User's Guide

LIDORT
Version 3.0

Robert Spurr

RT Solutions, Inc.
9 Channing Street, Cambridge, MA 02138, USA
Tel. +1 617 492 1183
Fax: +1 617 354 3415
email: rtsolutions@verizon.net

Abstract

This is the User's Guide to LIDORT Version 3.0, issued in December 2005 in conjunction with the release of the Version 3.0 software package and accompanying license. The code will for now remain in the public domain, and the associated license closely follows the GNU public license formulation. Version 3.0 is the third official release, following the distribution of Version 1.1 in November 1999, and Versions 2.1 and 2.3 in 2001 and 2002 respectively. Former releases were done while the author was at the Smithsonian Astrophysical Observatory (SAO); this release is the first from RT Solutions, Inc. Version 3.0 release supersedes all earlier official releases and also some research codes (Versions 2.4 and 2.2+) developed in 2002-2004. Earlier codes will no longer be supported; all enquiries and support regarding the present release should be addressed to R. Spurr at RT Solutions.

Table of Contents

1. Introduction to LIDORT 3.0

1.1. Motivation for the linearized models

Linearization techniques in radiative transfer (RT) theory have been developed for the automatic generation of weighting functions or Jacobians (analytic derivatives of the radiance field with respect to atmospheric and surface variables). These methods have helped to open the way for the direct fitting of moderate-resolution backscatter radiance measurements using classical inversion techniques based on nonlinear least-squares iteration (with and without regularizations from *a priori* estimates or other sources).

Analytic Jacobians have been a feature of infrared transmittance forward models for many years. Such models are based on Beer's law of extinction in the absence of scattering, and the differentiation of exponential attenuations is straightforward and fast. With the advent of remote sensing atmospheric chemistry instruments such as GOME and SCIAMACHY measuring at moderately high spectral resolution in the visible and ultraviolet, it is necessary to use multiple scattering radiative transfer models in the inversion process for the determination of products such as ozone profiles. Although weighting functions can be determined for these applications by finite difference estimation using repeated calls to the RT model, this process is time-consuming and computationally inefficient.

Indeed, the retrieval of ozone profiles from GOME measurements has provided the impetus for linearization of radiative transfer multiple scatter models in multi-layer atmospheres. With linearized RT code, one call to the model is sufficient to return both the simulated radiance field and all relevant Jacobians, the latter determined analytically. Aside from the operational generation of weighting functions for different types of remote sensing applications, the linearization facility is tremendously useful for sensitivity studies and error budget analyses.

1.2. Historical review of the LIDORT family

The first versions of LIDORT were 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 plane-parallel solar beam and thermal sources [1]. Version 1.1 of LIDORT was able to generate atmospheric profile weighting functions and surface albedo weighting functions (Lambertian). It also included a 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 solutions for solar beam source terms were also developed as an alternative to the classical substitution methods due to Chandrasekhar. This work culminated in the release of Versions 2.3S (radiance only) and 2.3E (with Jacobians) [2, 3].

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 version was used in a number of applications involving ozone profile and column retrievals from instruments such as GOME and OMI. The original LIDORT-RRS research code has recently been re-packaged and will be given public release early in 2006. A separate User's Guide is available for LIDORT-RRS, and the model itself will be linearized in 2006.

In 2003, the LIDORT Version 2.2+ code was developed as a super-environment for LIDORT [4]; 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 provided a number of extensions to deal in particular with bi-directionally reflecting surfaces [5]. 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 leaf area index and wind speed.

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., 2004]. This has opened the way for a new approach to simultaneous retrieval of atmospheric and ocean quantities from MODIS and related instruments.

A vector version VLIDORT was developed in 2004, based on the use of complex solutions of the homogeneous radiative transfer discrete ordinate equations. This was given the pseudo-spherical treatment, and in 2005 it was linearized and given the enhanced sphericity treatment. A new version (VLIDORT 2.0) was released late in 2005 in conjunction with the LIDORT 3.0 release. Work is progressing on the polarization code, and it is expected that the LIDORT and VLIDORT codes will have matching capabilities (in terms of output options) by the end of 2006.

1.3. Overview of LIDORT Version 3.0

Support for maintaining LIDORT codes has had to be extracted from a series of small 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 3.0; the present User's Guide has been written under this new aegis.

One of the problems with the LIDORT family has been the proliferation of research codes for specific applications, and in Version 3.0, I have attempted to bring together many of the LIDORT family members in a comprehensive whole. Thus Version 3.0 will encompass all capabilities in Versions 2.1, 2.2+, 2.3 and 2.4, as well as including a number of new features arising from recent research work. There is one remaining older version (LIDORT V2.5); it has

the specialist ability for fast generation of total column (as opposed to profile) Jacobians, and it is expected that this capability will be integrated into LIDORT V3.0 in 2006. Table 1 outlines the Version 3.0 capabilities as of December 2005.

Table 1. Major features of LIDORT 3.0.

Feature	*Origin*
Pseudo-spherical (solar beam attenuation)	2.1, 2.3
Enhanced spherical (line-of-sight)	2.2+
3-kernel BRDF + linearization	2.4
Multiple solar zenith angles	New
Solution saving, BVP telescoping	New

A description of the model is given in section 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. Many of these details may be found in the standard text [6], and in the papers by R. Spurr. Other sections describe the linearization process and the derivation of weighting functions for atmospheric and surface quantities.

The new features for Version 3.0 are mainly concerned with performance enhancement; they are described in detail below in Chapter 3. The multiple SZA facility (section 3.1) is particularly useful for look-up table generation and the deployment of the model for large numbers of RT simulations. The solutions saving (section 3.2) and BVP telescoping (section 3.3) options are labor saving devices designed to speed up performance through the elimination of unnecessary computation. The exact treatments of single scattering and direct beam contributions have been reviewed for this release, and a new strategy for Fourier-series convergence is outlined in section 3.4.

LIDORT is a scattering code that takes total layer optical properties for input; the code does not distinguish individual trace gas absorbers or particulate scatterers. In section 4.1, we outline the derivation of the standard set of optical properties required for the computation of the radiance field; this derivation is the same as that for DISORT [7]. In section 4.2 we present derivations of linearized optical property inputs for the generation of atmospheric Jacobians, and the analogous treatment for surface property Jacobians is done in section 4.3.

LIDORT usage is dealt with in Chapter 5. Following the directory structure overview in section 5.1, we give precise descriptions of the input and output variables (section 5.2) and a description of the configuration file for input settings (section 5.3). In section 5.4 we discuss the "makefile" production of executables, and discuss installation. Section 5.5 deals with exception handling and utilities in the model. This version of LIDORT is in public domain; the license is found in section 5.6.

A number of tests have been written for LIDORT. These tests cover a series of benchmark situations, and proper installation of the package will result in the confirmation of the test data set that accompanies the release.

2. Summary of the LIDORT 3.0 model

2.1. Homogeneous and particular RT solutions

2.2. Boundary value problems and post-processing

2.3. Linearization of the entire RT solution

2.4. BRDF and surface weighting function treatment

2.5. Enhanced sphericity: the LIDORT 3.0+ environment

This section is currently under construction.

3. New Performance Features in LIDORT 3.0

3.1. Multiple Solar Zenith Angles

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 q0 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 and for all 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 $\mathbf{AX}_g = \mathbf{B}_g$, where $\mathbf{X}_g$ 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 $\mathbf{A}$ contains only contributions from the RTE homogeneous solutions. The inverse matrix $\mathbf{A}^{-1}$ 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 $\mathbf{X}_g = \mathbf{A}^{-1}\mathbf{B}_g$ 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 LIDORT 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. In LIDORT 3.0, 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.

3.2. Solution saving

DISORT and LIDORT have always performed full determination of all solutions to the radiative transfer equation (RTE) in all layers and for all Fourier components contributing to radiance outputs. This is regardless of the scattering properties of the layer: the RTE is solved by first calling an eigen-solver 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: extinction across the layer with transmittance factors $T_n(\mu) = \exp[-\Delta_n/\mu]$, where μ is any polar direction and Δ_n is 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 j^{th} homogeneous solution vector $\mathbf{X}_j$ (size N) 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 $L[\Delta_n]$ exists, then the linearization will pick up an additional term $L[T_n(\mu)] = -\mu^{-1}\, T_n(\mu)\, L[\Delta_n]$.

Rayleigh scattering has a $P(\Theta) \sim \cos^2\Theta$ phase matrix 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 layers, there will be a substantial reduction in RTE solution computations, and consequently a marked improvement in performance (see section 2.4). In the more general case, 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 used 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 $0 \le \lambda \le 2N-1$. Solution saving occurs when $\beta_\lambda = 0$ for $m \le \lambda \le 2N\text{-}1$; there is then no scattering for Fourier component m and higher.

3.3. 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

$$I^{\pm}(x,\mu_i) = \sum_{\alpha=1}^{N}\left[L_{n\alpha}\, X^{\pm}_{in\alpha}\, e^{-k_{n\alpha}x} + M_{n\alpha}\, X^{\mu}_{in\alpha}\, e^{-k_{n\alpha}(\Delta_n - x)}\right] + G^{\pm}_{in}\, e^{-x/\mu_0}. \quad (1)$$

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 $X^{\pm}_{ip\alpha} = \delta_{i\alpha}$ and $G^{\pm}_{ip} = 0$ for all Fourier $m > 2$ and for all layers $p \ne n$. In this case the downwelling and upwelling solutions are:

$$I^{\downarrow}_{pj}(x) = L_{pj} \exp[-x/\mu_j]; \quad (2a)$$

$$I^{\uparrow}_{pj}(x) = M_{pj} \exp[-(\Delta_p - x)/\mu_j]. \quad (2b)$$

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

$$L_{p+1,j} = L_{pj} \exp[-\Delta_p / \mu_j]; \qquad (3a)$$

$$M_{p-1,j} = M_{pj} \exp[-\Delta_p / \mu_j]. \qquad (3b)$$

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 write down the boundary conditions for layer n using Eqs. (1), (2) and (3). At the top of the layer we have:

$$\sum_{\alpha=1}^{N} \left[L_{n\alpha} X^{+}_{in\alpha} + M_{n\alpha} X^{-}_{in\alpha} \Theta_{n\alpha} \right] + G^{+}_{in} = L_{n-1,i} C_{n-1,i}; \qquad (4a)$$

$$\sum_{\alpha=1}^{N} \left[L_{n\alpha} X^{-}_{in\alpha} + M_{n\alpha} X^{+}_{in\alpha} \Theta_{n\alpha} \right] + G^{-}_{in} = M_{n-1,i}. \qquad (4b)$$

At the bottom of the active layer, we have

$$\sum_{\alpha=1}^{N} \left[L_{n\alpha} X^{+}_{in\alpha} \Theta_{n\alpha} + M_{n\alpha} X^{-}_{in\alpha} \right] + G^{+}_{in} \Lambda_{n\alpha} = L_{n+1,i}; \qquad (5a)$$

$$\sum_{\alpha=1}^{N} \left[L_{n\alpha} X^{-}_{in\alpha} \Theta_{n\alpha} + M_{n\alpha} X^{+}_{in\alpha} \right] + G^{-}_{in} \Lambda_{n\alpha} = M_{n+1,i} C_{n+1,j}. \qquad (5b)$$

We have used the following abbreviations:

$$\Theta_{n\alpha} = \exp[-k_{n\alpha} \Delta_n], \; \Delta_n = \exp[-\eta_n \Delta_n], \; C_{nj} = \exp[-\Delta_n / \mu_j]. \qquad (6)$$

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. (3a) 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 $p = \mathrm{S}$ and hence by Eq. (14b) also for all $p > n$. With these conditions, Eqs. (4a) and (5b) become:

$$\sum_{\alpha=1}^{N} \left[L_{n\alpha} X^{+}_{in\alpha} + M_{n\alpha} X^{-}_{in\alpha} \Theta_{n\alpha} \right] = -G^{+}_{in}; \qquad (7a)$$

$$\sum_{\alpha=1}^{N} \left[L_{n\alpha} X^{-}_{in\alpha} \Theta_{n\alpha} + M_{n\alpha} X^{+}_{in\alpha} \right] = -G^{-}_{in} \Lambda_{n\alpha}. \qquad (7b)$$

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 (4b) to find $\mathbf{M}_{n-1}$ and for remaining layers to TOA we use (3b). Similarly for the layer immediately below n, we use (5a) to find $\mathbf{L}_{n+1}$ and for remaining layers to BOA, we use (3a). 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 $\mathbf{A}\mathrm{L_k}[\mathbf{Y}]=\mathbf{B^*}=\mathrm{L_k}[\mathbf{B}]-\mathrm{L_k}[\mathbf{A}]\mathbf{Y}$; 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 $\mathbf{B}^*$ 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 $\mathbf{B}^*$ 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.

3.4. Exact single scatter and direct beam contributions

The Nakajima-Tanaka TMS correction [*Nakajima and Tanaka*, 1988] has been a feature of LIDORT from Version 2.1 onwards. In essence, the correction involves an exact calculation of the single scatter contribution using an unlimited number of (non delta-M scaled) phase function moments, 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: $\mathbf{I'} = \mathbf{I} + \mathbf{I}_{\mathbf{SSexact}} - \mathbf{I}_{\mathbf{SStrunc}}$; Fourier convergence is applied to $\mathbf{I}$. In LIDORT, the unwanted truncated SS term is simply omitted from the start, with only the diffuse field being computed: $\mathbf{I'} = \mathbf{I}_{\mathbf{mult}} + \mathbf{I}_{\mathbf{SSexact}}$, with Fourier convergence applied only to the diffuse term $\mathbf{I}_{\mathbf{mult}}$. 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 LIDORT, 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 applying TMS first and including $\mathbf{I}_{\mathbf{SSexact}}$ 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 $\mathbf{I}_{\mathbf{SSexact}}$ offset, so that the addition of increasingly smaller Fourier terms will be less of an influence on the total. This is now the policy in LIDORT 3.0: the TMS correction is done first, and no longer applied as an *ex post facto* correction.

It turns out that a similar consideration applies to the direct beam intensity field (the direct solar beam reflected off the surface, with no atmospheric scattering). For a non-Lambertian surface with known BRDF functions, LIDORT will calculated the Fourier contributions from the BRDF terms, and (for the upwelling field) deliver the Fourier components of the post-processed direct-beam reflection as well as the diffuse and single scatter contributions. The complete Fourier-summed direct beam contribution is necessarily truncated because of the discrete ordinate process. It is possible to compute an exact BRDF contribution (no Fourier component) for the direct beam, using the original viewing angles, and this $\mathbf{I}_{\mathbf{DBexact}}$ term will then replace the truncated contribution $\mathbf{I}_{\mathbf{DBtrunc}}$.

This feature has now been installed in Version 3.0 and functions in the same way as the TMS correction: the truncated form $\mathbf{I}_{\mathbf{DBtrunc}}$ is simply not calculated, and the exact form $\mathbf{I}_{\mathbf{DBexact}}$ is computed right from the start as an initial correction, and included in the convergence testing along with the TMS contribution. 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.

3.5. Some performance tests

We look at improvements in the performance of LIDORT for a test case involving one active layer with aerosols (lowest layer) in a 13-layer atmosphere with Rayleigh scattering and ozone absorption at 324.85 nm (Table 1).

Table 1. Some performance tests for LIDORT 3.0.

Test	# SZAs	# Fourier	# Streams	Solution Saving	BVP Telescoping	Time (secs)	Saving (%)
1	15	7/8	10	N	N	0.538	-------
2	15	7/8	10	Y	N	0.505	6.2
3	15	7/8	10	Y	Y	0.357	33.7 (64.5)
4	5	8	10	N	N	0.209	-------
5	5	8	10	Y	N	0.188	10.0
6	5	8	10	Y	Y	0.135	35.4
7	1	8	10	N	N	0.067	-------
8	1	8	10	Y	N	0.057	14.9
9	1	8	10	Y	Y	0.043	37.3
This next set is for 15 SZAs but with separate calls							
10	1x15	7/8	10	N	N	1.005	-------
11	1x15	7/8	10	Y	N	0.855	14.9
12	1x15	7/8	10	Y	Y	0.545	37.3

We set the Jacobian input so that 2 atmospheric-property weighting functions are generated for each layer (a total of 26 Jacobians), and one surface weighting functions (with respect to the albedo). There are 3 output levels, and 4 user geometries; upwelling and downwelling fields are produced, making a total of 24 output options. Thus for each input solar zenith angle (SZA) there are 24 radiance outputs, 648 (=24x27) Jacobians, and hence 15 SZAs the total number of outputs is 10080 (=15x672) for one call to the model.

In Table 1 we indicate the number of solar zenith angles. All calculations were done with 10 discrete ordinate in the half-space (20 in the full space), and calculations typically required 7 or 8 Fourier component to converge to an accuracy of 1 in 10^4. It is clear that there is considerable advantage in using the multi-SZA facility. Without it, the timing for 15 SZAs is 1.005 for a full calculation without any performance enhancements. With it, and assuming also that the 2

performance enhancements are switched on, the timing is 0.357 (Line 3), a saving of 64.5% (figure in brackets): nearly 3 times as fast.

These figures are for modest discretizations. With larger values for the number of discrete ordinate streams and the number of atmospheric layers, the performance improvements will be more marked.

4. Preparation of inputs for LIDORT 3.0

4.1. Atmospheric optical property inputs for the intensity field

LIDORT will calculate the radiance field I and any normalized Jacobian $K_\zeta \equiv \zeta \partial I / \partial \zeta$, where ζ is any atmospheric or surface parameter. LIDORT (like DISORT) is a pure scattering model that ingests optical property inputs for scattering and extinction. It is up to the user to construct these inputs from the detailed physics entailed in the particular application. For Jacobian output, it is only necessary to input the derivatives of these optical properties with respect to the ζ parameters.

We consider the construction of optical property inputs $\{\tau, \omega, \gamma\}$ for one atmospheric layer. The symbols $\{\tau, \omega, \gamma\}$ refer to the *total layer optical thickness* for extinction, the *total single scatter albedo* and the *total phase function Legendre expansion coefficients* respectively. These are total inputs; there is no reference to the nature of the scatterer or the type of trace gas absorber. Layers are assumed to be optically uniform.

In our example here, radiative processes will include Rayleigh scattering by air molecules, trace gas absorption and scattering and extinction by aerosols. 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:

$$\tau_{aer} = Ne_{aer} \equiv N[fe_1 + (1-f)e_2]; \tag{4.1a}$$

$$\omega_{aer} = \frac{\beta_{aer}}{e_{aer}} \equiv \frac{fz_1e_1 + (1-f)z_2e_2}{e_{aer}}; \tag{4.1b}$$

$$\gamma_{aer}^{(l)} = \frac{fz_1e_1\gamma_1^{(l)} + (1-f)z_2e_2\gamma_2^{(l)}}{\beta_{aer}}. \tag{4.1c}$$

These optical properties refer to the aerosols; the quantity β_{aer} is the combined scattering coefficient and e_{aer} the combined extinction coefficient. The quantity $\gamma^{(l)}$ is the l-th coefficient in the Legendre polynomial expansion of the phase function. Here, e_1, z_1 and $\gamma_1^{(l)}$ are the extinction coefficient, single scatter albedo and Legendre expansion coefficient for aerosol type 1; similar definitions apply to aerosol type 2.

If in addition we have Rayleigh scattering optical depth β_{Ray} and trace gas absorption optical thickness α_{gas}, then the *total optical property inputs* are given by:

$$\tau = \alpha_{gas} + \beta_{Ray} + \tau_{aer}; \; \omega = \frac{N\beta_{aer} + \beta_{Ray}}{\tau}; \; \gamma^l = \frac{\gamma_{Ray}^l\beta_{Ray} + \gamma_{aer}^l N\beta_{aer}}{\beta_{Ray} + N\beta_{aer}}. \tag{4.2}$$

The quantity β_{aer} may be established as the product of the Rayleigh cross-section and the air column: $\beta_{Ray} = \rho_{air}\sigma_{Ray}$; similarly the trace gas absorption optical thickness is commonly given as a product of the layer column density and the trace gas absorption cross-section. The Rayleigh

phase function coefficients are expressed in terms of the depolarization ratio δ; the only surviving coefficient is: $\gamma_{Ray}^{(2)} = (1-\delta)/(2+\delta)$. Aerosol quantities must in general be derived from a suitable particle scattering model (Mie calculations, T-matrix methods, etc.).

4.2. Derivative inputs for the atmospheric weighting function fields

If we require Jacobian output, then LIDORT must also be given the derivative optical property inputs, that is, the partial derivatives of the original inputs $\{\tau, \omega, \gamma\}$ with respect to layer parameters for which we require weighting functions. These 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 an example (keeping to the notation used in section 4.1 above), 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. (4.2). After some algebra, one finds.

$$N\frac{\partial\tau}{\partial N} = N\frac{\partial\tau_{aer}}{\partial N} = \tau_{aer} \qquad f\frac{\partial\tau}{\partial f} = f\frac{\partial\tau_{aer}}{\partial f} = fN(e_1 - e_2)$$

$$N\frac{\partial\omega}{\partial N} = \frac{N\beta_{aer} - \omega\tau_{aer}}{\tau} \qquad f\frac{\partial\omega}{\partial f} = \frac{fN\left[\left(z_1e_1 - z_2e_2\right) - \omega(e_1 - e_2)\right]}{\tau}$$

$$N\frac{\partial\gamma}{\partial N} = \frac{N\beta_{aer}\left(\gamma_{aer} - \gamma\right)}{N\beta_{aer} + \beta_{Ray}} \qquad f\frac{\partial\gamma}{\partial f} = \frac{fN(z_1e_1 - z_2e_2)\left(\gamma_{aer} - \gamma\right)}{N\beta_{aer} + \beta_{Ray}}$$

In this set of results, we have dropped the Legendre moment index (l) in the interest of clarity. The derivatives here have been normalized.

For *sensitivity Jacobians*, the quantities β_{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:

$$\frac{\partial\tau}{\partial\beta_{Ray}} = 1; \qquad \frac{\partial\omega}{\partial\beta_{Ray}} = \frac{1-\omega}{\tau}; \qquad \frac{\partial\gamma}{\partial\beta_{Ray}} = \frac{\gamma_{Ray} - \gamma}{N\beta_{aer} + \beta_{Ray}};$$

$$\frac{\partial\tau}{\partial\alpha_{Gas}} = 1; \qquad \frac{\partial\omega}{\partial\alpha_{Gas}} = -\frac{\omega}{\tau}; \qquad \frac{\partial\gamma}{\partial\alpha_{Gas}} = 0;$$

$$\frac{\partial\tau}{\partial e_1} = Nf; \qquad \frac{\partial\omega}{\partial e_1} = \frac{Nf(z_1 - \omega)}{\tau}; \qquad \frac{\partial\gamma}{\partial e_1} = \frac{fz_1\left(\gamma_{aer} - \gamma\right)}{N\beta_{aer} + \beta_{Ray}};$$

$$\frac{\partial\tau}{\partial e_2} = N\left(1-f\right); \qquad \frac{\partial\omega}{\partial e_2} = \frac{N(1-f)(a_2 - \omega)}{\tau}; \qquad \frac{\partial\gamma}{\partial e_2} = \frac{(1-f)z_2\left(\gamma_{aer} - \gamma\right)}{N\beta_{aer} + \beta_{Ray}};$$

$$\frac{\partial \tau}{\partial z_1} = 0; \qquad \frac{\partial \omega}{\partial z_1} = \frac{Nfe_1}{\tau} \quad ; \qquad \frac{\partial \gamma}{\partial z_1} = \frac{fe_1\left(\gamma_{aer} - \gamma\right)}{N\beta_{aer} + \beta_{Ray}};$$

$$\frac{\partial \tau}{\partial z_2} = 0; \qquad \frac{\partial \omega}{\partial z_2} = \frac{N(1-f)e_2}{\tau}; \qquad \frac{\partial \gamma}{\partial z_2} = \frac{(1-f)e_2\left(\gamma_{aer} - \gamma\right)}{N\beta_{aer} + \beta_{Ray}}$$

In this example, we have determined derivative input for 2 retrieval parameters and 6 sensitivity parameters. With these inputs, LIDORT will generate 8 analytic weighting functions for this layer.

It should be noted that LIDORT actually takes fully normalized derivative inputs:

$$v_\xi \equiv \frac{\xi}{\tau}\frac{\partial \tau}{\partial \xi};\ u_\xi \equiv \frac{\xi}{\omega}\frac{\partial \omega}{\partial \xi};\ \varphi_\xi^{(l)} \equiv \frac{\xi}{\gamma^{(l)}}\frac{\partial \gamma^{(l)}}{\partial \xi}, \tag{4.3}$$

where ξ is any atmospheric quantity varying in the given layer. These quantities are easily established from the above definitions.

4.3. Additional input specifications (atmospheric)

LIDORT 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 Re and a height grid $\{z_n\}$ where n is a layer index running from $n = 0$ to $n =$ 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. The pseudo-spherical calculation inside LIDORT returns atmospheric slant path distances appropriate to solar beam attenuation. This option

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 LIDORT 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.

4.4. Inputs for surface properties

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 LIDORT will return the Lambertian weighting function $K_R \equiv R\partial I/\partial R$.

If the Lambertian flag is not set, then we will specify inputs for a surface with a bidirectional reflecting distribution function (BRDF). Following the methods summarized above in section 2.3, we specify the BRDF to be a linear combination of up to three BRDF kernel functions:

$$\rho(\theta,\theta',\phi-\phi') = \sum_{k=1}^{3} a_k f_k(\theta,\theta',\phi-\phi';\mathbf{b}_k)$$

Here, (θ,ϕ) indicates the pair of incident polar and azimuth angles, with the prime indicating the reflected angles. The a_k are linear combination coefficients, while the kernels $f_k(\theta,\theta',\phi-\phi';\mathbf{b}_k)$ 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 of a kernel function is the Cox-Munk BRDF which is a combination of a wave-facet probability distribution function (depending on wind-speed W), and a Fresnel reflection function (depending on relative refractive index $m_{\rm rel}$). In this case the vector $\mathbf{b}_k$ has two elements: $\mathbf{b}_k = \{W, m_{\rm rel}\}$. [The maximum dimension of these parameter vectors is 3].

Note that for a Lambertian surface, there is one kernel $f_1 \equiv 1$ for all incident and reflected angles and coefficient a_1 is just the Lambertian albedo. LIDORT 3.0 has 9 possible kernel functions, and these are listed in Table 2 below along with the references; the user must choose up to three from this list. A full discussion of these kernel types is given in [Spurr, 2004].

Table 2. The BRDF kernel functions in LIDORT Version 3.0

Index	Name	Size $\mathbf{b}_k$
1	Lambertian	0
2	Ross thick	0
3	Ross thin	0
4	Li parse	2
5	Li dense	2
6	Roujean	0
7	Hapke	3
8	Rahman	3
9	Cox-Munk	2
10	Snow (GISS)	Tbd
11	Soil (GISS)	Tbd

Thus for BRDF input, it is necessary for the user to specify up to three combination coefficients $\{a_k\}$ associated with the choice of kernel functions, and the corresponding vectors $\{\mathbf{b}_k\}$. 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.

Note that we do not need to specify full tables of BRDF values for each Fourier component. LIDORT has internal routines for calculating values of the kernel functions for all possible combinations of angles, and additional BRDF routines for delivering the Fourier components of the kernel functions. As noted in section 2.3, 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 $N_{\rm BRDF}$. The choice $N_{\rm BRDF} = 25$ is sufficient to obtain numerical accuracy of 10^{-4} in this Fourier component calculation. Nonetheless, the user is allowed to choose $N_{\rm BRDF}$.

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

5. The LIDORT 3.0 package

5.1. Directory Structure

The directory structure is summarized in Figure 1. From the parent directory, there are 4 main subdirectories. The "Lv3_environment" subdirectory contains the calling program for LIDORT, the 'makefile', all executables and shell scripts, input configuration files to read control options and any user-defined atmospheric setups (data files) containing optical property inputs. Object files for the LIDORT code are stored in a separate directory for clarity (the 'makefile' ensures this is done).

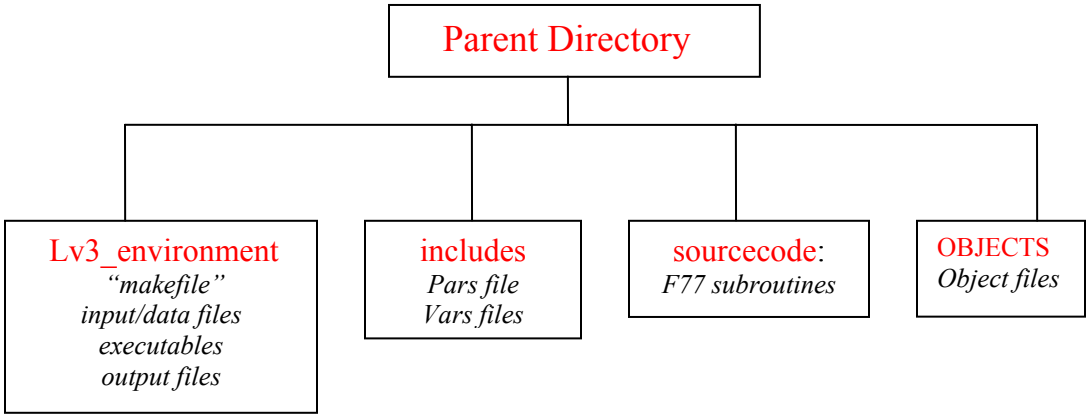

Figure 1. Directory structure for the LIDORT 3.0 installation package.

We describe here the Fortran 77 directory configuration (the Fortran 90 version of LIDORT 3.0 is currently under construction but the package will have a similar directory structure). The LIDORT code is contained in the two subdirectories "`includes`" and "`sourcecode`".

The directory "`includes`" contains only files with variable and parameter declarations. There is one file with parameter statements – this is LIDORT.PARS. All other files have the ".VARS" extension, and they contain declarations of variables stored and saved in common blocks. There are no executable statements in any of these files.

5.2. The `includes` directory

Table 3 lists all the files in this directory (16 in all). LIDORT.PARS contains all symbolic dimensioning parameters (integers), plus a number of fixed constants and numbers. Variables in the "VARS" files are grouped according to function, and all arrays are declared using the dimensioning specified in LIDORT.PARS; hence an include statement for LIDORT.PARS must precede all include statements for the variable ".VARS" files.

If additional memory is required for example for an application requiring a large number of layers or a large number of discrete ordinates, it is only necessary to go into the LIDORT.PARS

files in order to change the dimensioning parameters. Re-compilation with the "makefile" is then carried out to build the executable.

[In the Fortran 90 version, 'include' files will be replaced by modules, which will contain explicit fixed dimensioning of variables (no allocatable memory), and there will be no common blocks (this storage function is replaced in Fortran 90 by the "`USE, ONLY`" construct).]

Table 3. Files in the Includes directory

Include file Name	Function
LIDORT.PARS	Constants, Dimensions
LIDORT_INPUTS.VARS	Inputs (for Radiances)
LIDORT_L_INPUTS.VARS	Inputs (for Jacobians)
LIDORT_RESULTS.VARS	Output (Radiances)
LIDORT_L_RESULTS.VARS	Output (Jacobians)
LIDORT_BOOKKEEP.VARS	Internal, derived inputs
LIDORT_MULTIPLIERS.VARS	Internal, radiances
LIDORT_SOLUTION.VARS	Internal, radiances
LIDORT_REFLECTANCE.VARS	Internal, radiances
LIDORT_SINGSCAT.VARS	Internal, radiances
LIDORT_SETUPS.VARS	Internal, radiances
LIDORT_L_MULTIPLIERS.VARS	Internal (Jacobians)
LIDORT_L_SOLUTION.VARS	Internal (Jacobians)
LIDORT_L_REFLECTANCE.VARS	Internal (Jacobians)
LIDORT_L_SINGSCAT.VARS	Internal (Jacobians)
LIDORT_L_SETUPS.VARS	Internal (Jacobians)

There are 2 files for input variables, and two for output results. The remaining files are for the internal storage of variables and would not need to be looked at by the user. Below, we will go through the PARS files and the input and output files, and merely summarize the internal include files.

5.2.1 Parameter file LIDORT.PARS

This parameter file must be declared in every module in which LIDORT variables will be declared (this includes the environment program). 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 phase function moments and sufficient quadrature streams (discrete ordinates), so MAXSTREAMS and MAXMOMENTS should be increased in value. All other dimensioning parameters are combinations of this basic set, and are not described here. See Table A1 in Appendix 7.3.

In addition to the basic dimensioning parameters, LIDORT.PARS also contains fixed numbers such as π, 0.0, 1.0, etc. some fixed character strings used for output formatting, and some file

output numbers. Some of the critical physics numbers are given above. 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 used to indicate the error status for the package or any part of it:

LIDORT_SUCCESS	= 0 (status index for a successful execution, with no log-output).
LIDORT_DEBUG	= 1 (status for successful execution, with debug log-output).
LIDORT_INFO	= 2 (status for successful execution, with information log-output).
LIDORT_WARNING	= 3 (status for successful execution, with warning log-output).
LIDORT_SERIOUS	= 4 (status index for an aborted execution, with failure log-output).

If the output is not completely successful in any way (status not equal to LIDORT_SUCCESS), then information will be written to a log file, according to the severity of the exception. The "Warning" status has been introduced in Version 3.0 to deal with incorrect user-defined input values that can be re-set internally to allow the program to complete; a warning is written to the Log file. The debug status will be set automatically if the debug flag is set.

There is an additional set of indices for the BRDF. The BRDF kernel index names are explicitly named after the kernel types, and take the values indicated in the first column of Table 2 in section 4.4 above.

5.2.2 Input file LIDORT_INPUTS.VARS

This file contains all the input variables that must be specified by the user in order for LIDORT to carry out a radiance calculation (additional inputs for Jacobian production are discussed next). All variables may be set either by writing explicitly coded statements in the calling program, or by reading entries from an ASCII-type configuration file. Where appropriate, all variables are checked for consistency inside the LIDORT package, before execution of the main radiative transfer modules. The variables are divided into 4 groups: (A) high-level control flags; (B) control integers and user-defined output options; (C) atmospheric inputs, including all optical properties, and (D) surface inputs of all types. These groups are listed in Tables A2-A5 in Appendix 7.3.

Group (A) contains only Boolean flags and character strings. Group (B) contains major variables controlling the execution. In addition to such key numbers as NSTREAMS and NLAYERS (see below), this group contains all the geometrical input quantities (solar zenith angles, user-defined azimuth angles and off-quadrature zenith angles), as well as optical depth input.

Group (C) had all optical property variables required for input to the LIDORT package. It includes the phase function coefficients in each layer, total single scattering albedos in each layer, and layer optical depth coordinates. In Chapter 4 we discussed the creation of these optical property inputs for a simple example. We also have the height, pressure and temperature grids (required for the pseudo-spherical Chapman factor calculations).

Group (D) contains all surface information (Lambertian albedo, BRDF flags and parameter settings, surface emission Planck function value). The option for surface emissivity is currently disabled, and the inputs do not appear in the table.

5.2.3 Input file LIDORT_L_INPUTS.VARS

This file contains all the *additional* input variables that must be specified by the user in order for LIDORT to carry out a Jacobian calculation. As before, all variables may be set either by writing explicitly coded statements in the calling program, or by reading entries from an ASCII-type configuration file. Table A6 in Appendix 7.3 specifies all this input.

5.2.4 Output file LIDORT_RESULTS.VARS

This file contains all Radiance output variables divided into four groups - (i) the complete set of Fourier-summed intensities and the single Fourier-term values of the intensities; (ii) the mean-value output; (iii) layer-integrated multiple scatter source term output (Fourier component and summed values); and (iv) direct and diffuse source terms at the lower boundary (Fourier and summed). *The latter two groups should only be used in conjunction with the flag SAVE_LAYER_MSST*. The Fourier component terms are local to the Fourier loop inside the master LIDORT modules, and they are not saved. Fourier component results may be printed to file (optional debug), and will then be overwritten after the next Fourier component is calculated. See Table A7 in Appendix 7.3.

Table 4. Internal include files in the Includes directory

LIDORT_BOOKKEEP.VARS	Include file with derived inputs. These are created at an initial stage from the inputs and used repeatedly throughout the code.
LIDORT_MULTIPLIERS.VARS	Arrays required for the post-processing of the RT solution at arbitrary stream angles and optical depths.
LIDORT_SETUPS.VARS	Storage for local geophysical variables, all optical depth exponential transmittances, all Legendre polynomials.
LIDORT_SOLUTION.VARS	Homogeneous solution eigensolutions, particular integral solution vectors, and the matrices and column vectors for the boundary value problem.
LIDORT_SINGSCAT.VARS	Variables computed in the exact single scatter and direct beam modules.
LIDORT_REFLECTANCE.VARS	Storage for surface reflectance and BRDF variables.
LIDORT_L_MULTIPLIERS.VARS	Linearizations of the multiplier variables
LIDORT_L_SETUPS.VARS	Linearizations of the setup variables
LIDORT_L_SOLUTION.VARS	Linearizations
LIDORT_L_SINGSCAT.VARS	Linearizations
LIDORT_L_REFLECTANCE.VARS	Linearizations.

5.2.5 Output file LIDORT_L_RESULTS.VARS

This file contains all Linearization output variables divided into the same four groups - (i) the complete set of Fourier-summed weighting functions and the single Fourier-term values; (ii) weighting functions for the mean-value output; (iii) linearizations of the layer-integrated multiple scatter source term output (Fourier component and summed values); and (iv) linearizations of the direct and diffuse source terms at the lower boundary (Fourier and summed). As before, local

Fourier component terms are not saved but may be printed to file (optional debug). See Table A8 in Appendix 7.3.

5.2.5 Internal include files

This group is briefly listed in Table 4. There are 11 files in all. There are 5 files with the main calculation variables ("setups", "solution", "multipliers", "reflectnace" and "singscat"), and 5 additional files with the corresponding linearization variables ("_L" is added to the names, as indicated below in Table 4. In addition, there is a single file with bookkeeping information.

5.3. Sourcecode files

The listing of source code files is given next. They are grouped into two main classes: 17 files required for the Radiance calculation, and an additional 16 files for the production of Jacobians.

5.3.1 Source code files for Radiance calculations

LIDORT_MASTER.f.
This is the main module for the production of radiances, to be called from the environment.

LIDORT_FOURIER_MASTER.f.
This is the master module for calculation of a single Fourier component of the intensity output. It is called directly in LIDORT _MASTER as part of the Fourier loop,

LIDORT_INPUTS.f.
All input functions – initializes inputs and reads them from file, checks the inputs for mistakes and consistency, sets derived input variables for bookkeeping (for example, sorting the stream angles input, sorting and assigning masks for optical depth output).

LIDORT_MISCSETUPS.f.
Only called for Fourier component $m = 0$. This includes a number of set-up operations including the Delta-M scaling and the preparation of all optical depth exponentials that can be pre-calculated. This also includes the Chapman function calculation fore the curved atmosphere.

LIDORT_SOLUTION.f
Solution of the discrete ordinate radiative transfer equation. Returns the eigensolutions and separation constants from the homogeneous equation, plus the particular (beam) solution vectors for both the classical solution technique and the Green function method.

LIDORT_BVPROBLEM.f
Set-up and solution of the boundary-value problem (constants of integration) in a multi-layer atmosphere. This requires the L-U decomposition (matrix inversion). Also contains the modules dealing with boundary value telescoping.

LIDORT_INTENSITY.f
Computation of intensities at user-defined optical depths and stream angles; this is the post-processing (source function integration).

LIDORT_MULTIPLIERS.f
Computation of multipliers required for the layer source terms that form the heart of the source function integration technique.

LIDORT_CONVERGE.f
Examines convergence of Fourier series for all intensities; upgrades the intensity Fourier series.

LIDORT_BRDFMODULES.f
This module includes all BRDF calculations, including the kernel functions themselves and the determination of Fourier components of the bidirectional reflectance.

LIDORT_CORRECTIONS.f
If flagged, these modules calculate the exact Nakajima-Tanaka single scatter intensity and the exact direct beam intensity.

LIDORT_WRITEMODULES.f
This contains 4 modules for writing outputs to file, respectively full *intensity* results, Fourier components of intensity, a scenario description and a summary of the input data (corresponding to the write control flags set in LIDORT_INPUTS.VARS).

LIDORT_AUX.f
Standard numerical routines for eigenproblem solution (ASYMTX as used in DISORT), and linear algebra modules (LAPACK band storage and L-U decomposition modules). Legendre polynomial and Gauss quadrature evaluation. Includes the input file-read tool FINDPAR. Includes also error tracing modules.

5.3.2 Source code files for Radiance and Jacobian calculations

LIDORT_L_MASTER.f.
This is the main module to be called from the environment.

LIDORT_L_FOURIER_MASTER.f.
This is the master module for calculation of a single Fourier component of the intensity + Jacobian output output. It is called directly in LIDORT _L_MASTER as part of the Fourier loop,

LIDORT_L_INPUTS.f.
All input functions for linearization variables – initializes and reads them from file, checks the inputs for mistakes and consistency.

LIDORT_L_MISCSETUPS.f.
Only called for Fourier component $m = 0$. This includes set-up operations including the Delta-M scaling on optical property derivative inputs, and linearization of all optical depth exponentials that can be pre-calculated..

LIDORT_L_SOLUTION.f

Determines linearizations (with respect to atmospheric variables) of the homogeneous and particular (beam) solution vectors for both the classical solution technique and the Green function methods.

LIDORT_L_BVPROBLEM.f
Solution of 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 the modules dealing with linearization boundary value telescoping.

LIDORT_L_WFATMOS.f
Computation of weighting functions with respect to atmospheric properties at user-defined optical depths and stream angles; this is the post-processing (source function integration).

LIDORT_L_WFSURFACE.f
Computation of weighting functions with respect to surface properties at user-defined optical depths and stream angles; this is the post-processing (source function integration).

LIDORT_L_MULTIPLIERS.f
Computation of linearized multipliers.

LIDORT_L_CONVERGE.f
Adds the weighting function Fourier components to the total..

LIDORT_L_BRDFMODULES.f
This module includes all linearizations of the BRDF calculations, including the derivatives of the kernel functions with respect to surface parameters and the determination of Fourier components of the linearizations of the bidirectional reflectance.

LIDORT_L_CORRECTIONS.f
If flagged, these modules calculate the linearizations of the exact Nakajima-Tanaka single scatter intensity and the exact direct beam intensity. The N-T linearization is only for atmospheric variables, whereas the direct beam calculation may have atmospheric and surface linearizations.

LIDORT_L_WRITEMODULES.f
This contains 4 modules for writing *additional* outputs to file, respectively full *Jacobian* results, Fourier components of jacobians, and additional scenario descriptions and inputs.

5.4. Makefile, installation and the calling environment

5.4.1 Calling Environment – an example

We show how the master LIDORT module is used within a calling environment, by means of a simple example in the form of a schematic computational sequence (pseudo-code). Please refer to Appendix 7.1. [We use the "Courier" Font to denote text copied from software code. Comment lines are prefaced by the symbol "/*".]

LIDORT execution is controlled by a single module LIDORT_MASTER, which is called once for each wavelength (LIDORT is monochromatic). Note that within the wavelength loop, the call to the master module is preceded by the user-defined preparation module, which assigns values to Group (D) inputs in LIDORT_INPUTS.VARS, and Group (B) inputs in LIDORT_L_INPUTS.VARS.

The main call is preceded by a call to LIDORT_INPUT_MASTER master module to read the appropriate input from the *configuration file* LIDORT.INP (passed as a subroutine argument). If the STATUS_INPUTREAD integer output is not equal to LIDORT_SUCCESS, the program should stop and the user should examine the error file. *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. However, this requires a certain level of confidence in the model!*

In Appendix 7.2 we give an example of a typical configuration file. The file is read using the FINDPAR tool - this module looks for a character string comprising a prefix (in this case the word "LIDORT") and a text description of the variable 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. The file-read is divided into four parts corresponding to Groups (A) (B) and (D) variables in LIDORT_INPUTS.VARS, and Group (A) variables in LIDORT_L_INPUTS.VARS.

The subroutine output STATUS_INPUTCHECK is available for the checking of the input data once the file-read is complete. Checking is internal to LIDORT and is done first before any radiative transfer. If this integer output is equal to LIDORT_SERIOUS, LIDORT will exit without performing any calculations; if it equals LIDORT_WARNING, the model will execute but it means that some of the input is incorrect and that LIDORT has reverted to a default input and carried on with this default. The checking routine will write messages and traces to file. In either case (fatal or warning error) the user should examine the error (log) output file, which will be called LIDORT_OUTPUT.LOG unless the user specifies a name for this.

If there is a fatal error during the execution of LIDORT, then the model will bypass any further calculation and exit after an error message (and an error trace to indicate the source of the error) has been written to the Log file. In this case, the STATUS_CALCULATION integer output will have the value LIDORT_SERIOUS.. There are no warnings here, all errors in execution are fatal.

In most cases, it is sufficient to call LIDORT_INPUT_MASTER just once before a wavelength loop, though some of the input options may be changed before each call to LIDORT_MASTER. The output for some wavelengths may be suppressed (note that the output write files in LIDORT have "unknown" status – they are always overwritten).

5.4.2 Makefile – an example

We now discuss the *Makefile* in the lv3_environment subdirectory. Please refer to Appendix 7.3, which contains part of the Makefile, including the compilation flags, path variables and executable "build" statements. We leave out details of the compilations. The GNU compiler options shown here are designed to pick up any undeclared or unused variables. All LIDORT source code is Implicit None. Note in particular the "path" definitions – this is the place to change to your own home directory, and it is only necessary to change one of these statements,

as the routeing to the other directories is automatic. Note also that all object files are collected in one subdirectory to avoid cluttering up the environment directory.

5.4.3 Installation and testing

For Version 3.0, the installation was set up to work with the Makefile as outlined in the previous section using GNU compilation. The installation test was set up on the RT Solutions Linux PC (Dell 8400). To install, unpack the zipped tar file in a new "LIDORT home" directory. Go into the the "lv3_environment" subdirectory, where you will find the list of files presented in Table 6 below.

Enter the makefile, where you should change the absolute path to the "LIDORT home" directory you have just created. Running the makefile will create two executables "lv3_rad_test.exe" and "lv3_jac_test.exe". These two executables are for a radiance-only test and for a test with Jacobians respectively. There are two calling programs with the same name. Running each program will generate a number of test output files (3 tests for the radiance, 1 for the Jacobian facility), with the same names as those listed in the table, but without the "_save" suffix. When you compare this new output with the supplied test results, the answers should be identical. If this is the case, then the installation has been successful. Descriptions of the tests are given in a dedicated text file, but we give some details here.

Table 5. Files in the lv3_environment directory

Name of file	Description of file
`Lv3_rad_test.f`	Calling program for the radiance tests
`Lv3_jac_test.f`	Calling program for the Jacobian tests
`Lv3_config.inp`	Common configuration file input for all tests
`Clearsky_opdep.dat`	File of atmospheric data for use in tests 3 and 4 (13 layer atmosphere with Rayleigh and aerosol scattering)
`Test_description.tex`	Text file with a descriptions of the tests
`Makefile`	The makefile for building two executables
`Rad_test_1.out_save`	Results of radiance test #1
`Rad_test_2.out_save`	Results of radiance test #2
`Rad_test_3.out_save`	Results of radiance test #3
`Jac_test_4.out_save`	Results of Jacobian test #4
`Lv3_license.tex`	Public license for LIDORT Version 3.0 (see section 5.6)

Test #1. This is the slab problem (single layer) of optical depth 1.0 for a medium with Rayleigh scattering. The number of streams is 6 (half-space), and output is given for 10 solar zenith angles, 2 line-of-sight zenith angles and 2 azimuth angles. The albedo, assumed Lambertian, is 0.2. Output is specified at the top (upwelling, slab reflectance) and bottom (downwelling slab transmittance) boundaries. Plane parallel only, classical solution.

Test #2. This has the same setup as the previous test, except that there are now 4 identical layers, each with optical depth 0.25. The invariance principle tells us that the results should be the same as those from test #1; this is a test of the multi-layer facility.

Test #3. This calculation with a multilayer atmosphere with Rayleigh scattering, some trace gas absorption, and aerosol scattering and extinction. There are 13 layers, 10 streams in the half space, and the same geometrical inputs as those in the previous two tests. Output is specified at all layer boundaries for upwelling and downwelling. The delta-M scaling is applied, along with the exact single scatter calculation. The atmospheric inputs are provided in the file "clearsky_opdep.dat". The albedo is 0.2. Pseudo-spherical calculation using the Greens function formalism for solving the particular integral.

Test #4. This is the Jacobian test, and the setup is identical to that in test #3. Two profile weighting functions are output for each of the 13 layers, with respect to the total optical depth in each layer, and the total single scattering albedo in each layer (optical property inputs are easy to define for this case). We also output the albedo weighting function.

5.5. Error Handling and Utilities

5.5.1 Error handling

The master modules LIDORT_MASTER and LIDORT_L_MASTER have two status arguments: these are the integer variables STATUS_CHECKINPUT and STATUS_CALCULATION (see the example in Appedix 7.1). These integers can take one of the5 values indicated in the LIDORT.PARS structure (see section 5.2.1 above). Internal exception handling in LIDORT works as follows. If an error has been returned from one of the internal routines, then a message about the error is generated along with a trace for that error; these quantities are then written to the output error file LIDORT_LOGOUTPUT.DAT which will be generated automatically and opened in the Environment directory if these is any sort of error or if any of the debug or information options have been set.. If an internal error has occurred this will be traced through LIDORT and an error value will be given to one of the above two STATUS variables upon output from the master module.

STATUS_CHECKINPUT refers to the status of the inputs. If there is a fatal error in the input file-read (e.g. incorrect key words in the configuration file, detected by FINDPAR) or in the input checking, LIDORT will exit without any further calculation and the error messages written to file LIDORT_LOGOUTPUT.DAT. Most of the LIDORT inputs are checked internally before actual computations start. Errors produced in this way will be due to incorrect or inconsistent input specifications. In addition to each error message and trace, an additional character string is generated for the error file - this string provides a hint for the user to correct the appropriate input. Not all checking errors will be fatal. If there is a warning error, LIDORT will continue execution, but warning messages concerning the input will be generated and written to the Log-output file. If this occurs, LIDORT has actually corrected your input and gone on with an execution. As noted, the message output will contain hints about correcting the input.

STATUS_CALCULATION refers to the status of the radiative transfer calculation. Apart from the use of standard numerical routines to solve the eigensystem and a number of linear algebra problems, LIDORT is entirely analytical. Only in exceptional circumstances should an error condition be returned from the eigenroutine ASYMTX 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 invariable produced by bad optical property input that has somehow escaped the initial check.

Note on the use of L'Hopital's Rule. Occasionally, analytic expressions used in LIDORT break down when there is a coincidence of stream angles. The commonest example of this occurs for downwelling radiation in a plane-parallel atmosphere when one of the user stream angles is *identical* to the solar zenith angle. The corresponding layer source term multiplier must be replaced by a limit value found by the use of L'Hopital's Rule. The example noted here has already been dealt with in internally, but similar breakdowns could in theory happen say when one of the separation constants is vanishingly close to one of the user stream angles.

5.5.2 Utilities

All software in LIDORT was written by R. Spurr, with the exception of a number of utility routines taken from standard sources. All LIDORT utility routines are collected together in the source code file LIDORT_AUX.f. They include a selection of modules from the LAPACK library, plus a number of other standard numerical modules, and some file-read and error handling modules. The most important modules in the LAPACK selection are DGBTRF, DGBTRS, DGETRF, DGETRS, DGBTF2, DLASWP, XERBLA, DGETF2, DGEMM, DGEMV, DGER, DTBSV, and DTRSM. These LAPACK modules are not performance-optimized for the LIDORT package (there is in particular a lot of redundancy in the linear algebra problems). It is expected that the whole linear algebra package will be replaced by dedicated routines based in part on the above, but with enhanced performance in terms of run-time and efficiency.

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

5.6. Copyright Issues: GNU License

R. Spurr developed the original LIDORT model at the Smithsonian Astrophysical Observatory (SAO) over the years 1999-2004. Version 2.3 was given public release in 2002 and is now in use in a large number of remote sensing applications. All software generated at (SAO) is in the public domain. For continuity, it is desirable that newer LIDORT versions should also remain in the public domain, and the following copyright remarks will apply to any software distributed by RT Solutions, Inc.

Permission to use, copy, modify, and distribute any LIDORT software developed by RT Solutions Inc., any documentation appertaining to the LIDORT software and any results obtained using LIDORT 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 LIDORT SOFTWARE IN VERSION 3.0 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 LIDORT PROGRAM VERSION 3.0 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 LIDORT SOFTWARE IN VERSION 3.0.

5.7. Acknowledgments

All software for LIDORT, with the sole exception of certain utility modules in the source code file “LIDORT_AUX.f”, was written specifically by R. Spurr.

For software support, the author would like to thank Thomas Kurosu of SAO (Legendre polynomial module, makefile support), Werner Thomas of DFD (makefile support), Eberhard Mikusch of DLR, John Lavagnino formerly of SAO for the FINDPAR routines, and Knut Stamnes for the use of ASYMTX. The author would like to thank co-authors Thomas Kurosu and Kelly Chance of SAO for support with the initial work on Version 1.0. Roeland van Oss of KNMI provided much input and discussion for the Version 2.0 work on LIDORT.

User feedback has been especially helpful in ironing out bugs and providing useful hints for the usage. In this regard, the author would like to thank European colleagues Roeland van Oss, Piet Stammes, Johan de Haan, Diego Loyola, Werner Thomas, Michel van Roozendael and Jukka Kujanpaa, and in America, Knut Stamnes, Nick Krotkov, Mick Christi and Vijay Natraj.

Funding for the LIDORT work has come from a number of sources. The main one has been a series of Ozone SAF Visiting Scientist Grants from the Finnish Meteorological Institute, spread over the 5-year period 2000-2004. Funding has also come from internal money at SAO (1999), two grants from the European Space Agency to work on GOME total ozone retrieval algorithms (2003, 2004), and grants from NASA to work on LIDORT applications (2003, 2005). The amalgamation of previous LIDORT versions and the transition to Version 3.0 has been assisted in late 2005 with a new grant from NASA.

6. References

[1] Spurr, R.J.D., T.P. Kurosu, and K.V. Chance, A linearized discrete ordinate radiative transfer model for atmospheric remote sensing retrieval, *JQSRT*, **68**, 689-735, 2001.

[2] Spurr, R.J.D., Simultaneous derivation of intensities and weighting functions in a general pseudo-spherical discrete ordinate radiative transfer treatment, *JQSRT*, **75**, 129-175, 2002.

[3] 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, *JQSRT*, **75**, 177-220, 2002.

[4] 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.

[5] Spurr, R.J.D., A New Approach to the Retrieval of Surface Properties from Earthshine Measurements, *JQSRT*, **83**, 15-46, 2004.

[7] 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).

7. Appendices

7.1. Configuration file example

```
main LIDORT

/* Core fixed parameter file
include 'LIDORT.PARS'

/* Core input files
include 'LIDORT_INPUTS.VARS'
include 'LIDORT_L_INPUTS.VARS'

/* Core output files
include 'LIDORT_RESULTS.VARS'
include 'LIDORT_L_RESULTS.VARS'

/* status declarations
INTEGER STATUS_INPUTREAD, STATUS_INPUTCHECK, STATUS_CALCULATION

/* Define output in the way you want!
Declare output arrays

/* File-read variables in the input structures
call LIDORT_INPUT ('LIDORT_FILE.INP',STATUS_INPUTREAD)
if (STATUS_INPUTREAD = LIDORT_SERIOUS) write message & abort
if (STATUS_INPUTREAD = LIDORT_WARNING) write message & continue

/* Start wavelength loop
for i = 1, n_user_wavelengths, begin
/* Assign variables in LIDORT_INPUTS.VARS:
     call USER_LIDORT_PREPARE
/* LIDORT master call and error check
     call LIDORT_L_MASTER(STATUS_INPUTCHECK, STATUS_CALCULATION)
     call LIDORT_STATUS(STATUS_INPUTCHECK, STATUS_CALCULATION)
     Copy LIDORT output to user-defined output arrays
/* End wavelength loop
end for

/* finish
write user-defined output arrays
stop and end of program
```

7.2. Configuration file example

```
**** PART 1. GROUP (A) Variables in LIDORT_INPUTS.VARS ****

LIDORT - Do single scatter correction?
.TRUE.
LIDORT - Do direct beam correction?
.TRUE.
LIDORT - Output multiple scatter layer source functions?
.FALSE.
LIDORT - Perform double convergence test?
.TRUE.

LIDORT - Use classical beam solution?
.FALSE.
LIDORT – Plane-parallel treatment of direct beam?
.FALSE.
LIDORT - Perform internal Chapman function calculation?
.TRUE.
LIDORT - Beam path with refractive atmosphere?
.TRUE.

LIDORT - Rayleigh atmosphere only?
.FALSE.
LIDORT - Isotropic atmosphere only?
.FALSE.
LIDORT - No azimuth dependence in the solution?
.TRUE.
LIDORT - Compute all Fourier components?
.FALSE.

LIDORT - Include Delta-M scaling?
.TRUE.
LIDORT – Use solution saving?
.TRUE.
LIDORT – Use boundary value telescoping?
.TRUE.


LIDORT - Debug write?
.FALSE.
LIDORT - Input control write?
.TRUE.
LIDORT - Input scenario write?
.TRUE.
LIDORT - Fourier component output write?
.FALSE.
LIDORT - Results write?
.TRUE.

LIDORT - filename for input write
idort_myproblem.input
LIDORT - filename for scenario write
LIDORT_myproblem.scenario
LIDORT - Fourier output filename
LIDORT_myproblem.fourier
LIDORT - filename for main output
LIDORT_myproblem.result


LIDORT - Upwelling output?
.TRUE.
```

```
LIDORT - Downwelling output?
.TRUE.

LIDORT - Include quadrature angles in output?
.TRUE.
LIDORT - User-defined stream angles?
.TRUE.
LIDORT - User-defined optical depths?
.FALSE.
LIDORT - Layer boundary optical depths?
.TRUE.

LIDORT - Generate mean-value output additionally?
.TRUE.
LIDORT - Generate only mean-value output?
.FALSE.
```

**** PART 3. Group (C) Variables in LIDORT_INPUTS.VARS ****

```
LIDORT - Lambertian albedo?
.TRUE.

LIDORT - Value of Lambertian albedo
0.2

LIDORT - Include surface emission?
.FALSE.

LIDORT - Number of BRDF kernels
3
LIDORT - Number of BRDF azimuths
25
LIDORT - Formatted BRDF kernel inputs
```

**** Part 3. Group (B) Variables in LIDORT_INPUTS.VARS ****

```
LIDORT - Number of half-space streams
20
LIDORT - Number of atmospheric layers
22
LIDORT - Number of input Legendre moments
109

LIDORT - Fourier series convergence
0.001
LIDORT - Zenith tolerance level
0.0001

LIDORT - Earth radius (km)
6371.0
LIDORT - Refractive index parameter
0.000288

LIDORT - Solar flux constant
1.0
LIDORT - Number of solar zenith angles
3
LIDORT - Solar zenith angle inputs (degrees)
65.0
75.0
85.0
```

```
LIDORT - Number of user-defined relative azimuth angles
2
LIDORT - User-defined relative azimuth angles
60.0
50.0

LIDORT - Number of user-defined stream angles
3
LIDORT - User-defined stream angles
0.0
15.0
30.0

LIDORT - Number of user-defined optical depths
1
LIDORT - User-defined optical depths
0.2065
LIDORT - Number of layer boundary optical depths
2
LIDORT - Which layer boundaries for optical depth
0
1
```

7.3.Makefile example

```
################################################################## Makefile for LIDORT

# Gnu compiler instructions and macro
   LIDORT_COMPILE = g77 -c -O2 -Wimplicit -Wunused
   F77    = g77
   LINK.f = $(F77)

# Define a variable LIDORT_PATH and users must change only this!

   LIDORT_PATH = /home/myhome/LIDORT
   PATH    = $(LIDORT_PATH)/Lv3_environment/
   SPATH_S = $(LIDORT_PATH)/sourcecode/
   IPATH_S = $(LIDORT_PATH)/includes/
   OBJ     = $(LIDORT_PATH)/OBJECTS

#################################################################### OBJECT MODULES

OBJECTS_LIDORT_MASTERS =      $(OBJ)/lidort_master.o \
                              $(OBJ)/lidort_input_master.o \
                              $(OBJ)/lidort_brdf_master.o \
                              $(OBJ)/lidort_fourier_master.o
OBJECTS_LIDORT_SOLUTIONS =    $(OBJ)/lidort_solution.o \
                              $(OBJ)/lidort_multipliers.o \
                              $(OBJ)/lidort_bvproblem.o
OBJECTS_LIDORT_SETUPS =       $(OBJ)/lidort_miscsetups.o \
                              $(OBJ)/lidort_inputs.o \
                              $(OBJ)/lidort_chapman.o \
                              $(OBJ)/lidort_legendre.o
OBJECTS_LIDORT_INTENSITY =    $(OBJ)/lidort_intensity.o \
                              $(OBJ)/lidort_corrections.o \
                              $(OBJ)/lidort_converge.o
OBJECTS_LIDORT_AUX =          $(OBJ)/lidort_aux.o \
                              $(OBJ)/lidort_inputread.o \
                              $(OBJ)/lidort_writemodules.o
OBJECTS_LIDORT_BRDF =         $(OBJ)/lidort_brdf.o \
                              $(OBJ)/lidort_brdf_kernels.o \
OBJECTS_LIDORT_ENVIRON =      $(OBJ)/lv3_rad_test.o

#################################################################### EXECUTABLE

APPLICATION = lidort_rad.exe

all: $(APPLICATION)
$(APPLICATION):               $(OBJECTS_LIDORT_ENVIRON)   \
                              $(OBJECTS_LIDORT_MASTERS)   \
                              $(OBJECTS_LIDORT_SETUPS)    \
                              $(OBJECTS_LIDORT_SOLUTIONS) \
                              $(OBJECTS_LIDORT_INTENSITY) \
                              $(OBJECTS_LIDORT_AUX)       \
                              $(OBJECTS_LIDORT_BRDF)
$(LINK.f) -o $(APPLICATION) \
                              $(OBJECTS_LIDORT_ENVIRON)   \
                              $(OBJECTS_LIDORT_MASTERS)   \
                              $(OBJECTS_LIDORT_SETUPS)    \
                              $(OBJECTS_LIDORT_SOLUTIONS) \
                              $(OBJECTS_LIDORT_INTENSITY) \
                              $(OBJECTS_LIDORT_BRDF)      \
                              $(OBJECTS_LIDORT_AUX)       \
                              $(FLIBRARIES)

#################################################################### COMPILATIONS

Etc…
```

7.4. Tables for Input/Output variables

Table A1. Key parameters and dimensions in LIDORT.PARS

Name	Type	Description
MAXBEAMS	Dimension	Maximum number of solar zenith angles
MAXSTREAMS	Dimension	Maximum number of half-space *quadrature* streams.
MAXLAYERS	Dimension	Maximum number of layers in the atmosphere.
MAXMOMENTS	Dimension	Maximum number of Legendre expansion coefficients. This should be at least twice MAXSTREAMS.
MAXFOURIER	Dimension	Maximum number of allowed terms in the Fourier cosine azimuth series. At least twice MAXSTREAMS.
MAX_DIRECTIONS	Dimension	Maximum number of directions (2), up/down
MAX_USER_STREAMS	Dimension	Maximum number of user-defined *off-quadrature* stream angles
MAX_USER_RELAZMS		Maximum number of user-defined relative azimuth angles
MAX_OUT_USERTAUS	Dimension	Maximum number of user-defined optical depths
MAX_OFFGRID_USERTAUS	Dimension	Maximum allowed number of *off-grid* optical depths at which output is required. This number should always be less than MAX_OUT_USERTAUS.
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^{-6}
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.

Table A2: Group (A) variables in LIDORT_INPUT.VARS

Name	Type	Description
DO_CLASSICAL_SOLUTION	L	Flag for using the Chandrasekhar solution of the particular integral for the beam source. If not set, the Green's function solution is used.
DO_PLANE_PARALLEL	L	Flag for use of the plane-parallel approximation for the direct beam attenuation. If not set, the atmosphere will be pseudo-spherical.
DO_REFRACTIVE_GEOMETRY	L	Flag for using refractive geometry input in the pseudo-spherical approximation.
DO_CHAPMAN_FUNCTION	L	Flag for making an internal calculation of the slant path optical depths DELTA_SLANT_INPUT. If called, must specify height grid and earth radius (and pressure and temperature grids for refractive geometry).
DO_RAYLEIGH_ONLY	L	Flag for simulations in a Rayleigh atmosphere (molecules + trace gas absorptions). If set, only Fourier terms m=0,1,2 are calculated.
DO_ISOTROPIC_ONLY	L	Flag for simulations in an isotropically scattering atmosphere. If set, then only the Fourier m=0 (azimuth independent) term is computed DO_NO_AZIMUTH must be set "true". (Checked internally).
DO_LAMBERTIAN_ALBEDO	L	Flag for using the Lambertian surface reflection condition at the lower boundary. If "false", boundary is assumed fully bidirectional.
DO_NO_AZIMUTH	L	Flag for controlling inclusion of azimuth dependence in the output. If set, then only Fourier m=0 (azimuth-independent) term is calculated.
DO_ALL_FOURIER	L	Flag controlling calculation of Fourier azimuth terms. If set, LIDORT will calculate all Fourier components, regardless of any convergence criterion. If not set, usual convergence criterion will apply. Debug only.
DO_DELTAM_SCALING	L	Flag for controlling use of the Delta-M scaling option. In most circumstances, this flag will be set.
DO_ADDITIONAL_MVOUT	L	Flag to produce integrated (mean-value) output *in addition* to radiance.
DO_MVOUT_ONLY	L	Flag to generate mean-value output only. Since such outputs are hemisphere-integrated, there is no need for user-defined angles, and only the Fourier m=0 contributes.
SAVE_LAYER_MSST	L	Flag to produce layer-integrated multiple scatter source term output *in addition* to total radiances and Jacobians. This output is restricted to whole layers and can only be generated if flag DO_LBOUND_TAUS (see below) is set at the same time. (Output is also restricted to off-quadrature angles). *This is a specialist option and should be used only for enhanced sphericity corrections in LIDORT 3.0+.*.
DO_FULLRAD_MODE	L	If set, LIDORT will give a full calculation. The only time this flag should not be set is when SAVE_LAYER_MSST is set.
DO_SSCORRECTION	L	If set, LIDORT performs Nakajima-Tanaka single scatter correction. When DO_SSCORRECTION and DO_FULLRAD_MODE are both set, LIDORT does only the multiple-scatter computation.
DO_DBCORRECTION	L	If set, LIDORT will perform the direct beam correction, that is, a full-BRDF direct beam calculation is done in place of a truncated calculation. Fourier convergence is now tested only on the full field.
DO_DOUBLE_CONVTEST	L	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_QUAD_OUTPUT	L	If set, there will be output at the discrete ordinate stream angles. Not normally required, but may be of use in debugging. Cannot be used with the single scatter correction DO_SSCORRECTION.
DO_USER_STREAMS	L	If set, there will be output at a number of off-quadrature zenith angles to be specified by the user. This is the normal case.
DO_USER_TAUS	L	Flag for arbitrary optical depth output. This would normally be used in conjunction with a number of specified optical depth outputs.
DO_LBOUND_TAUS	L	Flag for output at layer boundaries. Exclusive of DO_USER_TAUS

		flag; the two cannot be set together. If you want just layer boundary output (e.g. TOA or BOA), this is the one to use.
DO_UPWELLING, DO_DNWELLING	L	Flag for obtaining upwelling and/or downwelling output. One or both must be set!
DO_DEBUG_INPUT	L	Flag for writing-to-file for debugging. *Do not use.*
DO_WRITE_INPUT	L	Flag for writing-to-file of input variables (Groups A and B). Optional.
DO_WRITE_SCENARIO	L	Flag for writing summary of the atmospheric input to file. Optional.
DO_WRITE_FOURIER	L	Flag for output of Fourier components. Not normally required.
DO_WRITE_RESULTS	L	Flag for writing output to file. This is not mandatory, as LIDORT may be embedded in a wavelength loop and results of each call may be stored and written at a later stage in the test environment.
INPUT_WRITE_FILENAME, SCENARIO_WRITE_FILENAME FOURIER_WRITE_FILENAME RESULTS_WRITE_FILENAME	C	These 4 character strings are for the specification of 4 output files corresponding to the above 4 flags controlling output.
LIDORT_LOG_FILENAME	C	Character string indicating name of log-output file. This file is opened at the first exception encounter and closed before exiting the package.

Table A3: Group (B) variables in LIDORT_INPUT.VARS

.

Name	Type	Description
NBEAMS	I	Number of solar beams.
NSTREAMS	I	Number of quadrature streams in the cosine half space [0,1]. Must be less than or equal to the symbolic dimension MAXSTREAMS.
NLAYERS	I	Number of layers in the atmosphere (NLAYERS = 1 is allowed). Must be less than or equal to the symbolic dimension MAXLAYERS.
NMOMENTS_INPUT	I	Number of Legendre expansion coefficients characterizing the phase function. In the delta-M approximation, this must be at least 2*NSTREAMS to ensure the delta-M truncation factor is meaningful. NMOMENTS_INPUT is used in the single scatter correction, so it should be greater than 2*NSTREAMS−1 in this case. Must be less than MAXMOMENTS.
FLUX_FACTOR	DP	Beam source flux, the same value to be used for all solar angles. Normally set equal to 1 for "sun-normalized" output.
LIDORT_ACCURACY	DP	Accuracy criterion for convergence of Fourier series in relative azimuth. If for each output stream, addition of the m^{th} Fourier harmonic changes the total (Fourier-summed) intensity by less than this value of the total, then we have passed the convergence test once (we may want to do this twice). For each solar angle, convergence is tested for intensities at all output stream angles, optical depths and azimuth angles. Once one solar beam result has converged, there is no further point in calculating any more Fourier terms for this beam, so the inhomogeneous source terms are then dropped after convergence.
SUN_ANGLES	DP	Solar zenith angles (degrees). Checked internally to be in range [0,90).
EARTH_RADIUS	DP	Earth's radius in [km]. Only required when DO_CHAPMAN_FUNCTION has been set. Checked internally to be in range [6320, 6420].
ZENITH_TOLERANCE	DP	Zenith tolerance (nearness of output zenith cosine to 1.0).
RFINDEX_PARAMETER	DP	Only required for DO_REFRACTIVE_GEOMETRY option.
N_USER_STREAMS	I	Number of user-defined polar streams (off-quadrature). Must be not greater than symbolic dimension MAX_USER_STREAMS.
USER_ANGLES_INPUT	DP	Array of user-defined angles (in degrees) for off-quadrature output. The ordering is not important (LIDORT orders and checks this input internally).
N_USER_RELAZMS	I	Number of relative azimuth angles (degrees) for output
USER_RELAZM_INPUT	DP	Array of relative azimuth angles (degrees). Should be between 0 and 180.
N_OUT_USERTAUS	I	Number of optical depth output values. Required if DO_USER_TAUS is set.
USER_TAUS_INPUT	DP	Output optical depth values. These can be in any order (LIDORT sorts them in ascending order internally). Checked to see if any value exceeds the total optical depth of the atmosphere. Repetition of input values is also checked.
LBOUND_TAUS_INPUT	I	Integer values indicating which layer boundaries are required for optical depth output. DO_LBOUND_TAUS should be set. Value 0 indicates top of the atmosphere; the value 1 indicates the bottom boundary of the first layer, etc. Not greater than NLAYERS (this is checked). Repetition is also checked.

Table A4: Group (C) variables in LIDORT_INPUT.VARS

.

Name	Type	Description
OMEGA_TOTAL_INPUT	DP	Single scattering albedos, all layers. Should not equal 1.0. Checked internally – the OMEGA_SMALLNUM toggle will generate a warning.
DELTAU_VERT_INPUT	DP	Vertical optical depth thickness values.
PHASMOMS_TOTAL_INPUT	DP	Legendre moments of the phase function expansion multiplied by $(2l+1)$; the initial value (indexed by 0) should always be 1 (checked).
HEIGHT_GRID	DP	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).
FINEGRID	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.
PRESSURE_GRID	DP	Pressure in [mb] from TOA to BOA. Only required for internal Chapman factor calculation with refractive geometry.
TEMPERATURE_GRID	DP	Temperature in [K] from TOA to BOA. Only required for internal Chapman factor calculation with refractive geometry.
DELTA_SLANT_INPUT	DP	This quantity $\tau(n,k,b)$ is the slant optical thickness in layer k for a solar beam atmospheric path ending at the lower boundary of layer n. This can be calculated by LIDORT's internal Chapman factor routine if the flag is set (recommended). If not, this optical thickness must be supplied.

Table A5: Group (D) variables in LIDORT_INPUT.VARS

Table A6: Variables in LIDORT_L_INPUT.VARS

Name	Type	Description
DO_SIMULATION_ONLY	L	If set, no weighting functions will be calculated
DO_ATMOS_LINEARIZATION	L	Flag for output of profile Jacobians
DO_SURFACE_LINEARIZATION	L	Flag for output of Surface Jacobians
L_OMEGA_TOTAL_INPUT(n,q)	D	Relative variation in the total single scattering albedo in layer *n*, with respect to varying parameter *q* in that layer.
L_DELTAU_VERT_INPUT(n,q)	D	Relative variation in extinction coefficient for layer *n* with respect to varying parameter *q* in that layer.
L_PHASMOMS_TOTAL_INPUT(l,n,q)	D	Relative variation in phase function moment coefficients. For Legendre moment *l* in layer *n* w.r.t. parameter *q* in that layer.
L_DELTAU_SLANT_INPUT(n,k,q)	D	Relative variation in slant path optical depth (must be input if the internal Chapman function is not done)
LAYER_VARY_NUMBER(n)	I	Number of profile weighting functions in layer *n*
N_TOTALATMOS_WFS	I	Maximum value of LAYER_VARY_NUMBER
LAYER_VARY_FLAG(n)	L	Flag for calculating profile weighting functions in layer *n*
N_KERNEL_FACTOR_WFS	I	Number of weighting functions w.r.t. linear combination coefficients in BRDF kernel sum
N_KERNEL_PARAMS_WFS(k)	I	Number of Jacobians for (nonlinear) parameters in kernel *k*
N_TOTALBRDF_WFS	I	Sum of the previous two entries
DO_KERNEL_PARAMS_WFS(k,b)	L	Flags for Jacobians for (nonlinear) parameter *b* in kernel *k*
DO_KERNEL_FACTOR_WFS(k)	L	Flags for weighting functions w.r.t. linear combination coefficient *k* in BRDF kernel sum

Table A7: Variables in LIDORT_RESULTS.VARS

Name	Gp	Description
INTENSITY	1	Total Intensity $\mathbf{I}(t,v,d)$ at any optical depth t, output geometry v direction d
INTENSITY_F	1	Fourier component Intensity $\mathbf{I}_m(t,u,s,d)$ at any optical depth t, output stream angle u, solar beam angle s, and direction d
MEAN_INTENSITY	2	Mean Intensity $\mathbf{J}(t,s,d)$ for optical depth t, solar angle s, direction d
FLUX	2	Flux $\mathbf{F}(t,s,d)$ for optical depth t, solar angle s, direction d.
MSCATSTERM	3	Layer-integrated multiple scatter source term $\mathbf{\Lambda}(n,v,d)$ for layer n, output geometry v and direction d
MSCATSTERM_F	3	Fourier component $\mathbf{\Lambda}_m(n,u,s,d)$ of layer-integrated multiple scatter source term layer n, off-quadrature user angle u, relative azimuth angle a, solar angle s, and direction d
MSCATBOA_SOURCETERM	3	Bottom-of-the-atmosphere (BOA) multiple scatter source term $\mathbf{B}(v)$ for geometry v. (Upwelling direction only)
MSCATBOA_SOURCETERM_F	3	Fourier component $\mathbf{B}_m(u,s)$ for user angle u and solar angle s
DIRECTBOA_SOURCETERM	3	Bottom-of-the-atmosphere (BOA) direct-beam source term $\mathbf{B}^*(v)$ for geometry v. (Upwelling direction only)
DIRECTBOA_SOURCETERM_F	3	Fourier component $\mathbf{B}^*_m(u,s)$ for user angle u and solar angle s

Table A8: Variables in LIDORT_L_RESULTS.VARS

Name	Gp	Description
ATMOSWF	1	Weighting functions $K(q,n,t,v,d)$ with respect to atmospheric variable q in layer n, at optical depth t, geometry v, direction d.
SURFACEWF	1	Weighting functions $KR(r,t,v,d)$ with respect to surface variable r, at optical depth t, geometry v, direction d.
ATMOSWF_F	1	Fourier component WFs $Km(q,n,t,u,s,d)$ w.r.t. variable q in layer n, at optical depth t, output stream angle u, solar beam s, direction d.
SURFACEWF_F	1	Fourier component WFs $KRm(r,t,u,s,d)$ w.r.t. surface variable r, at optical depth t, output stream angle u, solar beam s, direction d.
MINT_ATMOSWF	2	Weighting functions of Mean intensity $KI(q,n,t,s,d)$ w.r.t. variable q in layer n, at optical depth t, solar beam s, direction d.
MINT_SURFACEWF	2	Weighting functions of Mean intensity $KIR(r,t,s,d)$ with respect to surface variable r, at optical depth t, solar beam s, direction d.
FLUX_ATMOSWF	2	Weighting functions of Flux $KF(q,n,t,s,d)$ w.r.t. atmospheric variable q in layer n, at optical depth t, solar beam s, direction d.
FLUX_SURFACEWF	2	Weighting functions of Flux $KFR(r,t,s,d)$ wrt. surface variable r, at optical depth t, solar beam s, direction d.
ATMOSWF_MSCATSTERM	3	Layer-integrated multiple scatter source term WFs $L(q,n,k,v,d)$ for layer k, geometry v, direction d, w.r.t. variable q varying in layer n.
SURFACEWF_MSCATSTERM	3	Layer-integrated multiple scatter source term WFs $LR(r,k,v,d)$ w.r.t. surface variable r, for layer k, geometry v, direction d.
ATMOSWF_MSCATSTERM_F	3	Layer-integrated multiple scatter source term Fourier component WFs $Lm(q,n,k,u,s,d)$ for layer k, stream angle u ,solar beam s, direction d, w.r.t variable q varying in layer n.
SURFACEWF_MSCATSTERMF_F	3	Layer-integrated multiple scatter source term Fourier component WFs $LRm(r,k,u,s,d)$ with respect to surface variable r, for layer k, stream angle u ,solar beam s, direction d.
ATMOSWF_MSCATBOA_STERM	3	BOA multiple scatter source term weighting functions $BL(q,n,v)$ for geometry v, w.r.t. atmospheric variable q varying in layer n.
ATMOSWF_DIRECTBOA_STERM	3	BOA direct-beam source term weighting functions $BD(q,n,v)$ for geometry v, w.r.t. atmospheric variable q varying in layer n.
SURFACEWF_MSCATBOA_STERM	3	BOA multiple scatter source term weighting functions $BLR(r,s)$ with respect to surface variable r, solar beam s.
SURFACEWF_DIRECTBOA_STERM	3	BOA direct-beam source term weighting functions $BDR(r,s)$ with respect to surface variable r, solar beam s.
ATMOSWF_MSCATBOA_STERM_F	3	BOA multiple scatter source term Fourier WFs $BLm(q,n,u,s)$ for stream angle u, solar beam s, w.r.t. variable q varying in layer n.
ATMOSWF_DIRECTBOA_STERM_F	3	BOA direct-beam source term Fourier WFs $BDm(q,n,u,s)$ for stream angle u, solar beam s, w.r.t. variable q varying in layer n.
SURFACEWF_MSCATBOA_STERM_F	3	BOA multiple scatter source term Fourier WFs $BLRm(r,u,s)$ w.r.t. surface variable r, for stream angle u, solar beam s.
SURFACEWF_DIRECTBOA_STERM_F	3	BOA direct-beam source term Fourier WFs $BLRm(r,u,s)$ w.r.t. surface variable r, for stream angle u, solar beam s.