kind	Character string in Configuration file
DO_DEBUG_WRITE	Logical	Do debug write? (RT Solution use only)
DO_WRITE_INPUT	Logical	Do input control write?
DO_WRITE_SCENARIO	Logical	Do input scenario write?
DO_WRITE_FOURIER	Logical	Do Fourier component output write? (not active)
DO_WRITE_RESULTS	Logical	Do results write?
INPUT_WRITE_FILENAME	Character	filename for input write
SCENARIO_WRITE_FILENAME	Character	filename for scenario write
FOURIER_WRITE_FILENAME	Character	filename for Fourier output write (not active)
RESULTS_WRITE_FILENAME	Character	filename for main output

6.1.2.2 VLIDORT basic modified inputs

Table K1: File-read Character strings for Modified Boolean variables (Table B2)

Name	Kind	Character string in Configuration file
DO_SSCORR_NADIR	Logical	Do nadir single scatter correction?
DO_SSCORR_OUTGOING	Logical	Do outgoing single scatter correction?
DO_DOUBLE_CONVTEST	Logical	Do double convergence test?
DO_SOLAR_SOURCES	Logical	Use solar sources?
DO_REFRACTIVE_GEOMETRY	Logical	Do refractive geometry?
DO_CHAPMAN_FUNCTION	Logical	Do internal Chapman function calculation?
DO_RAYLEIGH_ONLY	Logical	Do Rayleigh atmosphere only?
DO_DELTAM_SCALING	Logical	Do delta-M scaling?
DO_SOLUTION_SAVING	Logical	Do solution saving?
DO_BVP_TELESCOPING	Logical	Do boundary-value telescoping?
DO_USER_VZANGLES	Logical	Use user-defined viewing zenith angles?
DO_ADDITIONAL_MVOUT	Logical	Do mean-value output additionally?
DO_MVOUT_ONLY	Logical	Do only mean-value output?
DO_THERMAL_TRANSONLY	Logical	Do thermal emission, transmittance only?
DO_OBSERVATION_GEOMETRY	Logical	Do Observation Geometry?

Table K2: File-read Character strings for Modified Control variables (Table B3)

Name	Kind	Character string in Configuration file
NGREEK_MOMENTS_INPUT	Integer	Number of scattering matrix expansion coefficients

Table K3: File-read Character strings for Modified Sunrays variables (Table B4)

Name	Kind	Character string in Configuration file
N_SZANGLES	Integer	Number of solar zenith angles
SZANGLES	Real*8	Solar zenith angles (degrees)

Table K4: File-read Character strings for Modified UserValues variables (Table B5)

Name	Kind	Character string in Configuration file
N_USER_RELAZMS	Integer	Number of user-defined relative azimuth angles
USER_RELAZMS	Real*8	User-defined relative azimuth angles (degrees)
N_USER_VZANGLES	Integer	Number of user-defined viewing zenith angles
USER_VZANGLES_INPUT	Real*8	User-defined viewing zenith angles (degrees)
USER_LEVELS	Real*8	User-defined output levels
GEOMETRY_SPECHEIGHT	Real*8	Input geometry specification height (km)
N_USER_OBSGEOMS	Integer	Number of Observation Geometry inputs
USER_OBSGEOM_INPUT	Real*8	Observation Geometry inputs

Table K5: File-read Character strings for *some* Modified Chapman variables (Table B6)

Name	Kind	Character string in Configuration file
EARTH_RADIUS	Real*8	Earth radius (km)

6.1.2.3. VLIDORT linearized fixed inputs

Table L1: File-read Character strings for *some* Fixed LinControl variables (Table E2)

Name	Kind	Character string in Configuration file
N_TOTALCOLUMN_WFS	Integer	Number of atmospheric column weighting functions (total)
N_TOTALPROFILE_WFS	Integer	Number of atmospheric profile weighting functions (total)
COLUMNWF_NAMES	Character	Atmospheric column Jacobian names (character*31)
PROFILEWF_NAMES	Character	Atmospheric profile Jacobian names (character*31)

6.1.2.4 VLIDORT linearized modified inputs

Table M1: File-read Character strings for *some* Modified LinControl variables (Table F2)

Name	Kind	Character string in Configuration file
DO_SIMULATION_ONLY	Logical	Do simulation only?
DO_COLUMN_LINEARIZATION	Logical	Do atmospheric column weighting functions?
DO_PROFILE_LINEARIZATION	Logical	Do atmospheric profile weighting functions?
DO_SURFACE_LINEARIZATION	Logical	Do surface property weighting functions?
DO_ATMOS_LBBF	Logical	Atmospheric BB emission weighting functions?
DO_SURFACE_LBBF	Logical	Surface BB emission weighting functions?

6.2 Environment programs

6.2.1 Programs for VLIDORT scalar tests

The text below provides gives some description of the current scalar tests for VLIDORT (see Section 4.3.4). General comments are followed by more specific information regarding each test.

General Comments

There are 23 levels from 50.0 km down to 0.0 km, with a pre-prepared atmosphere with height, layer optical depth for molecules and layer single scatter albedo for molecules. The lowest 6 layers have a uniform slab of aerosol, inserted by hand, with the phase function determined through the asymmetry parameter (up to 81 expansion coefficients). Aerosol control values are:

    Total aerosol optical depth over 6 layers = 0.5
    Single scattering albedo of aerosol   = 0.95
    Asymmetry parameter for aerosol   = 0.80

The surface albedo is 0.05. There are 36 geometries consisting of combinations of the following angles (in degrees):

    4 Solar zenith angles (in degrees)   - 35.0, 67.0, 75.0, 82.0
    3 User-defined viewing zenith angles (in degrees)   - 10.0, 20.0, 40.0
    3 User-defined relative azimuth angles (in degrees) - 0.0, 90.0, 180.0

There are 5 levels of output:

    Level = 0.0   --> Top of first layer   ----> This is TOA
    Level = 1.0   --> Bottom of first layer
    Level = 2.5   --> Half-way into third layer
    Level = 22.5 --> Half-way into 23rd layer
    Level = 23.0 --> Bottom of 23rd layer   ----> This is BOA

Upwelling and downwelling field is specified throughout. Actinic and regular Fluxes are specified for every level, both up and down. These fluxes are integrated outputs and valid for each solar zenith angle (SZA).

Note that for results obtained using the FO single-scatter option, level output is only available at layer boundaries (0.0, 1.0 and 23.0); this is because the FO code does not have capability at the present time for off-boundary output.

We now turn to more specific information regarding these tests.

Solar Tester

Master program   : 2p7_solar_tester.f90
Executable   : s2p7_solar_tester.exe
Output file   : results_solar_tester.all

This is an intensity-only calculation with 6 task options:

  Option 1: No single-scatter correction, no delta-M scaling.
  Option 2: No single-scatter correction, with delta-M scaling.
  Option 3: Ingoing-only single-scatter correction, with delta-M scaling
              (SUN in curved atmosphere).
  Option 4: In/outgoing single-scatter correction, with delta-M scaling
              (SUN+LOS in curved atmosphere).
  Option 5: Ingoing-only FO single-scatter, with delta-M scaling
  Option 6: In/outgoing FO single-scatter computation, with delta-M scaling
  Option 7: Same as task #4, but using solution-saving.
  Option 8: Same as task #4, but using boundary-value problem telescoping.

The output file contains intensities for all 8 task options, all 36 geometries and (excepting the FO cases, where 3 output levels are present) all 5 output levels. The integrated output (actinic and regular fluxes) are produced only for options 1 and 2 (not dependent on the SS correction), but for all SZAs and all 5 output levels.

Linearized Solar Profile and Column Tester

Master program   : 2p7_solar_lpcs_tester.f90
Executable   : s2p7_solar_lpcs_tester.exe
Output file   : results_solar_lcs_tester.all
                       results_solar_lcs_tester.all_FO
                       results_solar_lps_tester.all
                       results_solar_lps_tester.all_FO

The first part is a calculation for intensity, profile Jacobians and surface albedo Jacobian.

There are 2 types of NORMALIZED profile weighting functions:
    1. w.r.t. layer trace gas absorption optical depths.
    2 w.r.t. layer aerosol optical depths in bottom 6 layers.

There is 1 surface weighting function (this is UNNORMALIZED!):
    1. w.r.t. Lambertian albedo.

We use the in/outgoing single-scatter correction with delta-M scaling (this is a standard default and the most accurate calculation). The first option is the baseline calculation of intensity and all Jacobians. The other options are designed to test Jacobians by finite differencing. They are:

  Option 1: Finite difference, perturb Lambertian albedo.
  Option 2: Finite difference, perturb molecular absorption in Layer 1.
  Option 3: Finite difference, perturb molecular absorption in Layer 21.
  Option 4: Finite difference, perturb aerosol optical depth in layer 23.

The output file contains (for all 36 geometries and 5 output levels) the baseline intensities, baseline Analytic weighting functions (AJ1, AJ2 etc.), and the corresponding finite difference

weighting functions (FD1, FD2, etc…). The integrated output (actinic and regular fluxes) are output for all SZAs and all 5 output levels.

The second part is a calculation for intensity, column Jacobians and surface albedo Jacobian.

There are 2 NORMALIZED column weighting functions:
  1. w.r.t. total trace gas absorption optical depth of the whole atmosphere.
  2 w.r.t. total aerosol optical depth in bottom 6 layers.

There is 1 surface weighting function (this is UNNORMALIZED!):
  1. w.r.t. Lambertian albedo.

We are using the in/outgoing single-scatter correction with delta-M scaling (this is a standard default, and the most accurate calculation). The first option is the baseline calculation of intensity, 2 column Jacobians, and 1 Surface Jacobian. The other options are designed to test Jacobians by finite differencing. They are:

  Option 5: Finite difference, perturb Lambertian albedo.
  Option 6: Finite difference, perturb total molecular absorption optical depth.
  Option 7: Finite difference, perturb total aerosol optical depth.

The output file contains (for all 36 geometries and 5 output levels) the baseline intensities, baseline Analytic weighting functions (AJ1, AJ2 etc...), and the corresponding finite difference weighting functions (FD1, FD2 etc…). The integrated output (actinic and regular fluxes) are output for all SZAs and all 5 output levels.

Note that separate output files are generated for the above linearized profile and linearized column results, depending on whether they were obtained using VLIDORT's native SS correction routines (e.g. the results in "results_solar_lcs_tester.all") or using the newer first order (FO) code (e.g. results in "results_solar_lcs_tester.all_FO"). These are generated automatically when the driver is run. Note again that FO results are only available at 3 output levels.

Thermal Tester

Master program   : 2p7_thermal_tester.f90
Executable   : s2p7_thermal_tester.exe
Output file   : results_thermal_tester.all

This is an intensity-only calculation with 8 task options:

  Option 1: Thermal only, with scattering, + delta-M, Lambertian.
  Option 2: Thermal transmittance only.
  Option 3: Crossover Ingoing-only Single-scatter correction + delta-M, Lamb.
  Option 4: Crossover In/Outgoing Single-scatter correction + delta-M, Lamb.
  Option 5: Crossover In/Outgoing Internal Single-scatter correction + delta-M, BRDF1.
  Option 6: Crossover In/Outgoing Internal Single-scatter correction + delta-M, BRDF3.

Option 7: Crossover In/Outgoing FO Single-scatter correction + delta-M, BRDF1.
Option 8: Crossover In/Outgoing FO Single-scatter correction + delta-M, BRDF3.

The output file contains intensities for all 8 of these task options, all 36 geometries and all 5 output levels (again, only 3 levels for the FO results). The integrated output (actinic and regular fluxes) are output for all SZAs and all 5 output levels.

Linearized Thermal Profile and Column Tester

Master program   : 2p7_thermal_lpcs_tester.f90
Executable   : s2p7_thermal_lpcs_tester.exe
Output file   : results_thermal_lcs_tester.all
                         results_thermal_lcs_tester.all_FO
                         results_thermal_lps_tester.all
                         results_thermal_lps_tester.all_FO

The first part is a calculation for intensity, profile Jacobians and surface albedo Jacobian.

There are 2 types of NORMALIZED profile weighting functions:
   1. w.r.t. layer trace gas absorption optical depths.
   2 w.r.t. layer aerosol optical depths in bottom 6 layers.

There is 1 surface weighting function (this is UNNORMALIZED!):
   1. w.r.t. Lambertian albedo.

We are using the in/outgoing single-scatter correction with delta-M scaling (this is a standard default, and the most accurate calculation).   The first option is the baseline calculation of intensity, 3 profile Jacobians, and 1 Surface Jacobian. The other options are designed to test Jacobians by finite differencing. They are:

 Option 1: Finite difference, perturb Lambertian albedo.
 Option 2: Finite difference, perturb molecular absorption in Layer 1.
 Option 3: Finite difference, perturb molecular absorption in Layer 21.
 Option 4: Finite difference, perturb aerosol optical depth in layer 23.

The output file contains (for all 36 geometries and 5 output levels) the baseline intensities, baseline Analytic weighting functions (AJ1, AJ2, etc…), and the corresponding finite difference weighting functions (FD1, FD2, etc…). The integrated output (actinic and regular fluxes) are output for all SZAs and all 5 output levels.

The second part is a calculation for intensity, column Jacobians and surface albedo Jacobian.

There are 2 NORMALIZED column weighting functions:
   1. w.r.t. total trace gas absorption optical depth of the whole atmosphere.
   2 w.r.t. total aerosol optical depth in bottom 6 layers.

There is 1 surface weighting function (this is UNNORMALIZED!):
   1. w.r.t. Lambertian albedo.

We are using the in/outgoing single-scatter correction with delta-M scaling (this is a standard default, and the most accurate calculation). The first call is the baseline calculation of intensity, 2 column Jacobians, and 1 Surface Jacobian. The 3 threads are designed to test Jacobians by finite differencing. They are:

  Option 5: Finite difference, perturb Lambertian albedo.
  Option 6: Finite difference, perturb total molecular absorption optical depth.
  Option 7: Finite difference, perturb total aerosol optical depth.

The output file contains (for all 36 geometries and 5 output levels) the baseline intensities, baseline Analytic weighting functions (AJ1, AJ2, etc…), and the corresponding finite difference weighting functions (FD1, FD2, etc…). The integrated output (actinic and regular fluxes) are output for all SZAs and all 5 output levels.

As in the case of the linearized solar driver, separate output files are generated for the above linearized profile and linearized column results depending on whether they were obtained using VLIDORT's native SS correction routines (e.g. the results in "results_thermal_lcs_tester.all") or using the newer first order (FO) code (e.g. results in "results_thermal_lcs_tester.all_FO"). These are also generated automatically when the driver is run.

Standard and Linearized BRDF Surface Tester

Master program   : 2p7_brdfplus_tester.f90
Executable   : s2p7_brdfplus_tester.exe
Output files   : results_brdf_supcheck.res
                       results_brdf_supcheck.wfs
                       results_brdfplus_tester.all

The surface BRDF is one consisting of three BRDF kernels: a Ross-thin kernel, a Li-dense kernel, and a Cox-Munk kernel.

The first part of the test program has the BRDF supplement code calculate the BRDF and the BRDF weighting functions to be passed to VLIDORT later in the test. These two sets of results are output to two BRDF output files.

The BRDF supplement output check file "results_brdf_supcheck.res" has the following format:

First, exact direct beam BRDF reflectance output for each of the 36 geometries.

Also, for one azimuth angle, the Fourier components of the BRDF reflectance in the test scenario are output for each of the 10 upwelling quadrature angles (QUAD), 4 solar zenith angles (SZA), and 3 user-specified angles (VZA) in the following format:

Ten pairs of BRDF results while varying QUAD versus:

* QUAD (ten results).
* SZA (four results).

Three pairs of BRDF results while varying VZA versus:

* QUAD (ten results).
* SZA (four results).

In the BRDF output file "results_brdf_supcheck.wfs", we have a similar configuration of results as in "results_brdf_supcheck.res", but here there are 6 sets of results, each corresponding to one of the following 6 surface Jacobians:

1. Ross-thin kernel - kernel factor.
2. Li-dense kernel - kernel factor.
3. Li-dense kernel - kernel parameter #1.
4. Li-dense kernel - kernel parameter #2.
5. Cox-Munk kernel - kernel factor.
6. Cox-Munk kernel - kernel parameter #1.

The Jacobians displayed are a result of the following BRDF weighting function inputs in the BRDF input configuration file "2p7_BRDF_ReadInput.cfg":

VLIDORT - Kernels, indices, # pars, Jacobian flags
Ross-thin   2 0 T   F F F
Li-dense   5 2 T   T T F
Cox-Munk   9 2 T   T F F

The second part of the test program is an intensity and surface albedo weighting function calculation done by VLIDORT.

There are six UNNORMALIZED surface weighting functions calculated:
    1. w.r.t. Ross-thin kernel - kernel factor.
    2. w.r.t. Li-dense kernel - kernel factor.
    3. w.r.t. Li-dense kernel - kernel parameter #1.
    4. w.r.t. Li-dense kernel - kernel parameter #2.
    5. w.r.t. Cox-Munk kernel - kernel factor.
    6. w.r.t. Cox-Munk kernel - kernel parameter #1.

We are using the in/outgoing single-scatter correction with delta-M scaling (this is a standard default, and the most accurate calculation). The first option is the baseline calculation of intensity and all 6 surface Jacobians. The options are designed to test Jacobians by finite differencing. They are:

 Option 1: Finite difference, perturb Ross-thin kernel - kernel factor.
 Option 2: Finite difference, perturb Li-dense kernel - kernel factor.
 Option 3: Finite difference, perturb Li-dense kernel - kernel parameter #1.
 Option 4: Finite difference, perturb Li-dense kernel - kernel parameter #2.
 Option 5: Finite difference, perturb Cox-Munk kernel - kernel factor.

Option 6: Finite difference, perturb Cox-Munk kernel - kernel parameter #1.

The output file contains (for all 36 geometries and 5 output levels) the baseline intensities, baseline Analytic weighting functions (AJ1, AJ2, etc…), and the corresponding finite difference weighting functions (FD1, FD2, etc…). The integrated output (actinic and regular fluxes) are output for all SZAs and all 5 output levels.

6.2.2 Programs for VLIDORT vector tests

Since the five vector tests are very similar to the scalar tests above, a detailed description of each test will not be repeated here; however, we make a few comments regarding these inputs.

In these tests, the main difference is a more sophisticated treatment of aerosol in the bottom six layers. Here, expansion coefficients for the Greek scattering matrix were generated by a Mie scattering code and are read in from the input file ProblemIII.Moms. [Mie results were generated for a 2-parameter Gamma-function size distribution with an effective radius of 1.05 µm, an effective variance of 0.07 µm, and a refractive index of $1.43+ 0.001i$.

An additional test is performed to compare the results of VLIDORT with those found in Siewert (2000c). In this test and in that work, the optical property data set "Problem IIA" of Wauben and Hovenier (1992) is used. This benchmark test considers a 1-layer "slab" problem with scattering by randomly-oriented oblate spheroids with an aspect ratio of 1.999987, a size parameter of 3, and refractive index of $1.53-0.006i$. The tables of results generated by VLIDORT for this case may then be compared with Tables 2-9 of Siewert (2000c).

6.2.3 Solar programs to test using VLIDORT in an OpenMP environment

The two solar programs in "vlidort_v_test" which use OpenMP directives perform computations similar to the two serial solar programs in the same directory, but for a simulated set of wavelengths as one might do in computing the radiative transfer solutions for a given wavelength band. The main point of these programs is to give the user guidance in setting up VLIDORT in an OpenMP parallel computing framework to accelerate the performance of such computations. The output files contain the same output as the serial versions of the programs, but for multiple OpenMP threads (currently, there are outputs for two threads).

Before attempting to run these VLIDORT package programs, we make the following recommendations to the user:

- Check the version of OpenMP installed on your system. The OpenMP facilities used here are compatible with OpenMP version 3.1 or later.

- For running the codes with OpenMP, it is necessary to use the appropriate compiler flags in the makefile. For "gfortran", it is "`-fopenmp -frecursive`", for "ifort", `-openmp`.

- Memory usage is very important in parallel programming applications. To avoid unnecessary problems, consider:

  o The amount of memory being made available to the program's main thread (e.g. to make an unlimited amount of stack memory available to the main thread in Linux, use "ulimit -s unlimited").

- The amount of memory being made available to the OpenMP-spawned threads (the OMP_STACKSIZE environment is used for this purpose). In addition, depending on the compiler you are using, a special environment variable may also be available.

Note that the bash script provided with the VLIDORT package addresses these issues when running in Linux.

Before attempting to set up one's own driver program, consider the following:

- Carefully observe the use of the following OpenMP subroutines
  - OMP_SET_NUM_THREADS
  - OMP_GET_NUM_THREADS
  - OMP_GET_THREAD_NUM
- Carefully observe variables which are shared among the OpenMP threads using the SHARED attribute and those which are private to each thread (in these programs, the variables passed in are considered PRIVATE by default unless otherwise specified).

6.3 BRDF Supplement

Here, the bidirectional reflectance distribution function (BRDF) supplement is described. The BRDF supplement is a separate system of VLIDORT-based software that has the purpose of providing total BRDF inputs for the main VLIDORT programs. In other words, we wish to fill up the BRDF inputs in Tables C2 and G2 in sections 6.1.1.3 and 6.1.1.7, respectively. We note that the supplement also has the observational geometry facility like VLIDORT itself.

Section 6.3.1 has an overview of BRDF construction and we discuss the available options. A sample calling sequence for the supplement is given in section 6.3.2. The supplement inputs and outputs are listed in the tables of section 6.3.3. Descriptions of the ocean and land kernels are given in sections 6.3.4-6.3.6, and this is followed by some comments on direct-bounce BRDF treatment and surface emission in sections 6.3.7 and 6.3.8. Lastly, information regarding the calculation and usage of white-sky or black-sky albedos is given in section 6.3.9.

6.3.1 BRDFs as a sum of kernel functions

A scalar 3-kernel bidirectional reflectance distribution function (BRDF) scheme was implemented in LIDORT [*Spurr*, 2004]; a similar scheme is implemented in VLIDORT. Here we confine ourselves to the scalar description. The BRDF $\rho_{total}(\mu, \mu', \phi - \phi')$ is specified as a linear combination of (up to) three semi-empirical kernel functions:

$$\rho_{total}(\mu, \mu', \phi - \phi') = \sum_{k=1}^{3} R_k \rho_k(\mu, \mu', \phi - \phi'; \mathbf{b}_k) . \qquad (6.3.1)$$

Here, (θ, ϕ) indicates the pair of incident polar and azimuth angles, with the prime indicating the reflected angles. The R_k are linear combination coefficients or "kernel amplitudes", while the kernels $\rho_k(\theta, \theta', \phi - \phi'; \mathbf{b}_k)$ are derived from semi-empirical models of surface reflection for a variety of surfaces. For each kernel, the geometrical dependence is known, but the kernel function depends on the values taken by a vector $\mathbf{b}_k$ of pre-specified parameters.

A well-known example is the single-kernel Cox-Munk BRDF for glitter reflectance from the ocean [*Cox and Munk*, 1954a, 1954b]; the kernel is a combination of a Gaussian probability distribution function for the square of the wave facet slope (a quantity depending on wind-speed W), and a Fresnel reflection function (depending on the air-water relative refractive index m_{rel}). In this case, vector $\mathbf{b}_k$ has two elements: $\mathbf{b}_k = \{W, m_{rel}\}$. For a Lambertian surface, there is one kernel: $\rho_1 \equiv 1$ for all angles, and coefficient R_1 is just the Lambertian albedo.

In order to develop solutions in terms of a Fourier azimuth series, Fourier components of the total BRDF are calculated through:

$$\rho_k^m(\mu, \mu'; \mathbf{b}_k) = \frac{1}{2\pi} \int_0^{2\pi} \rho_k(\mu, \mu', \phi; \mathbf{b}_k) \cos m\phi \, d\phi \, . \tag{6.3.2}$$

This integration over the azimuth angle from 0 to 2π is done by double numerical quadrature over the ranges $[0, \pi]$ and $[-\pi, 0]$; the number of BRDF azimuth quadrature abscissa N_{BRDF} is set to 50 to obtain a numerical accuracy of 10^{-4} for all kernels considered in [*Spurr*, 2004].

Linearization of this BRDF scheme was reported in [*Spurr*, 2004], and a mechanism developed for the generation of surface property weighting functions with respect to the kernel amplitudes R_k and also to elements of the non-linear kernel parameters $\mathbf{b}_k$. It was shown that the entire discrete ordinate solution is differentiable with respect to these surface properties, once we know the following kernel derivatives:

$$\frac{\partial \rho_{total}(\theta, \alpha, \phi)}{\partial b_{p,k}} = \frac{\partial \rho_k(\theta, \alpha, \phi; \mathbf{b}_k)}{\partial b_{p,k}} \tag{6.3.3}$$

$$\frac{\partial \rho_{total}(\theta, \alpha, \phi)}{\partial R_k} = \rho_k(\theta, \alpha, \phi; \mathbf{b}_k) \tag{6.3.4}$$

The amplitude derivative is trivial. The parameter derivative (6.3.3) depends on the empirical formulation of the kernel in question, but all kernels in the LIDORT and VLIDORT BRDF schemes are analytically differentiable with respect to their parameter dependencies.

The choice of a Lambertian surface is a special case, controlled by a single flag. If this flag is set it is only necessary to specify the Lambertian albedo R (between or equal to one of the limit values 0.0 and 1.0). If the surface weighting function option is also set, then VLIDORT will return the Lambertian weighting function $K_R \equiv \partial I / \partial R$.

The VLIDORT BRDF supplement has 15 possible kernel functions, and these are listed in Table 6.3.1 along with the number of non-linear parameters; the user can choose up to three from this list. A full discussion of the first 9 scalar kernel types is given in [*Spurr*, 2004]. Kernels 12-14 are polarized land-surface reflectances based on code provided by François-Marie Bréon; these kernels have been revised for Version 2.7 in line with the non-polarized (scalar) kernels found in the LIDORT supplement. Kernel 15 is based on new water-leaving and ocean reflectance parameterizations provided in the context of the 6S model (Andrew Sayer, private communication).

Table 6.3.1 The BRDF kernel functions for VLIDORT

Index	Name	Size b_k	Reference	Scalar/Vector
1	Lambertian	0		Scalar
2	Ross thin	0	*Wanner et al.*, 1995	Scalar
3	Ross thick	0	*Wanner et al.*, 1995	Scalar
4	Li sparse	2	*Wanner et al.*, 1995	Scalar
5	Li dense	2	*Wanner et al.*, 1995	Scalar
6	Hapke	3	*Hapke*, 1993	Scalar
7	Roujean	0	*Wanner et al.*, 1995	Scalar
8	Rahman	3	*Rahman et al.*, 1993	Scalar
9	Cox-Munk	2	*Cox/Munk,* 1954	Scalar
10	Giss Cox-Munk	2	*Mishchenko/Travis* 1997	Vector
11	Giss Cox-Munk Cri	2	[*V. Natraj*, 2010, personal communication]	Vector
12	BPDF Soil	1	[*Maignan et al.*, 2009]	Vector
13	BPDF Vegetation	1	[*Maignan et al.*, 2009]	Vector
14	BPDF NDVI	3	[*Maignan et al.*, 2009]	Vector
15	New Cox-Munk	3	[*A. Sayer*, 2014]	Scalar

<u>*Remark*</u>. In VLIDORT, the BRDF is a 4 x 4 matrix linking incident and reflected Stokes 4-vectors. Most kernels are scalar, so that in VLIDORT, we set the {1,1} element of a 4 x 4 vector kernel ρ_k equal to the corresponding scalar kernel function ρ_k and all other elements are zero.

<u>*Remark*</u>. The above description is based on the use of 3 BRDF kernels. However, the kernel-BRDF programming in VLIDORT was set up in a general manner, and it is quite possible to use more than 3 kernels if desired. Although the Version 2.7 package is limited to 3 kernels through the setting of the parameter MAX_BRDF_KERNELS = 3 in the VLIDORT_PARS.f90 module, is only necessary to set this parameter equal to *n* if an *n*-kernel BRDF admixture is desired.

6.3.2 Example calling sequence

For an intensity calculation with a BRDF surface, the BRDF inputs required by VLIDORT_MASTER are those specified in section 6.1.1.3 Table C2 - namely, the direct-bounce BRDF for all solar incident and reflected line-of-sight directions, plus the four sets of Fourier components for the multiple scatter calculation. For a surface property Jacobian calculation (using the VLIDORT_LCS_MASTER or the VLIDORT_LPS_MASTER subroutines), VLIDORT also requires the linearized BRDF inputs in section 6.1.1.7 Table G2.

The test subdirectories "vlidort_s_test" and "vlidort_v_test" have one example of a calling environment for generating the Fourier components for BRDFs and their derivatives with respect to a number of surface properties. For a calculation of BRDF inputs alone (i.e. no linearizations), the calling program sequence is:

```
!  Obtain control variables for the vector BRDF input structure from the BRDF
!  input configuration file
```

```
    call VBRDF_INPUTMASTER ( &
        'VBRDF_ReadInput.cfg', & ! Input
        VBRDF_Sup_In,          & ! Outputs
        VBRDF_Sup_InputStatus )  ! Outputs


!   Call the vector BRDF supplement master
    call VBRDF_MAINMASTER ( &
        DO_DEBUG_RESTORATION, & ! Inputs
        BS_NMOMENTS_INPUT,    & ! Inputs
        VBRDF_Sup_In,         & ! Inputs
        VBRDF_Sup_Out,          ! Outputs
        VBRDF_Sup_OutStatus )   ! Outputs


!   Finish
    write BRDF Fourier component to file
```

The first subroutine (VBRDF_INPUTMASTER) reads inputs from a BRDF configuration file. These include specifications of the numbers and values of angles (solar and viewing angle zeniths, relative azimuths), the number of discrete ordinates, and the BRDF kernel choices. Angular and control inputs for the BRDFs must match equivalent inputs for VLIDORT before a VLIDORT radiance calculation with supplement-computed BRDF inputs is performed. The BRDF input read routine VBRDF_INPUTMASTER is of course optional - it is perfectly possible to set these inputs in another manner inside the calling environment itself.

Table A in the next section describes the kernel inputs required for a basic BRDF calculation. One can choose up to 3 kernels (see remark at the end of section 6.3.1 above for more on this), and for each kernel, one must specify the amplitude factors that go into the final linear-weighted combination of kernels that make up the total, and any non-linear parameters (such as wind speed for the glitter kernel) that characterize the kernels. Some kernels (e.g. the Ross-type kernels) are purely geometrical (no characterizing parameters). Also, an isotropic (Lambertian) kernel is allowed. The module file `vbrdf_sup_kernels.f90` contains a series of kernel subroutines (one for each of the entries in Table 6.3) delivering BRDFs for given incident and reflected angles. *With the use of the BRDF supplement, kernel input is not required for main VLIDORT calculations*.

The main subroutine (VBRDF_MAINMASTER) then carries out 3 tasks: (i) for the given choice of BRDF kernels, the kernel BRDFs themselves are created for all angles and streams; (ii) Fourier components of the BRDF kernels are generated by integrating over azimuth from 0 to 2π with a double Gaussian quadrature scheme; (iii) the total BRDF Fourier components are then created by a weighted combination of kernel components. The output from this subroutine is then written to file for subsequent use in VLIDORT itself; it is also possible to combine the BRDF supplement with the main VLIDORT call inside one environment, as has been done in 2p7_brdfplus_tester.f90 and V2p7_brdfplus_tester.f90.

For a calculation with surface property weighting functions, additional BRDF inputs are required. These are listed in Table C in the next section. One can obtain Jacobians with respect to the kernel amplitude factors and/or the non-linear characterizing parameters such as wind speed in the Cox-Munk glitter BRDF. Now, we use the file-read subroutine VBRDF_LIN_INPUTMASTER for all kernel inputs (regular and linearized), and the user

environment will then call the subroutine VBRDF_LIN_MAINMASTER which will deliver the total BRDF Fourier components for all the required geometrical configurations, as well as the linearizations of these total BRDF Fourier components with respect to a number of BRDF properties.

The total number of surface weighting functions (N_SURFACE_WFS) encompasses both the amplitude factor and the non-linear characterizing parameter Jacobians. The Jacobian property is ordered by kernels, with the amplitude factor followed by the non-linear parameters for each kernel in succession. For example, if we have a 3-kernel BRDF comprising a combination of Lambertian, Ross-thin, Li-Sparse in that order, then we can define 5 possible surface weighting functions: (1) amplitude for the Lambertian albedo (kernel #1), (2) amplitude for the Ross-thin (kernel #2), (3) amplitude for the Li-sparse (kernel #3), (4) non-linear parameter #1 for the Li-sparse kernel, and (5) non-linear parameter #2 for the Li-sparse kernel.

Note. This kernel bookkeeping applies only to the BRDF supplement. The main VLIDORT calculation has no knowledge of individual kernels or the order or type of surface property Jacobians. VLIDORT calculations only deal with the total BRDFs and their derivatives with respect to a set number of surface properties.

Note. The BRDF supplement is not required for a pure Lambertian surface calculation in VLIDORT; it is only necessary then to set the flag DO_LAMBERTIAN_ALBEDO and specify the albedo itself (LAMBERTIAN_ALBEDO in section 6.1.1.1 Table A7). Lambertian albedo weighting functions do not require any additional input information.

6.3.3 BRDF inputs and outputs

This section contains tables outlining the BRDF supplement input and output type structures (section 6.3.3.1) and tables of corresponding file-read character strings found in the input configuration file VBRDF_ReadInput.cfg (section 6.3.3.2).

Notes for Version 2.7.

(1) Following extensive user feedback on the use of BRDFs based on the MODIS 3-kernel combinations, a number of changes were made to the BRDF supplement software. In particular there is now an option to output the total white- and black sky albedos appropriate to the choice of kernels, and further options to scale the entire BRDF output with an external value of the white-sky albedo (WSA) or the black-sky albedo (BSA). Since the BSA in sun-angle dependent, only one value must be used for BSA scaling. There is also a flag for outputting the WSA and BSA values (these are useful diagnostics).

(2) As noted above, the land-surface kernels have been revised for Version 2.7 of VLIDORT. Formerly, only the BPDF "NDVI" kernel was present, but now the BPDF "Vegetation" and "Soil" kernels have been added as recommended by [*Maignan et al*., 2009]. All three kernels are based on Fresnel reflection. For each of these kernels, the relative refractive index is the first of the non-linear kernel parameters. Only the "NDVI" kernel has two additional parameters - these are the actual NDVI value itself, and an overall scaling factor for the kernel.

(3) Kernel 15 (New CM) was developed for Version 2.7, and is based on a Cox-Munk reflectance that includes a "whitecap" correction and a surface-leaving term. The details are found below in section 6.3.4.2.

(4) The use of more than 3 kernels is permitted (see remark above)

(5) For BRDF calculations using kernel 15, a specification of wavelength is needed (see table A below). This BRDF wavelength must be the same as the wavelength at which the atmospheric optical properties were prepared for the main VLIDORT calculation (see Table A7). An additional check on these wavelengths has been added.

6.3.3.1. Input and output type structures

Table A: Type Structure VBRDF_Sup_Inputs

Name	Kind/Intent	Description
DO_USER_STREAMS	Logical (I)	If set, there will be output at a number of off-quadrature zenith angles specified by user. This is the normal case.
DO_BRDF_SURFACE	Logical (I)	If set, calculations for more complex surface BRDF kernels will be done.
DO_SURFACE_EMISSION	Logical (I)	If set, calculations of surface thermal emission will be done.
DO_SOLAR_SOURCES	Logical (I)	Flag for solar beam source of light.
DO_USER_OBSGEOMS	Logical (I)	If set, supplement will compute BRDF quantities at observational geometry triplets specified by the user for multiple geometries. Used in conjunction with input variables N_USER_OBSGEOMS and USER_OBSGEOMS.
NSTOKES	Integer (I)	Number of Stokes vector parameters for which calculations will be done.
NSTREAMS	Integer (I)	Number of quadrature values used in the azimuth integration of the BRDF kernels in order to get Fourier components of BRDF kernels. Recommended value 25 for most kernels, 50 for Cox-Munk.
NBEAMS	Integer (I)	Number solar beams. Must ≤ symbolic dimension MAXBEAMS.
BEAM_SZAS	Real*8 (I)	Solar zenith angles (degrees). Checked internally range [0,90).
N_USER_RELAZMS	Integer (I)	Number of user-defined relative azimuth angles. Must not be greater than symbolic dimension MAX_USER_RELAZMS.
USER_RELAZMS	Real*8 (I)	Array of user-defined elative azimuth angles (in degrees) for off-quadrature output. Ordering is not important. Must be between 0 and 180.
N_USER_STREAMS	Integer (I)	Number of user-defined viewing zenith angles. Must be not greater than symbolic dimension MAX_USER_STREAMS.
USER_ANGLES_INPUT	Real*8 (I)	Array of user-defined viewing zenith angles (in degrees) for off-quadrature output. Must be between 0 and 90 degrees.
N_USER_OBSGEOMS	Integer (I)	Number of user-defined observational geometry triplets. Must not be greater than the symbolic dimension MAX_USER_OBSGEOMS.

USER_OBSGEOMS (g,3)	Real*8 (I)	Array of *g* user-defined observational geometry triplets (in degrees) for off-quadrature output. It consists of the geometry triplets (solar zenith angle, viewing angle, relative azimuth angle) for which BRDF quantities are desired.
N_BRDF_KERNELS	Integer (I)	Number of BRDF kernels to be used (up to 3 allowed).
BRDF_NAMES	Character (I)	Names of BRDF kernels to be used (up to 3 allowed).
WHICH_BRDF(k)	Integer (I)	Index numbers for BRDF kernels to be used (see the file VLIDORT.PARS or for values and comments).
N_BRDF_PARAMETERS (k)	Integer (I)	For each kernel *k*, the number of non-linear parameters characterizing kernel shape. Non-zero only for Li-sparse, Li-dense, Hapke, Rahman and Cox-Munk kernels.
BRDF_PARAMETERS (k,b)	Real*8 (I)	For kernel *k*, and *b* = 1, N_BRDF_PARAMETERS(k), these are the BRDF parameters. E.g.. for Cox-Munk, BRDF_PARAMETERS(k,1) and BRDF_PARAMETERS(k,2) are Wind speed and refractive index respectively.
LAMBERTIAN_KERNEL_FLAG	Logical (I)	Flag to indicate surface is purely Lambertian so only Lambertian calculations are done internally.
BRDF_FACTORS	Real*8 (I)	Amplitude factor associated with a BRDF kernel.
NSTREAMS_BRDF	Integer (I)	Number of angles used in azimuthal integration during BRDF calculation.
DO_SHADOW_EFFECT	Logical (I)	If set, calculations for incorporating the Shadow effect for the sea-surface glitter reflectance BRDF model will be done. Recommended.
DO_DIRECTBOUNCE_ONLY	Logical (I)	If set, *only* the direct-bounce BRDF will be calculated (i.e. BRDF Fourier components are NOT calculated).
DO_WSABSA_OUTPUT	Logical (I)	If set, white-sky and black-sky surface albedo values will be output by the BRDF supplement.
DO_WSA_SCALING	Logical (I)	If set, BRDF calculations using white-sky surface albedo will be done.
DO_BSA_SCALING	Logical (I)	If set, BRDF calculations using black-sky surface albedo will be done.
WSA_VALUE	Real*8 (I)	White-sky surface albedo.
BSA_VALUE	Real*8 (I)	Black-sky surface albedo.
DO_NewCMGLINT	Logical (I)	If set, BRDF calculations using new Cox-Munk (NCM) ocean BRDF kernel will be done.
SALINITY	Real*8 (I)	Salinity (in ppt). Only for NCM
WAVELENGTH	Real*8 (I)	Current wavelength (in μm). Only for NCM. This is now checked against ATMOS_WAVELENGTH (Table A7).
WINDSPEED	Real*8 (I)	Wind speed (in m/s) (only for non-isotropic water leaving). Only for NCM
WINDDIR (b)	Real*8 (I)	Wind direction relative to the azimuthal position of the sun for each solar angle *b* (only for non-isotropic water leaving). Only for NCM
DO_GlintShadow	Logical (I)	If set, calculations accounting for shadowing of wave facets during sunglint will be done. Only for NCM
DO_FoamOption	Logical (I)	If set, calculations accounting for ocean foam will be done. Only for NCM
DO_FacetIsotropy	Logical (I)	If set, wave facets will be considered isotropic (no use of wind direction). Only for NCM
DO_GLITTER_MSRCORR	Logical (I)	If set, multiple reflectance correction for all GLITTER kernels will be done.
DO_GLITTER_MSRCORR_DBONLY	Logical (I)	If set, multiple reflectance correction for only the direct-bounce Glitter kernels will be done.
GLITTER_MSRCORR_ORDER	Integer (I)	Order of correction for multiple reflectance computations (

		= 0 (no correction), 1, 2, 3, etc…). Warning, using S > 0 can increase CPU time dramatically.
GLITTER_MSRCORR_NMUQUAD	Integer (I)	Number of angles used in zenith integration during multiple reflectance correction.
GLITTER_MSRCORR_NPHIQUAD	Integer (I)	Number of angles used in azimuthal integration during multiple reflectance correction.

Table B1: Type structure VBRDF_Sup_Outputs

Name	*Kind/Intent*	*Description*
DBOUNCE_BRDFUNC (S,a,b,s)	Real*8 (O)	Direct-bounce BRDF for Stokes vector component *S*, incident solar angle *s*, reflected line-of-sight angle *a*, and relative azimuth *b*.
BRDF_F_0 (M,S,k,s)	Real*8 (O)	Fourier components *M* of total BRDF for Stokes vector component *S*, incident solar angle *s* and reflected discrete ordinate *k*.
BRDF_F (M,S,k,j)	Real*8 (O)	Fourier components *M* of total BRDF for Stokes vector component *S*, incident discrete ordinate *j* and reflected discrete ordinate *k*.
USER_BRDF_F_0 (M,S,a,s)	Real*8 (O)	Fourier components *M* of total BRDF for Stokes vector component *S*, incident solar angle *s* and reflected line-of-sight zenith angle *a*.
USER_BRDF_F (M,S,a,j)	Real*8 (O)	Fourier components *M* of total BRDF for Stokes vector component *S*, incident discrete ordinate *j* and reflected line-of-sight zenith angle *a*.
EMISSIVITY (S,k)	Real*8 (O)	Surface emissivity for Stokes vector component *S* and emitted discrete ordinate *k*.
USER_EMISSIVITY (S,a)	Real*8 (O)	Surface emissivity for Stokes vector component *S* and emitted line-of-sight zenith angle *a*.
WSA_CALCULATED	Real*8 (O)	Total White-sky surface albedo.
BSA_CALCULATED	Real*8 (O)	Total Black-sky surface albedo (first SZA only).
WSA_KERNELS	Real*8 (O)	Individual kernel White-sky surface albedos.
BSA_KERNELS	Real*8 (O)	Individual kernel Black-sky surface albedos (first SZA).

Table B2: Type Structure VBRDF_Input_Exception_Handling

Name	*Kind/Intent*	*Description*
STATUS_INPUTREAD	Integer (O)	Overall status of input read.
NINPUTMESSAGES	Integer (O)	Number of input read error messages.
INPUTMESSAGES	Character (O)	Array of input-read error messages.
INPUTACTIONS	Character (O)	Array of input-read actions to take.

Table B3: Type Structure VBRDF_Output_Exception_Handling

Name	*Kind/Intent*	*Description*
STATUS_OUTPUT	Integer (O)	Overall status of output.
NOUTPUTMESSAGES	Integer (O)	Number of output error messages.
OUTPUTMESSAGES	Character (O)	Array of output error messages.

Table C: Type Structure VBRDF_LinSup_Inputs

Name	*Kind/Intent*	*Description*
DO_KERNEL_FACTOR_WFS (k)	Logical (I)	Flags for weighting functions w.r.t. linear combination coefficient *k* in BRDF kernel sum.
DO_KERNEL_PARAMS_WFS (k,b)	Logical (I)	Flags for weighting functions for (nonlinear) parameter *b* in BRDF kernel *k*.

129

DO_KPARAMS_DERIVS (k)	Logical (I)	If set for a given BRDF kernel *k*, the chosen weighting functions for that BRDF kernel will be done.
N_SURFACE_WFS	Integer (I)	Sum of the following two entries. Should be set equal to N_SURFACE_WFS in linearization control Type structure (Table A12). Should not exceed dimension MAX_SURFACEWFS.
N_KERNEL_FACTOR_WFS	Integer (I)	Number of weighting functions w.r.t. linear combination coefficients in BRDF kernel sum
N_KERNEL_PARAMS_WFS	Integer (I)	Number of weighting functions for (nonlinear) BRDF parameters.
DO_WSAVALUE_WF	Logical (I)	If set, the white-sky albedo weighting function will be done.
DO_BSAVALUE_WF	Logical (I)	If set, the black-sky albedo weighting function will be done.
DO_WINDSPEED_WF	Logical (I)	If set, the wind speed weighting function will be done. May be used when using the new Cox-Munk ocean kernel.

Table D: Type structure VBRDF_LinSup_Outputs

Name	*Kind/Intent*	*Description*
LS_DBOUNCE_BRDFUNC (q,S,a,b,s)	Real*8 (O)	Linearized direct-bounce BRDF for Stokes vector component *S*, incident solar angle *s*, reflected line-of-sight angle *a*, and relative azimuth *b*, w.r.t. surface property *q*.
LS_BRDF_F_0 (q,M,S,k,s)	Real*8 (O)	Linearized Fourier components *M* of total BRDF for Stokes vector component *S*, incident solar angle *s* and reflected discrete ordinate *k*, w.r.t. surface property *q*.
LS_BRDF_F (q,M,S,k,j)	Real*8 (O)	Linearized Fourier components *M* of total BRDF for Stokes vector component *S*, incident discrete ordinate *j* and reflected discrete ordinate *k*, w.r.t. surface property *q*.
LS_USER_BRDF_F_0 (q,M,S,a,s)	Real*8 (O)	Linearized Fourier components *M* of total BRDF for Stokes vector component *S*, incident solar angle *s* and reflected line-of-sight zenith angle *a*, w.r.t. surface property *q*.
LS_USER_BRDF_F (q,M,S,a,j)	Real*8 (O)	Linearized Fourier components *M* of total BRDF for Stokes vector component *S*, incident discrete ordinate *j* and reflected line-of-sight zenith angle *a*, w.r.t. surface property *q*.
LS_EMISSIVITY (q,S,k)	Real*8 (O)	Linearized surface emissivity for Stokes vector component *S* and emitted discrete ordinate *k*, w.r.t. surface property *q*.
LS_USER_EMISSIVITY (q,S,a)	Real*8 (O)	Linearized surface emissivity for Stokes vector component *S* and emitted line-of-sight zenith angle *a*, w.r.t. surface property *q*.

6.3.3.2 BRDF configuration file character strings

Table E1: File-read Character strings for *some* variables in BRDF Supplement Table A

Name	Kind	Character string in Configuration file
DO_SOLAR_SOURCES	Logical	Use solar sources?
DO_USER_STREAMS	Logical	Use user-defined viewing zenith angles?
DO_BRDF_SURFACE	Logical	Do BRDF surface?
DO_NewCMGLINT	Logical	Do NewCM Ocean BRDF reflectance?
DO_SURFACE_EMISSION	Logical	Do surface emission?
NSTOKES	Integer	Number of Stokes vector components
NSTREAMS	Integer	Number of half-space streams
NBEAMS	Integer	Number of solar zenith angles
BEAM_SZAS	Real*8	Solar zenith angles (degrees)
N_USER_RELAZMS	Integer	Number of user-defined relative azimuth angles
USER_RELAZMS	Real*8	User-defined relative azimuth angles (degrees)
N_USER_STREAMS	Integer	Number of user-defined viewing zenith angles
USER_ANGLES_INPUT	Real*8	User-defined viewing zenith angles (degrees)
DO_OBSERVATION_GEOMETRY	Logical	Do Observation Geometry?
N_USER_OBSGEOMS	Integer	Number of Observation Geometry inputs
USER_OBSGEOM_INPUT	Real*8	Observation Geometry inputs
DO_GlintShadow	Logical	Do NewCM glint shadowing?
DO_FoamOption	Logical	Do NewCM whitecap (foam) reflectance?
DO_FacetIsotropy	Logical	Do NewCM facet isotropy?
WAVELENGTH	Real*8	NewCM Wavelength [Microns]?
SALINITY	Real*8	NewCM Ocean water salinity [ppt]
WINDSPEED	Real*8	NewCM Windspeed in [m/s]
WINDDIR	Real*8	NewCM Wind directions (degrees) relative to sun positions
N_BRDF_KERNELS	Integer	Number of BRDF kernels
DO_WSABSA_OUTPUT	Logical	Do white-sky and black-sky albedo output?
DO_WSA_SCALING	Logical	Do white-sky albedo scaling?
DO_BSA_SCALING	Logical	Do black-sky albedo scaling?
WSA_VALUE	Real*8	White-sky albedo value
BSA_VALUE	Real*8	Black-sky albedo value
NSTREAMS_BRDF	Integer	Number of BRDF azimuth angles
DO_SHADOW_EFFECT	Logical	Do shadow effect for glitter kernels?
DO_DIRECTBOUNCE_ONLY	Logical	Do direct-bounce only (no multiple-scatter contributions to BRDF)?
DO_GLITTER_MSRCORR	Logical	Do multiple reflectance for All glitter kernels?
DO_GLITTER_MSRCORR_DBONLY	Logical	Do multiple reflectance for just the direct-bounce glitter kernels?
GLITTER_MSRCORR_ORDER	Integer	Multiple reflectance scattering order for glitter kernels
GLITTER_MSRCORR_NMUQUAD	Integer	Multiple reflectance scattering; Polar quadrature order
GLITTER_MSRCORR_NPHIQUAD	Integer	Multiple reflectance scattering; Azimuth quadrature order
DO_WSAVALUE_WF	Logical	Do white-sky albedo Jacobian?
DO_BSAVALUE_WF	Logical	Do black-sky albedo Jacobian?
DO_WINDSPEED_WF	Logical	Do wind-speed (NewCM) Jacobian?

Table E2: File-read Character strings for grouped basic kernel variables in
BRDF Supplement Table A

Name	Kind	Character string in Configuration file
BRDF_NAMES	Character*10	Kernel names, indices, amplitudes, # parameters, parameters
WHICH_BRDF	Integer	*These quantities are formatted together for each kernel using Format(A10,I2,F8.4,I2,3F12.6). See example below.*
BRDF_FACTORS	Real*8	
N_BRDF_PARAMETERS	Integer	
BRDF_PARAMETERS	Real*8	.

```
Example of BRDF inputs: configuration file settings for 3 BRDF kernels as
indicated:

BRDFSUP - Kernel names, indices, amplitudes, # parameters, parameters
Cox-Munk  9 0.1000 2    0.079800    1.779556     0.000000
Ross-thin 2 0.3000 0    0.000000    0.000000     0.000000
Li-dense  5 0.1000 2    2.000000    1.000000     0.000000
```

Note that the formatting was changed for Version 2.7 to allow more decimal places for the kernel amplitudes (formerly F6.2, now F8.4).

Special note regarding Cox-Munk type ocean BRDF kernels:

The Cox-Munk kernel uses $\sigma^2 = 0.003 + 0.00512*W$ where W is the wind speed in meters/second for the first parameter. For example, if W = 10, then $\sigma^2 = 0.054200$. In contrast, the Giss-Cox-Munk kernel uses $0.5*\sigma^2$ for the first parameter (half the value!). Thus, for this value of W, the Giss-Cox-Munk kernel would a value of $0.5*\sigma^2 = 0.027100$ for its first parameter.

Also, the Cox-Munk kernel uses the *square* of the refractive index for the second parameter. For example, if the refractive index is 1.334, then the second parameter would be $1.334*1.334 = 1.779556$. In contrast, the Giss-Cox-Munk kernel uses just the refractive index itself for the second parameter. Thus, the Giss-Cox-Munk kernel would a value of 1.334 for its second parameter.

Table F: File-read Character strings for linearized kernel variables in
BRDF Supplement Table C

Name	Kind	Character string in Configuration file
DO_KERNEL_FACTOR_WFS	Logical	Kernels, indices, # pars, Factor Jacobian flag, Par Jacobian flags
DO_KERNEL_PARAMS_WFS	Logical	*These quantities are formatted together for each kernel using Format (A10,I3,I2,4L). See example below.*

```
Example of linearized BRDF inputs: configuration file settings for 3 BRDF
kernels as indicated:

BRDFSUP - Kernels, indices, # pars, Factor Jacobian flag, Par Jacobian flags
Cox-Munk  9 2  T    T T F
Ross-thin 2 0  T    F F F
Li-dense  5 2  T    T T F
```

6.3.4 Ocean glitter kernel functions

6.3.4.1 Original Cox-Munk Glint reflectance

We now turn to descriptions of the individual BRDF kernels. The ocean glitter kernels are described here and the land kernels in the following section. For glitter, we use the well-known geometric-optics regime for a single rough-surface redistribution of incident light, in which the reflection function is governed by Fresnel reflectance and takes the form [*Jin et al.*, 2006]:

$$\rho_{CM}(\mu,\mu',\phi-\phi',m,\sigma^2) = r(\theta_r,m).\frac{1}{\mu\mu'|\gamma_r|^4}.P(\gamma_r,\sigma^2).D(\mu,\mu',\sigma^2) \qquad (6.3.5)$$

Here, σ^2 is the slope-squared variance (also known as the MSS or mean slope square) of the Gaussian probability distribution function $P(\gamma,\sigma^2)$ which has argument γ (the polar direction of the reflected beam); $r(\theta,m)$ is the Fresnel reflection for incident angle θ and relative refractive index m, and $D(\mu,\mu',\sigma^2)$ is a correction for shadowing. The two non-linear parameters are σ^2 and m. We have the usual Cox-Munk empirical relation [*Cox and Munk*, 1954a]:

$$\sigma^2 = 0.003 + 0.00512W \qquad (6.3.6)$$

in terms of the wind speed W in m/s. A typical value for m is 1.33. The MSS Gaussian is:

$$P(\alpha,\sigma^2) = \frac{1}{\pi\sigma^2}\exp\left[-\frac{\alpha^2}{\sigma^2(1-\alpha^2)}\right]; \qquad (6.3.7)$$

The shadow function of [*Sancer*, 1969] is widely used, and is given by:

$$D(\alpha,\beta,\sigma^2) = \frac{1}{1+\Lambda(\alpha,\sigma^2)+\Lambda(\beta,\sigma^2)}; \qquad (6.3.8a)$$

$$\Lambda(\alpha,\sigma^2) = \frac{1}{2}\left(\left[\frac{(1-\alpha^2)}{\pi}\right]^{1/2}\frac{\sigma}{\alpha}\exp\left[-\frac{\alpha^2}{\sigma^2(1-\alpha^2)}\right]-erfc\left[\frac{\alpha}{\sigma\sqrt{(1-\alpha^2)}}\right]\right). \qquad (6.3.8b)$$

Both the Gaussian function and the shadow correction are *fully differentiable* with respect to the defining parameters σ^2 and m. Indeed, we have:

$$\frac{\partial P(\alpha,\sigma^2)}{\partial\sigma^2} = \frac{P(\alpha,\sigma^2)}{\sigma^4}\left[\frac{\alpha^2}{(1-\alpha^2)}-\sigma^2\right]. \qquad (6.3.9)$$

The shadow function can be differentiated in a straightforward manner since the derivative of the error function is another Gaussian. The complete kernel derivative with respect to σ^2 is then:

$$\frac{\partial\rho_{CM}(\mu,\mu',\phi-\phi',m,\sigma^2)}{\partial\sigma^2} = r(\theta_r,m).\frac{1}{\mu\mu'|\gamma_r|^4}.$$

$$\left[\frac{\partial P(\gamma_r,\sigma^2)}{\partial\sigma^2}.D(\mu,\mu',\sigma^2)+P(\gamma_r,\sigma^2).\frac{\partial D(\mu,\mu',\sigma^2)}{\partial\sigma^2}\right] \qquad (6.3.10)$$

VLIDORT has a vector kernel function for sea-surface glitter reflectance, based on the description in [*Mischenko and Travis*, 1997]; this "GISS-COXMUNK" kernel has also been completely linearized with respect to the MSS [*Natraj and Spurr*, 2007].

With this formulation of linearized input for the glitter kernel, LIDORT and VLIDORT are thus able to deliver *analytic weighting functions with respect to the wind speed*. This is important for remote sensing instruments with a glitter viewing mode; an example is the Orbiting Carbon Observatory [*Crisp et al.*, 2004]. Note that it is possible to use other parameterizations of the MSS [*Zhao and Toba*, 2003] in this glitter formalism.

The above formulation is for a single Fresnel reflectance by wave facets. In reality, glitter is the result of many reflectances. Writing $R(\Omega,\Omega_0)$ for the glitter BRDF for incident and reflected directions $\Omega_0 = \{\mu_0, \phi_0\}$ and $\Omega = \{\mu, \phi\}$, respectively, we have for one extra order of scattering:

The above formulation is for a single Fresnel reflectance by wave facets. In reality, glitter is the result of many reflectances. It is possible to incorporate a correction for multiple reflectances for this glitter contribution, both for the direct-bounce term and the Fourier components. Given that the glitter maximum is typically dominated by direct reflectance of the solar beam, we confine the multiple-reflectance option to this term. Thus, we consider only one extra order of scattering. Writing $R(\Omega,\Omega_0)$ for the glitter BRDF for incident and reflected directions $\Omega_0 = \{\mu_0, \phi_0\}$ and $\Omega = \{\mu, \phi\}$, respectively, we have for one extra order of scattering:

$$R(\Omega,\Omega_0) = R_0(\Omega,\Omega_0) + R_1(\Omega,\Omega_0); \tag{6.3.11a}$$

$$R_1(\Omega,\Omega_0) = \int_0^{2\pi}\int_0^1 R_0(\Omega,\Omega'')R_0(\Omega'',\Omega_0)d\mu''d\phi''. \tag{6.3.11b}$$

The azimuthal integration for the additional correction is done by double Gaussian quadrature over the intervals $[-\pi,0]$ and $[0,\pi]$. The polar stream integration in (6.3.11b) is also done by Gauss-Legendre quadrature. Both these equations are differentiable with respect to the slope-squared parameter, so that Jacobians for the wind speed can also be determined for this case. It is possible to calculate higher order contributions, so that $R(\Omega,\Omega_0) = \sum_s R_s(\Omega,\Omega_0)$ for scattering order s, but this is a time-consuming process. For more details, see [*Jin et al.*, 2006].

In general, one extra order of scattering enhances the BRDFs by 5-15%, depending on the geometrical configuration; second and higher order contributions are generally at the 1% level or less. We have found that the neglect of multiple glitter reflectances can lead to errors of 1-3% in the upwelling intensity at the top of the atmosphere, the higher figures being for larger solar zenith angles.

In Versions 2.4 and 2.5 of VLIDORT, this multiple-reflected glitter was confined to the direct beam calculation for one extra order of scattering. In version 2.6 of VLIDORT, we have developed a facility for computing multiply-reflected glitter for any order of scattering, applicable to the direct beam BRDF computation as well as the four Fourier component terms.

6.3.4.2 Alternative Cox-Munk Glint reflectance

The existing rough-surface ocean-glint reflectance option in VLIDORT is based on an older implementation of the well-known Cox-Munk Gaussian distribution of wave facets - this does not

include any treatment of whitecaps (foam), and there is no directionality in the wind. Further, the CM implementation is based on a fixed real-valued refractive index for water.

We have now added an alternative Cox-Munk implementation which addresses these issues. This new-CM option is based on the glint treatment in the 6S code [*Vermote et al.*, 1997; *Kotchenova et al.*, 2006], and includes an empirical whitecap contribution [*Koepke*, 1984]. The latter model also includes a more recent treatment of water-leaving radiance [*Morel and Gentili*, 2009] and the VLIDORT SLEAVE supplement has been updated according to the 6S treatment (see section 6.2.1 also). These new 6S-based options in the VLIDORT BRDF and SLEAVE supplements are designed to operate in tandem. Indeed, the total surface radiance in the 6S model is given by

$$I(\mu_0, \mu_1, \varphi_0 - \varphi_1) = (1 - R_{wc})S(\mu_0, \mu_1)$$
$$+ R_{wc} + (1 - R_L)\,\rho_{NCM}(\mu_0, \mu_1, \varphi_0 - \varphi_1), \qquad (6.3.12)$$

where the water-leaving term $S(\mu_0, \mu_1)$ does depend on the outgoing and incoming directions but is azimuth-independent, ρ_{NCM} is the Cox-Munk glint reflectance, and R_{wc} and R_L are the empirically-derived whitecap contributions (final and Lambertian).

For VLIDORT to possess this functionality for the ocean surface, the BRDF supplement must provide the second and third terms on the right-hand-side of (6.3.12), while the SLEAVE supplement supplies the first term. Note that both supplements require the same whitecap formulation. The derivation of the water-leaving term is done in the next section.

The Cox-Munk calculation for ρ_{NCM} depends on the wind speed, wind-direction and refractive index. The latter is a complex variable that depends on the salinity of the ocean, and we use the 6S formulation. For isotropic facet reflectance, the facet slope variance is given by $\sigma^2 = 0.00512 * W + 0.003$ independent of wind direction, where W is the wind speed in [m/s]. The anisotropic treatment assumes an (azimuthal) wind direction relative to the solar incident beam, and this follows the formulae developed by Cox and Munk in the 1950s. The shadowing formula is the same as that for the original Cox-Munk kernel.

For facet anisotropy, the wind-direction is dependent on the solar position, and it follows that any BRDF quantity will pick up additional dependence on the solar angle. It is then only possible to use this "New-CM" option with a single solar zenith angle (and hence a single wind-direction azimuth) - multiple solar beams are ruled out. See also the remark in section 6.3.8 below concerning the use of the black-sky albedo. Again, there is exception handling for this eventuality.

Owing to the rather specific nature of this "New-CM" option, we have kept it separate from the usual 3-kernel BRDF implementation and the albedo-scaling options noted in section 6.3.8. There is a general flag (Boolean variable) DO_NEWCMGLINT for choosing this option - this will invoke choices for 3 dedicated inputs which are the wind speed itself (WINDSPEED), the wind direction for one solar position (WINDDIR), the wavelength (WAVELENGTH) in [Microns] and the salinity in [ppt] (SALINITY). We have also introduced options for the whitecap term (DO_FOAMOPTION), the shadowing term (DO_GLINTSHADOW) and the isotropic facet-slope variance (DO_FACETISOTROPY). If the latter flag is set, then the usual Cox-Munk formula $\sigma^2 = 0.00512 * W + 0.003$ will apply.

In addition, we have linearized this "New-CM" glint reflectance with respect to the wind speed - this option is governed by the flag DO_WINDSPEED_WF. Finally, we note that this glint

reflectance is scalar only, so that the above considerations apply only to the (1,1) element of the reflectance matrix in the vector BRDF supplement for VLIDORT. Polarization of this contribution is currently undergoing investigation.

6.3.5 Scalar land-surface BRDF kernels(MODIS system)

VLIDORT has an implementation of a set of 5 semi-empirical MODIS-type kernels applicable to vegetation canopy [*Wanner et al*., 1995]; each such kernel must be used in a linear combination with a Lambertian kernel. Thus, for example, a Ross-thin BRDF _surface type_ requires a combination of a Ross-thin _kernel_ and a Lambertian kernel:

$$\rho_{total}(\theta,\alpha,\phi) = c_1\rho_{Rossthin}(\theta,\alpha,\phi) + c_2 \qquad (6.3.13a)$$

Linear factors c_1 and c_2 are interdependent, and are specified in terms of basic quantities of the vegetation canopy. The kernels divide into two groups: those based on *volume scattering* empirical models of light reflectance (Ross-thin, Ross-thick), and those based on *geometric-optics* modeling (Li-sparse, Li-dense, Roujean). See [*Wanner et al*., 1995] and [*Spurr*, 2004] for details of the kernel formulae.

In fact, it is standard practice in MODIS BRDF retrievals to use a combination of Lambertian, Ross-thick and Li-Sparse kernels, and this 3-kernel combination is common.

$$\rho_{total}(\theta,\alpha,\phi) = f_{iso} + f_{vol}\rho_{Rossthin}(\theta,\alpha,\phi) + f_{geo}\rho_{LiSparse}(\theta,\alpha,\phi) \qquad (6.3.13b)$$

VLIDORT also has implementations of two other semi-empirical kernels for vegetation cover; these are the Rahman [*Rahman et al*., 1993] and Hapke [*Hapke*, 1993] BRDF models. Both kernels have three nonlinear parameters, and both contain parameterizations of the backscatter hot-spot effect. Here is the Hapke formula:

$$\rho_{hapke}(\mu_i,\mu_j,\phi) = \frac{\omega}{8(\mu_i+\mu_j)}$$
$$\left\{\left(1+\frac{Bh}{h+\tan\alpha}\right)(2+\cos\Theta) + \frac{(1+2\mu_i)(1+2\mu_j)}{\left(1+2\mu_i\sqrt{1-\omega}\right)\left(1+2\mu_j\sqrt{1-\omega}\right)} - 1\right\} \qquad (6.3.14)$$

In this equation, the three nonlinear parameters are the single scattering albedo ω, the hotspot amplitude h and the empirical factor B; μ_i and μ_j are the directional cosines, and Θ is the scattering angle, with $\alpha = \frac{1}{2}\Theta$.

The important point to note here is that all these kernels are fully differentiable with respect to any of the non-linear parameters defining them. For details of the kernel derivatives, see [*Spurr*, 2004]. It is thus possible to generate analytic weighting functions for a wide range of surfaces in the models. Surface reflectance Jacobians have also been considered in other linearized RT models [*Hasekamp and Landgraf*, 2002; *Ustinov*, 2005].

These kernels were developed for scalar BRDFs – the (1, 1) component of the polarized surface reflectance matrix. All scalar BRDFs have been implemented as part of the VLIDORT package. Polarized BRDFs over land surfaces are harder to come by. Here we report briefly on some new semi-empirical formulae for BPDFs (Bidirectional Polarized Distribution Functions) [*Maignan et al.*, 2009]. These BPDF kernels were supplied by F.-M. Bréon, and permission has been granted

to use them in VLIDORT, provided the work is properly acknowledged using the above 2009 reference. See also section 3.2.4.

6.3.6 Polarized land surface BRDF kernels

In general, BPDFs are "spectrally neutral", and modeling using specular reflection has become the accepted way of generating these functions. An empirical model was developed in part from specular reflection and in part from an analysis of POLDER measurements [Nada and Bréon, 1999]. Recently, a great deal more BPDF information has been gleaned from data analysis of several years of measurements from the PARASOL instrument. Based on this analysis, the paper of [*Maignan et al.*, 2009] gives the following empirical formula for the BPDF:

$$\mathbf{Rp}_{NDVI}(\mu_i, \mu_j, \varphi_i - \varphi_j) = \frac{C\mathbf{Fp}(m, \theta_R)}{4(\mu_i + \mu_j)} \exp[-\tan \theta_R] \exp[-NDVI]. \tag{6.3.15}$$

Here, $\theta_R = \frac{1}{2}\theta_{SCAT}$, and vector Fresnel reflectance $\mathbf{Fp}(m, \theta_R)$ depends only on θ_R and refractive index m. The scattering angle θ_{SCAT} is determined in the usual manner from the incident and reflected directional cosines μ_i and μ_j, and the relative azimuth $\varphi_i - \varphi_j$. Calculation of $\mathbf{Fp}$ follows the specification given above for the Giss Cox-Munk BRDF. There are exponential attenuation terms, one of which depends on the NDVI Vegetation Index; in this formula, the scaling factor is C (nominally, this is set to 1.0).

There are three parameters characterizing the reflectance - the refractive index *m*, the Vegetation Index *NDVI*, and the scaling factor *C*. The NDVI varies from -1 to 1 and is defined as the ratio of the difference to the sum of two radiance measurements, one in the visible and one in the infrared. Linearization of the kernel with respect to refractive index *m* will require the partial derivatives $\partial \mathbf{Fp}(m, \theta_R)/\partial m$, which are easy to determine from the Fresnel formulation. Linearizations with respect to the parameters *NDVI* and *C* are easy to establish. Equation (6.3.15) is the BPDF "NDVI" kernel.

The other BPDF kernels are defined similarly. For the "Soil" type, we have

$$\mathbf{Rp}_{SOIL}(\mu_i, \mu_j, \varphi_i - \varphi_j) = \frac{\mathbf{Fp}(m, \theta_R)}{4\mu_i\mu_j}(1 - \sin \theta_R). \tag{6.3.16}$$

Here, $\theta_R = \frac{1}{2}\theta_{SCAT}$ as before, and the refractive index *m* is the only parameter.

For the BDPF "Vegetation" type, there is dependence on the leaf orientation probability $\sigma(\alpha)$ and the leaf facet projections, through use of a plagiophile distribution:

$$\mathbf{Rp}_{VEGN}(\mu_i, \mu_j, \varphi_i - \varphi_j) = \frac{\mathbf{Fp}(m, \theta_R)}{4\mu_i\mu_j}\frac{\sigma(\alpha)}{P}(1 - \sin \theta_R); \tag{6.3.17}$$

$$\cos \alpha = \frac{(\mu_i + \mu_j)}{2\cos \theta_R}; \qquad \sigma(\alpha) = \frac{16}{\pi}\cos^2 \alpha \sin \quad ;$$

$$P = \frac{\sum_{k=0}^3 a_k \mu_i^k}{\mu_i} + \frac{\sum_{k=0}^3 a_k \mu_j^k}{\mu_j}.$$

where the leaf projection *P* depends on the set of "plagiophile coefficients" $\{a_k\}$. Again, the refractive index is the only surviving kernel parameter to be considered for linearization.

6.3.7 The direct bounce correction for BRDFs

For RT calculations with BRDF surfaces, the radiation field has contributions from the direct-bounce BRDF and (for each Fourier term) the diffusely reflected BRDF components. One can compute the direct-bounce BRDF with a full set of BRDF kernels rather than use their truncated forms based on a (finite) Fourier series expansion. This is the "direct-bounce (DB) correction" in VLIDORT , and it is done *before* the diffuse field calculation (Fourier convergence of the whole field is discussed in section 3.3.6). The direct-bounce upwelling reflection of the solar beam (assuming plane-parallel attenuation) to optical depth τ may be written:

$$I_{REX}^{\uparrow}(\mu,\phi,\tau) = I_0 \rho_{DB}(\mu,\mu_0,\phi-\phi_0) \exp\left[\frac{-\tau_{atmos}}{\mu_0}\right] \exp\left[\frac{-(\tau_{atmos}-\tau)}{\mu}\right]. \tag{6.3.18}$$

For surface property Jacobians, we require computation of the derivatives of this DB correction with respect to the kernel amplitudes and parameters; this is straightforward based on the discussion in section 6.3.1. For atmospheric profile weighting functions, the solar beam and line-of-sight transmittances that form part of the DB correction in Eq. (6.3.18) need to be differentiated with respect to variables ξ_p varying in layer p.

When this DB correction is in force, the corresponding truncated Fourier-sum for a single reflectance should be omitted from the diffuse field calculations. As with the single scatter case, $I_{REX}^{\uparrow}(\mu,\phi,\tau)$ should be added to the total field just after calculation of the azimuth-independent Fourier term, and before the higher-order Fourier are computed and the total radiance field examined for convergence; once again, this will make the calculation faster and more accurate.

In the Version 2.7 BRDF supplement, the direct-bounce calculation is always performed automatically, regardless of whether it will be used in VLIDORT or not (the default is to use it); *there is no separate flag for this correction.*

6.3.8 Surface emission in the VLIDORT model

In addition to the surface reflection of diffuse and direct radiation, there is a surface emission source term, which will be present for all Fourier components for a bidirectional surface:

$$I_{n,e}^{-}{}_{m}(\tau_n,\mu) = \delta_{m,0}\kappa(\mu)B(T_g) \tag{6.3.19}$$

Here, the emissivity is given by Kirchhoff's law:

$$\kappa(\mu) = 1 - 2\int_0^1 \mu'\rho_0(\mu,\mu')d\mu' . \tag{6.3.20}$$

Here, $\rho_0(\mu,\mu')$ is the azimuth independent component of the total BRDF kernel Fourier expansion. For the Lambertian surface with albedo R_0, we have $\kappa(\mu) = 1 - R_0$ for all directional cosines.

Note that the emissivity Eq. (6.3.20) will have derivatives with respect to the surface kernel amplitudes R_k and the kernel parameters $\mathbf{b}_k$ in Eq. (6.3.1).

6.3.9 White-sky and Black-sky albedo scaling

There are now two options to normalize the multiple-kernel LIDORT and VLIDORT BRDFs according to a choice of spherical albedo - either the total spherical or *white-sky albedo* (WSA), or the directional *black-sky albedo* (BSA), the latter being dependent on the solar zenith angle. These options are designed to work alongside the MODIS-based system of semi-empirical kernel BRDF models which has already been implemented in the LIDORT and VLIDORT BRDF supplements - see Equation (6.3.13b) above. For review material on the MODIS-BRDF system, refer to [*Lucht and Roujean*, 2000; *Lucht et al.*, 2000].

We consider the scalar model in the description below; the treatment is similar for the vector model, where the albedos are defined only for the (1, 1) element of the 4x4 reflectance matrix.

Assuming the kernel BRDFs to be normalized to $(1/\pi)$, the two albedos are defined through:

$$A_{WSA} = 4 \int_0^1 \int_0^1 \mu\mu'\rho_0(\mu,\mu')d\mu d\mu' \; ; \; A_{BSA}(\mu_0) = 2 \int_0^1 \mu'\rho_0(\mu_0,\mu')d\mu' . \qquad (6.3.21)$$

Here μ_0 is the cosine of the SZA, and $\rho_0(\mu,\mu')$ is the $m = 0$ component of the BRDF expressed as a Fourier series in cosine azimuth:

$$\rho(\mu,\mu',\varphi - \varphi') = \sum_{m=0}(2 - \delta_{m0})\rho_m(\mu,\mu')\cos m(\varphi - \varphi'). \qquad (6.3.22)$$

The LIDORT BRDF supplement allows us to define a 3-kernel BRDF in terms of amplitude factors $R^{(k)}$ and individual kernels $\rho^{(k)}(\mu,\mu',\varphi - \varphi')$:

$$\rho(\mu,\mu',\varphi - \varphi') = \sum_{k=1}^3 R^{(k)}\rho^{(k)}(\mu,\mu',\varphi - \varphi'). \qquad (6.3.23)$$

Then the albedo-scaled BRDF is

$$\tilde{\rho}(\mu,\mu',\varphi - \varphi') = \frac{\tilde{A}}{A_{SA}}\sum_{k=1}^3 R^{(k)}\rho^{(k)}(\mu,\mu',\varphi - \varphi'). \qquad (6.3.24)$$

Here, $\tilde{A}$ is an <u>external</u> spherical albedo, and the <u>internal</u> spherical albedo is $A_{SA} = \sum_{k=1}^3 R^{(k)}A_{SA}^{(k)}$, where the kernel albedo $A_{SA}^{(k)}$ is either $A_{WSA}^{(k)} = 4 \int_0^1 \int_0^1 \mu\mu'\rho_0^{(k)}(\mu,\mu')d\mu d\mu'$ for the white-sky case, or $A_{BSA}^{(k)}(\mu_0) = 2 \int_0^1 \mu'\rho_0^{(k)}(\mu_0,\mu')d\mu'$ for the black-sky case. For a Lambertian kernel, $A_{SA}^{(k)}$ is just the Lambertian albedo.

In order to obtain values of $A_{SA}^{(k)}$, the half-space integrals are done by Gaussian quadrature using abscissa and weights $\{\mu_p, w_p\}$, $p = 1 \cdots N_p$. In other words:

$$A_{BSA}^{(k)}(\mu_0) = 2 \sum_{p=1}^{N_p} \mu_p w_p \rho_0^{(k)}(\mu_0, \mu_p); \; A_{WSA}^{(k)} = 4 \sum_{q=1}^{NP} \mu_q w_q \sum_{p=1}^{NP} \mu_p w_p \rho_0^{(k)}(\mu_q, \mu_p).$$

$$(6.3.25)$$

This quadrature $\{\mu_p, w_p\}$ is completely separate from the discrete ordinate quadrature that is used for LIDORT radiative transfer, and is only used in the BRDF supplement. The default value of N_p is currently 24. Eqs. (6.3.25) require computation of the Fourier components $\rho_0^{(k)}(\mu_0, \mu_p)$ (BSA case), or $\rho_0^{(k)}(\mu_q, \mu_p)$ (WSA).

There is a consistency check on the magnitude of the overall internally-calculated BRDF spherical albedo A_{SA}. The BRDF supplement ensures that A_{SA} is non-negative and lies

somewhere in the interval [0, 1]; if this condition is violated, the exception handling will return a fatal status when running the BRDF supplement.

Remark. The BRDF supplement not only generates BRDFs $\rho(\mu_0, \mu_1, \varphi_0 - \varphi_1)$ for incoming SZA cosines μ_0, outgoing line-of-sight cosines μ_1 and relative azimuth angles $\varphi_0 - \varphi_1$, but it also calculates Fourier-series components $\rho_m(\mu_0, \mu_k)$, $\rho_m(\mu_j, \mu_i)$, $\rho_m(\mu_j, \mu_1)$ and $\rho_m(\mu_0, \mu_1)$ as required for multiple-scattering (MS) of surface reflectance. Here, $\{\mu_j\}$, $j = 1 \cdots N_d$ are the N_d discrete ordinate polar streams to be used for the LIDORT MS calculations, and the Fourier index is $m = 0, 1 \cdots 2N_d - 1$. Albedo-scaling (if selected) applies to all these Fourier components. In the BSA case, albedo scaling is dependent on the SZA through its cosine μ_0, and it then follows that the _scaled_ components $\tilde{\rho}_m(\mu_j, \mu_i)$ and $\tilde{\rho}_m(\mu_j, \mu_1)$ will pick up dependence on SZA which the equivalent unscaled components did not possess. Because of this additional dependence, the BSA scaling can only be applied to all BRDF outputs for a single value of the SZA - no multiple SZA calculations are allowed. The BRDF code checks for this eventuality.

From a practical stand-point, there are 4 new inputs associated with these albedo scaling choices. Boolean flags DO_WSA_SCALING and DO_BSA_SCALING control the options - these two are mutually exclusive (this is checked). The scaling is done with user-supplied floating point variables WSA_VALUE or BSA_VALUE. The latter inputs are checked for the [0, 1] range. . See the entries in Tables A and E1.

Linearizations

The multi-kernel BRDFs have analytic partial derivatives with respect to the amplitude factors $R^{(k)}$ and also to parameters $b^{(k)}$ which are inherent to the kernels themselves (e.g. $b^{(k)}$ could be the wind speed for the Cox-Munk glint kernel). Application of the albedo scaling requires additional differentiation. Indeed, taking the derivative of Eq. (6.3.25) for the amplitude factor $R^{(k)}$ yields (we have dropped the geometrical variables for convenience):

$$\frac{\partial \tilde{\rho}}{\partial R^{(k)}} = \frac{\tilde{A}\rho^{(k)} - \tilde{\rho}A_{SA}^{(k)}}{A_{SA}}. \tag{6.3.26}$$

Now suppose that the unscaled kernel $\rho^{(k)}$ has derivative $\frac{\partial \rho^{(k)}}{\partial b^{(k)}}$ with respect to parameter $b^{(k)}$. Then differentiation of (6.3.24) yields:

$$\frac{\partial \tilde{\rho}}{\partial b^{(k)}} = \frac{\tilde{A}R^{(k)}\frac{\partial \rho^{(k)}}{\partial b^{(k)}} - \tilde{\rho}\frac{\partial A_{SA}^{(k)}}{\partial b^{(k)}}}{A_{SA}}. \tag{6.3.27}$$

This last result requires the following computations to be added to the BRDF supplement:

$$\frac{\partial A_{WSA}^{(k)}}{\partial b^{(k)}} = 4 \int_0^1 \int_0^1 \mu\mu' \frac{\partial \rho_0^{(k)}}{\partial b^{(k)}}(\mu, \mu') d\mu d\mu'; \quad \frac{\partial A_{BSA}^{(k)}}{\partial b^{(k)}}(\mu_0) = 2 \int_0^1 \mu' \frac{\partial \rho_0^{(k)}}{\partial b^{(k)}} d\mu'. \tag{6.3.28}$$

In line with the additional computations in equations (6.3.25) using quadrature $\{\mu_p, w_p\}$, the derivatives in Eqs. (6.3.28) are computed via:

$$\frac{\partial A_{BSA}^{(k)}}{\partial b^{(k)}}(\mu_0) = 2 \sum_{p=1}^{N_p} \mu_p w_p \frac{\partial \rho_0^{(k)}}{\partial b^{(k)}}(\mu_0, \mu_p); \quad \frac{\partial A_{WSA}^{(k)}}{\partial b^{(k)}} = 4 \sum_{q=1}^{NP} \mu_q w_q \sum_{p=1}^{NP} \mu_p w_p \frac{\partial \rho_0^{(k)}}{\partial b^{(k)}}(\mu_q, \mu_p). \tag{6.3.29}$$

It is possible to generate derivatives of the scaled BRDFs with respect to the user-supplied external spherical albedo $\tilde{A}$. These options are controlled by flags DO_WSASCALING_WF and DO_BSASCALING_WF, which are again mutually exclusive. If either of these flags is set, the linearized BRDF supplement will deliver a Jacobian with respect to the WSA or BSA. This is an option that is treated separately from the other kernel-property or kernel-amplitude linearizations. The WSA/BSA-derivative is easy to write down: from (6.3.24), we have

$$\frac{\partial \tilde{\rho}}{\partial \tilde{A}} = \frac{\tilde{\rho}}{\tilde{A}}. \tag{6.3.30}$$

This completes the additional work on the BRDF supplement.

These notes are applicable for the scalar BRDFs (LIDORT); for the vector BRDF supplement that goes with VLIDORT, the BRDF $\rho(\mu, \mu', \varphi - \varphi')$ is the 4x4 reflection matrix, and the spherical albedo is then only derived from the (1, 1) element of the (Fourier-zero) component of the BRDF matrix. All other elements are scaled by the same amount.

There is now an option to output the calculated WSA and BSA values (before any scaling is applied), both for the total albedos, and for the individual kernel WSAs and BSAs. These are useful diagnostics.

6.4 SLEAVE Supplement

Here, the surface-leaving (SL) supplement is described. The SL supplement is a separate system of VLIDORT-based software that generates a source of radiance at the lower boundary, for instance, the "water-leaving" radiance from the ocean, or a near-infrared fluorescence signature from vegetation. This SL contribution is an upwelling radiance from the lower boundary, and is present in addition to existing diffuse and directly reflected. Here, we are concerned with specifying the appropriate inputs in Tables C4 and G4 in sections 6.1.1.3 and 6.1.1.7, respectively. We note that the supplement also has the observational geometry facility like VLIDORT itself.

In section 6.4.1, we present an overview of SL, and discuss the surface-leaving constructions that are available. This is followed by a note of the supplement's current development state in section 6.4.2. Next, a sample calling sequence for the supplement is given in section 6.4.3. The supplement inputs and outputs are then given in tables in section 6.4.4.

6.4.1 SLEAVE formulation

The SL contribution output from the supplement consists of 3 terms which will in general depend on the solar illumination angle θ_o and the relative azimuth angle $\varphi - \varphi_o$ (think of the water-leaving radiance). These terms are (we have omitted the Stokes-component index for clarity):

$$S_m(\mu_i, \theta_o); \ S_m(\gamma, \theta_o); \ S_{exact}(\gamma, \theta_o, \varphi - \varphi_o). \tag{6.4.1}$$

Here, the Fourier component index $m = 0, 1, ..., 2N - 1$, where N is the number of discrete ordinates in the half-space. The first term is the (Fourier component of) upwelling radiance into the polar discrete ordinate directions μ_i, and this is required for the diffuse-scattering boundary

condition at the lower surface. The second term is the (Fourier component of) upwelling radiance into user-defined stream directions γ, and this is required for post-processing of the discrete ordinate solution (source function integration).

The first two terms arise from the Fourier cosine-azimuth expansions of the full functions $S_{exact}(\mu_i,\theta_o,\varphi-\varphi_o)$ and $S_{exact}(\gamma,\theta_o,\varphi-\varphi_o)$; thus for example:

$$S_{exact}(\mu_i,\theta_o,\varphi-\varphi_o) = S_o(\mu_i,\theta_o) + 2\sum_{m=1}^{\infty} S_m(\mu_i,\theta_o)\cos(\varphi-\varphi_o) \qquad (6.4.2)$$

In the discrete-ordinate approximation we can only use $2N$-1 components in the sum in Eq. (6.4.2). In the post-processing, it is more accurate to use the complete term $S_{exact}(\gamma,\theta_o,\varphi-\varphi_o)$ in place of the Fourier-series truncation, and this "correction" is implemented. This is akin to make a precise calculation of the direct-beam surface reflection, and indeed the "direct" surface term will now include $S_{exact}(\gamma,\theta_o,\varphi-\varphi_o)$ if desired. In this case, the Fourier terms $S_m(\gamma,\theta_o)$ are not needed.

We will also consider the simpler situation where the SL contribution consists of an isotropic term $S^*(\theta_o)$ which depends only on the incoming solar direction (no azimuth dependence, all outgoing directions equal). In this case, $S_m(\mu,\theta_o) = 0 \ (m\geq1)$ and $S_o(\mu,\theta_o) = S^*(\theta_o)$ for all outgoing polar directions μ, and also $S_{exact}(\gamma,\theta_o,\varphi-\varphi_o) = S^*(\theta_o)$.

Linearization. We assume that there is no effect of the atmosphere on the magnitudes of the surface-leaving terms - they depend solely on intrinsic quantities. We will therefore require the supplement to define partial derivatives of the SL terms in Eq. (6.4.1) with respect to some surface property ξ (which might be the wind speed or the fluorescence at 755 nm):

$$\frac{\partial S_{exact}}{\partial\xi}(\gamma,\theta_o,\varphi-\varphi_o); \frac{\partial S_m}{\partial\xi}(\mu_i,\theta_o); \frac{\partial S_m}{\partial\xi}(\gamma,\theta_o) . \qquad (6.4.3)$$

The linearized SL supplement will generate these derivatives, which VLIDORT is then able to ingest, thereby making it possible to generate VLIDORT Jacobian output with respect to surface property ξ.

6.4.2 Current implementations

Water-leaving. In Version 2.6, the water-leaving contribution in the SLEAVE supplements were regarded as isotropic (no angular dependence at all), and equal to the flux-normalized (underwater) upwelling radiance S^* in the ocean at the surface - Fresnel transmittance through the surface was taken to be unity. The value of S^* was obtained through an empirical formulation based on scattering and absorption by ocean optical contributors (pure water, pigment (chlorophyll) and CDOM). The original 6S formulation has been changed several times as new data emerge, and the newest formulation of this underwater term is based on modified-6S code by A. Sayer (private communication); this update is based in part on the recent work of [*Morel*

and Gentili, 2009]. The underwater term is currently dependent only on the wavelength (again in Microns) and the pigment concentration in [mg/M].

The 6S formulation also included a treatment of the ocean-air transmittance factor for a rough surface; this is based on the azimuthal integration (for each solar angle and line of sight angle in the atmosphere) of the reverse-medium glint calculation for the rough-surface interface, taking into account Snell's law of refraction and the conservation of light intensity divided by the square of the reflective index. Glint calculations here are done using the same Cox-Munk calculation as that noted above for the BRDF (with wind-directionality and complex refractive index determined through salinity). As seen in Eq. (6.3.12), the whitecap correction is also required if we are computing water-leaving radiance in conjunction with atmospheric glint with foam.

The newer formulation now creates $S(\mu_0, \mu_1)$ for the direct water leaving term into line of sight direction μ_1, and it is also necessary to compute the Fourier components $S_m(\mu_0, \mu_j)$ for the discrete ordinate directions $\{\mu_j\}$, $j = 1 \cdots N_d$ in order to deal with multiple scattering; with no azimuth dependence, only $S_0(\mu_0, \mu_j)$ for $m = 0$ survives.

Clearly, this water-leaving option has the same inputs as that for the BRDF discussed in the previous section (wavelength, wind speed and direction, salinity, whitecap and facet isotropy flags). The shadow option is not required. The only new variable is the pigment concentration (called here CHLORCONC).

Although the SLEAVE and BRDF supplements are using the same software for glint reflectance, Fresnel coefficients, refractive index calculation and whitecap determination, the supplements have been kept completely separate in the interests of modularity. When using the two supplements together, it is essential that they both operate with a common set of inputs - we have therefore written a subroutine which checks the compatibility of SLEAVE and BRDF inputs in this situation. An input-check routine has already been written to ensure that input solar and line-of-sight input geometrical variables are consistent between VLIDORT and the BRDF supplement, and now that the SLEAVE supplement has non-isotropic functionality, a similar checking routine has been constructed for VLIDORT and SLEAVE consistency.

__Fluorescence__. This is based on the double-Gaussian model in [*Frankenburg et al*., 2012], which has now been used in a number of studies on the fluorescence signature. We would like to thank Chris O'Dell for allowing us to use this model. The calculation is simple:

$$S^*(\theta_b) = F(\lambda, \theta_b) = F_{755}(\theta_b)\{A_1 \exp[-\frac{(\lambda - \lambda_1)^2}{\sigma_1^2}] + A_2 \exp[-\frac{(\lambda - \lambda_2)^2}{\sigma_2^2}]\} \tag{6.4.4}$$

The wavelengths λ_1 and λ_2 correspond to peaks at 683 nm and 730 nm respectively, and all the Gaussian constants are tabulated in the aforementioned reference. The fluorescence at 755 nm is based on a huge multi-year data set derived from satellite observations, and it depends on the solar angle, the 'epoch' (year, month, day, hour, etc.) and the latitude and longitude coordinates.

It follows that the SL supplement input required for use with VLIDORT will require the following inputs: the wavelength λ, a series of solar zenith angles θ_o, and time and geographical variables. Equation (6.4.4) is easy to differentiate with respect to the defining parameters. The main interest here is generation of VLIDORT Jacobians for the parameter $\xi_q = F_{755}(\theta_o)$, for which $\partial S^*(\theta_o) / \partial \xi_q$ is trivial. This option is controlled by a separate Boolean flag. Technically it is

possible to define Jacobians with respect to the Gaussian-function parameters in Eq. (6.4.4), and there is optional code for this possibility.

Remark.

Water-leaving or fluorescence calculations both depend on an input wavelength value (WAVELENGH and FL_WAVELENGTH in Table A below in section 6.4.4.1). These wavelengths must be the same as the wavelength at which the atmospheric optical properties were prepared for the main VLIDORT calculation (see Table A7). An additional check on these wavelengths has been added.

6.4.3 Example calling sequence

For an intensity calculation with SL reflection, the SL inputs required by VLIDORT_MASTER are those specified in section 6.1.1.3 Table C4. In its present state, the SL supplement is only able to supply the isotropic SL term (first row of this table). For a surface property weighting function calculation (using the VLIDORT_LCS_MASTER or VLIDORT_LPS_MASTER subroutine), VLIDORT also requires the linearized SL inputs in section 6.1.1.7 Table G4; again only the first row (isotropic) term is currently available

For a calculation of SL inputs alone (i.e. no linearizations), the calling program sequence is:

```
!  Obtain control variables for the vector SLEAVE input structure from the
!  SLEAVE input configuration file
   call VSLEAVE_INPUTMASTER ( &
        'VSLEAVE_ReadInput.cfg', & ! Input
         VSLEAVE_Sup_In,         & ! Outputs
         VSLEAVE_Sup_InputStatus ) ! Outputs

!  Call the vector SLEAVE supplement master
   call VSLEAVE_MAINMASTER ( &
         VSLEAVE_Sup_In,      & ! Inputs
         VSLEAVE_Sup_Out )      ! Outputs

!  Finish
   write VSLEAVE output to file
```

The first subroutine (VSLEAVE_INPUTMASTER) reads inputs from a SLEAVE configuration file. These include specifications of the numbers and values of angles (solar and viewing angle zeniths, relative azimuths), the number of discrete ordinates, along with (for example) fluorescence and associated control inputs. Angular and control inputs for the SLEAVE supplement must match equivalent inputs for VLIDORT before a VLIDORT radiance calculation with supplement-computed SLEAVE inputs is performed. The SLEAVE input read routine VSLEAVE_INPUTMASTER is of course optional - it is perfectly possible to set these inputs in another manner inside the calling environment itself. Table A in the next section describes the inputs required for a basic SLEAVE calculation.

The main subroutine (VSLEAVE_MAINMASTER) then carries out the computation of the SLEAVE quantities in Eq. (6.4.1), or the isotropic term $S^*(\theta_0)$. The output from this subroutine can then be written to file for subsequent use in VLIDORT itself; it is also possible to combine

the SLEAVE supplement with the main VLIDORT call inside one environment similar to the example above.

For a calculation with SL weighting functions, additional SLEAVE inputs are required. These are listed in Table C in the next section. One can then obtain Jacobians with respect to wind speed or the fluorescence at 755 nm for example. Now, we use the file-read subroutine VSLEAVE_LIN_INPUTMASTER for all inputs (regular and linearized), and the user environment will then call the subroutine VSLEAVE_LIN_MAINMASTER which will deliver the SLEAVE quantities in Eq. (6.4.1) for all the required geometrical configurations, as well as the linearizations of these in Eq. (6.4.3) with respect to a number of SLEAVE properties. For the isotropic option, VSLEAVE_LIN_MAINMASTER will deliver just the isotropic surface leaving term and associated linearizations.

6.4.4 SLEAVE inputs and outputs

This section contains tables regarding (1) SLEAVE supplement input and output type structures and (2) file-read character strings found in the input configuration file VSLEAVE_ReadInput.cfg. Shaded entries in Table A of Section 6.4.4.1 are currently not enabled.

6.4.4.1 Input and output type structures

Table A: Type Structure VSLEAVE_Sup_Inputs

Name	Kind/Intent	Description
DO_SLEAVING	Logical (I)	If set, SLEAVE calculations will be done.
DO_ISOTROPIC	Logical (I)	If set, calculations for only doing the isotropic SLEAVE term will be done.
DO_EXACT	Logical (I)	If set, calculations for doing the exact SLEAVE term will be calculated.
DO_EXACTONLY	Logical (I)	If set, calculations for *only* doing the exact SLEAVE term will be done (no Fourier).
DO_FLUORESCENCE	Logical (I)	If set, calculations for fluorescence will be done.
DO_SOLAR_SOURCES	Logical (I)	Flag for solar beam source of light.
DO_USER_OBSGEOMS	Logical (I)	If set, supplement will compute SLEAVE quantities at observational geometry triplets specified by the user for multiple geometries. Used in conjunction with input variables N_USER_OBSGEOMS and USER_OBSGEOMS.
NSTOKES	Integer (I)	Number of Stokes vector parameters for which calculations will be done.
NSTREAMS	Integer (I)	discrete ordinates (this should correspond to the same quantity in VLIDORT). Must ≤ symbolic dimension MAXSTREAMS.
NBEAMS	Integer (I)	Number solar beams. Must ≤ symbolic dimension MAXBEAMS.
BEAM_SZAS	Real*8 (I)	Solar zenith angles (degrees). Checked internally range [0,90).
N_USER_RELAZMS	Integer (I)	Number of user-defined relative azimuth angles. Must not be greater than symbolic dimension MAX_USER_RELAZMS.
USER_RELAZMS	Real*8 (I)	Array of user-defined elative azimuth angles (in

		degrees) for off-quadrature output. Ordering is not important. Must be between 0 and 180.
DO_USER_STREAMS	Logical (I)	If set, there will be output at a number of off-quadrature zenith angles specified by user.
N_USER_STREAMS	Integer (I)	Number of user-defined viewing zenith angles. Must be not greater than symbolic dimension MAX_USER_STREAMS.
USER_ANGLES_INPUT	Real*8 (I)	Array of user-defined viewing zenith angles (in degrees) for off-quadrature output. Must be between 0 and 90 degrees.
N_USER_OBSGEOMS	Integer (I)	Number of user-defined observational geometry triplets. Must not be greater than the symbolic dimension MAX_USER_OBSGEOMS.
USER_OBSGEOMS (g,3)	Real*8 (I)	Array of g user-defined observational geometry triplets (in degrees) for off-quadrature output. It consists of the geometry triplets (solar zenith angle, viewing angle, relative azimuth angle) for which SLEAVE quantities are desired.
SALINITY	Real*8 (I)	Salinity (in ppt).
CHLORCONC	Real*8 (I)	Chlorophyll concentration (in mg/M).
WAVELENGTH	Real*8 (I)	Current wavelength (in μm). Now checked against ATMOS_WAVELENGTH from Table A7.
WINDSPEED	Real*8 (I)	Wind speed (in m/s) (only for non-isotropic water leaving).
WINDDIR (b)	Real*8 (I)	Wind direction relative to the azimuthal position of the sun for each solar angle b (only for non-isotropic water leaving).
NSTREAMS_AZQUAD	Integer (I)	Number of angles used in azimuthal integration during SLEAVE calculation.
DO_GlintShadow	Logical (I)	If set, calculations accounting for shadowing of wave facets during sunglint will be done.
DO_FoamOption	Logical (I)	If set, calculations accounting for ocean foam will be done.
DO_FacetIsotropy	Logical (I)	If set, wave facets will be considered isotropic.
FL_WAVELENGTH	Real*8 (I)	Current wavelength (in nm). Now checked against ATMOS_WAVELENGTH from Table A7.
FL_LATITUDE	Real*8 (I)	Current latitude (in degrees).
FL_LONGITUDE	Real*8 (I)	Current longitude (in degrees).
FL_EPOCH(6)	Integer (I)	Current epoch (year, month, day, hour, minute, second).
FL_AMPLITUDE755	Real*8 (I)	Amplitude of fluorescence at 755nm. This is actually a scaling of the 755 nm amplitude produced internally.
FL_DO_DATAGAUSSIAN	Logical (I)	Flag for using Gaussian data provided by [*Frankenburg et al.*, 2012]. Suggested default.
FL_INPUTGAUSSIANS(3,2)	Real*8 (I)	If the flag FL_DO_GAUSSIAN is not set, then the user can input Gaussian data as defined in Eq. (6.4.4). These are the amplitude (in W/m^2/sr), central wavelengths (in nm), and standard deviations (also in nm) of Gaussians 1 and 2.

Table B1: Type structure VSLEAVE_Sup_Outputs

Name	Kind/Intent	Description
SLTERM_ISOTROPIC (S,s)	Real*8 (I)	Isotropic surface leaving term for Stokes vector component S and incident solar angle s. Only calculated if flag DO_ISOTROPIC is set (current default)
SLTERM_USERANGLES (S,a,b,s)	Real*8 (I)	Exact surface leaving term for Stokes vector component S, incident solar angle s, reflected line-of-sight angle a, and relative azimuth b.
SLTERM_F_0 (M,S,k,s)	Real*8 (I)	Fourier components M of surface leaving term for Stokes vector component S, incident solar angle s and reflected discrete ordinate k.
USER_SLTERM_F_0 (M,S,a,s)	Real*8 (I)	Fourier components M of surface leaving term for Stokes vector component S, incident solar angle s and reflected line-of-sight zenith angle a.

Table B2: Type Structure VSLEAVE_Input_Exception_Handling

Name	Kind/Intent	Description
STATUS_INPUTREAD	Integer (O)	Overall status of input read.
NINPUTMESSAGES	Integer (O)	Number of input read error messages.
INPUTMESSAGES	Character (O)	Array of input-read error messages.
INPUTACTIONS	Character (O)	Array of input-read actions to take.

Table C: Type Structure VSLEAVE_LinSup_Inputs

Name	Kind/Intent	Description
DO_SL_JACOBIANS	Logical (I)	General flag for doing SL weighting functions.
DO_ISO_JACOBIANS	Logical (I)	If set, an isotropic weighting function will be done.
DO_FL_755_JACOBIANS	Logical (I)	If set, a weighting function w.r.t. the fluorescence amplitude at 755nm will be done.
DO_FL_GAUSS_JACOBIANS(6)	Logical (I)	If set, a weighting function w.r.t. the respective fluorescence Gaussian parameter will be done.
DO_SALINITY_WFS	Logical (I)	If set, a weighting function w.r.t. the ocean salinity will be done.
DO_CHLORCONC_WFS	Logical (I)	If set, a weighting function w.r.t. the ocean chlorophyll concentration will be done.
DO_WINDSPEED_WF	Logical (I)	If set, the wind speed weighting function will be done. May be used when using the new Cox-Munk ocean kernel.

Table D: Type structure VSLEAVE_LinSup_Outputs

Name	Kind/Intent	Description
L_SLTERM_ISOTROPIC (q,S,s)	Real*8 (I)	Linearized Isotropic surface leaving term for Stokes vector component S and incident solar angle s, w.r.t. surface property q.
L_SLTERM_USERANGLES (q,S,a,b,s)	Real*8 (I)	Linearized Exact surface leaving term for Stokes vector component S, incident solar angle s, reflected line-of-sight angle a, and relative azimuth b, w.r.t. surface property q.
L_SLTERM_F_0 (q,M,S,k,s)	Real*8 (I)	Linearized Fourier components M of surface leaving term for Stokes vector component S, incident solar angle s and reflected discrete ordinate k, w.r.t. surface property q.

L_USER_SLTERM_F_0 (q,M,S,a,s)	Real*8 (I)	Linearized Fourier components *M* of surface leaving term for Stokes vector component *S*, incident solar angle *s* and reflected line-of-sight zenith angle *a,* w.r.t. surface property *q*.

6.4.4.2 SLEAVE configuration file character strings

Table E1: File-read Character strings for *some* variables in SLEAVE Supplement Table A

Name	*Kind*	*Character string in Configuration file*
DO_USER_STREAMS	Logical	Use user-defined viewing zenith angles?
DO_SLEAVING	Logical	Do surface-leaving Contributions?
DO_FLUORESCENCE	Logical	Do surface-leaving Fluorescence?
DO_ISOTROPIC	Logical	Do Isotropic surface-leaving?
DO_EXACT	Logical	Do Overall-Exact kernels?
DO_EXACTONLY	Logical	Do Exact-only (no Fourier-term contributions)?
DO_SL_JACOBIANS	Logical	Do surface-leaving Jacobians?
DO_ISO_JACOBIANS	Logical	Do Isotropic surface-leaving Jacobians?
DO_OBSERVATION_GEOMETRY	Logical	Do Observation Geometry?
SALINITY	Real*8	Ocean water salinity [ppt]
CHLORCONC	Real*8	Chlorophyll concentration in [mg/M]
WAVELENGTH	Real*8	Wavelength in [Microns]
WINDSPEED	Real*8	Windspeed in [m/s]
WINDDIR	Real*8	Wind directions (degrees) relative to sun positions
DO_FoamOption	Logical	Do whitecap (foam) calculation?
DO_FacetIsotropy	Logical	Do glint calculation with facet isotropy?
DO_GlintShadow	Logical	Do glint calculation with shadowing?
DO_CHLORCONC_WF	Logical	Do pigment concentration weighting function?
DO_WINDSPEED_WF	Logical	Do wind-speed weighting function?
NSTOKES	Integer	Number of Stokes vector components
NSTREAMS_AZQUAD	Integer	Number of azimuth quadrature streams
FL_LATITUDE	Real*8	Latitude for Fluorescence model [degs]
FL_LONGITUDE	Real*8	Longitude for Fluorescence model [degs]
FL_EPOCH(6)	Integer	Epoch for Fluorescence model
FL_WAVELENGTH	Real*8	Wavelength for Fluorescence model in [nm]
FL_AMPLITUDE755	Real*8	Amplitude for Fluorescence model at 755 nm
FL_DO_GAUSSIAN	Logical	Do Data Gaussians in Fluorescence?
DO_FL_755_JACOBIANS	Logical	Do Jacobians for F755 Fluorescence value?
DO_FL_GAUSS_JACOBIANS(6)	Logical	Do Gaussian parameter Jacobians for Fluorescence?
NSTREAMS	Integer	Number of half-space streams
NBEAMS	Integer	Number of solar zenith angles
BEAM_SZAS	Real*8	Solar zenith angles (degrees)
N_USER_RELAZMS	Integer	Number of user-defined relative azimuth angles
USER_RELAZMS	Real*8	User-defined relative azimuth angles (degrees)
N_USER_STREAMS	Integer	Number of user-defined viewing zenith angles
USER_ANGLES_INPUT	Real*8	User-defined viewing zenith angles (degrees)
N_USER_OBSGEOMS	Integer	Number of Observation Geometry inputs
USER_OBSGEOM_INPUT	Real*8	Observation Geometry inputs

6.5 Using VLIDORT for certain applications

6.5.1 Generating AMFs and Scattering-weight AMFs with VLIDORT

6.5.1.1 Traditional Definition

For the DOAS-style retrieval of the vertical total column of trace-gas "g" which is an optically thin absorber, the AMF definition that is used in many DOAS application is

$$W = -\frac{1}{\tau}\log\left[\frac{I_{nogas}}{I_{gas}}\right] \qquad (6.5.1)$$

Here, I_{nogas} is the radiance calculated without the presence of absorption by trace gas g, I_{gas} is the radiance calculated with this absorption by trace gas g, and τ is the total atmospheric vertical optical depth of the absorber gas. From a practical point of view, this requires two separate calculations using VLIDORT, one in which the trace-gas absorption is included in every layer of the atmosphere, and the other in which these trace gas layer optical depths for absorption are omitted.

It is also possible to define the AMF for a single layer:

$$W_n = -\frac{1}{\tau_n}\log\left[\frac{I_{nogas,n}}{I_{gas}}\right] \qquad (6.5.2)$$

where τ_n is the layer optical depth of trace gas g, and $I_{nogas,n}$ is the radiance with absorption omitted in layer n.

6.5.1.2 Scattering weight AMF

Also of use in many DOAS applications is the so-called scattering-weight AMF W_n, which is defined by

$$W_n = -\frac{\partial \log I}{\partial \tau_n} \qquad (6.5.3)$$

I is the radiance calculated including absorption. This is very simply written down in terms of the profile Jacobian output K_n (with respect to τ_n in layer n) from VLIDORT:

$$W_n = -\frac{1}{\tau_n}\frac{K_n}{I}; \qquad K_n = \tau_n\frac{\partial I}{\partial \tau_n}. \qquad (6.5.4)$$

This quantity is sometimes normalized to the geometrical AMF: $W_{geo} = \mu_0^{-1} + \mu_1^{-1}$ for solar and viewing zenith angle cosines μ_0 and μ_1.

The profile Jacobian facility is very useful for this quantity, and it requires the user to set up the appropriate linearized optical property inputs to VLIDORT. We give one example. For Rayleigh scattering with one trace gas absorber, the VLIDORT bulk optical properties are $\Delta_n = R_n + \tau_n$, $\omega_n = R_n/\Delta_n$, where R_n is the Rayleigh scattering optical depth in layer n, and Δ_n, ω_n the layer optical depth for extinction and scattering albedo respectively. VLIDORT requires as input the linearized quantities:

$$V_n \equiv \frac{\tau_n}{\Delta_n} \frac{\partial \Delta_n}{\partial \tau_n} = \frac{\tau_n}{\Delta_n}; \quad U_n \equiv \frac{\tau_n}{\omega_n} \frac{\partial \omega_n}{\partial \tau_n} = - V_n \tag{6.5.5}$$

[The phase function has no derivatives in this case].

6.5.2 Computations with Planck Functions: Relations between Spectral Grids

The following are some relations which may serve as a handy reference when performing computations involving Planck functions on different spectral grids (a wavelength (λ) grid and wavenumber ($\tilde{\nu}$) grid being used here). In the expressions below, we define

$$\lambda_1 = \frac{1}{\tilde{\nu}_2}, \ \lambda_2 = \frac{1}{\tilde{\nu}_1}, \ \Delta\lambda = \lambda_2 - \lambda_1, \ \Delta\tilde{\nu} = \tilde{\nu}_2 - \tilde{\nu}_1$$

where $\lambda_2 > \lambda_1$ and $\tilde{\nu}_2 > \tilde{\nu}_1$.

- For a monochromatic case:

$$B_\lambda(T)d\lambda = -B_{\tilde{\nu}}(T)d\tilde{\nu} \tag{6.5.6a}$$
$$B_\lambda(T) \quad = -B_{\tilde{\nu}}(T)\frac{d\tilde{\nu}}{d\lambda} \tag{6.5.6b}$$

$$= -B_{\tilde{\nu}}(T)(-\frac{\tilde{\nu}}{\lambda})$$

$$B_\lambda(T) = B_{\tilde{\nu}}(T) \cdot \frac{\tilde{\nu}}{\lambda} \tag{6.5.6c}$$

- For a finite spectral subinterval case:

$$\int_{\lambda_1}^{\lambda_2} B_\lambda(T)d\lambda = \int_{\tilde{\nu}_1}^{\tilde{\nu}_2} B_{\tilde{\nu}}(T)d\tilde{\nu} \tag{6.5.7a}$$

By the mean value theorem for integrals from the calculus, for a function $y = f(x)$ that is continuous, there exists a mean value $\bar{f}$ such that $\bar{f} \cdot \Delta x = \int_{x_1}^{x_2} f(x)dx$. Since the Planck function is such a function, on the λ-grid we may write $\overline{B_\lambda(T)} \cdot \Delta\lambda = \int_{\lambda_1}^{\lambda_2} B_\lambda(T)d\lambda$. Using this and defining the quantity $B_{\Delta\tilde{\nu}}(T) = \int_{\tilde{\nu}_1}^{\tilde{\nu}_2} B_{\tilde{\nu}}(T)d\tilde{\nu}$, (6.5.7a) may be re-written as:

$$\overline{B_\lambda(T)} \cdot \Delta\lambda = B_{\Delta\tilde{\nu}}(T) \tag{6.5.7b}$$
$$\overline{B_\lambda(T)} = \frac{B_{\Delta\tilde{\nu}}(T)}{\Delta\lambda} \tag{6.5.7c}$$

We note that the quantity $B_{\Delta\tilde{\nu}}(T)$ is a band intensity in units [W/(m^2·sr)] and can be generated by the LIDORT family subroutine `get_planckfunction`. Also, on the $\tilde{\nu}$-grid we may write

$$\overline{B_{\tilde{\nu}}(T)} \cdot \Delta\tilde{\nu} = \int_{\tilde{\nu}_1}^{\tilde{\nu}_2} B_{\tilde{\nu}}(T)d\tilde{\nu} \tag{6.5.8a}$$

$$\overline{B_{\tilde{\nu}}(T)} \cdot \Delta\tilde{\nu} = B_{\Delta\tilde{\nu}}(T) \tag{6.5.8b}$$

$$\overline{B_{\tilde{\nu}}(T)} = \frac{B_{\Delta\tilde{\nu}}(T)}{\Delta\tilde{\nu}} \tag{6.5.8c}$$

Specifically, using (6.5.7b) and (6.5.8b) yields

$$\overline{B_\lambda(T)} \cdot \Delta\lambda = \overline{B_{\tilde{\nu}}(T)} \cdot \Delta\tilde{\nu} \tag{6.5.9a}$$

$$\overline{B_\lambda(T)} = \overline{B_{\tilde{\nu}}(T)} \cdot \frac{\Delta\tilde{\nu}}{\Delta\lambda} \tag{6.5.9b}$$

$$\approx \overline{B_{\tilde{\nu}}(T)} \cdot \left|\frac{d\tilde{\nu}}{d\lambda}\right| \tag{6.5.9c}$$

$$\overline{B_\lambda(T)} \approx \overline{B_{\tilde{\nu}}(T)} \cdot \frac{\tilde{\nu}}{\lambda} \tag{6.5.9d}$$

Eq. (6.5.9d) is similar to the monochromatic relation given in (6.5.6c); however, care must be taken to insure that the spectral subinterval over which the values in (6.5.9d) are computed is such that the approximation $\frac{\Delta\tilde{\nu}}{\Delta\lambda} \approx \left|\frac{d\tilde{\nu}}{d\lambda}\right|$ holds. Based on numerical experimentation, with $\Delta\tilde{\nu} = 1cm^{-1}$ this approximation holds well in the spectral range $\tilde{\nu} \in [50cm^{-1}, 50000cm^{-1}]$ (equivalent to the range $\lambda \in [200nm, 200000nm]$). This information is particularly useful as it applies to the spectral range covered by the range of solar flux data found in data files of solar fluxes (e.g. in the solar spectrum file "newkur.dat" which is sometimes used in certain LIDORT- or VLIDORT-based RT simulation packages).