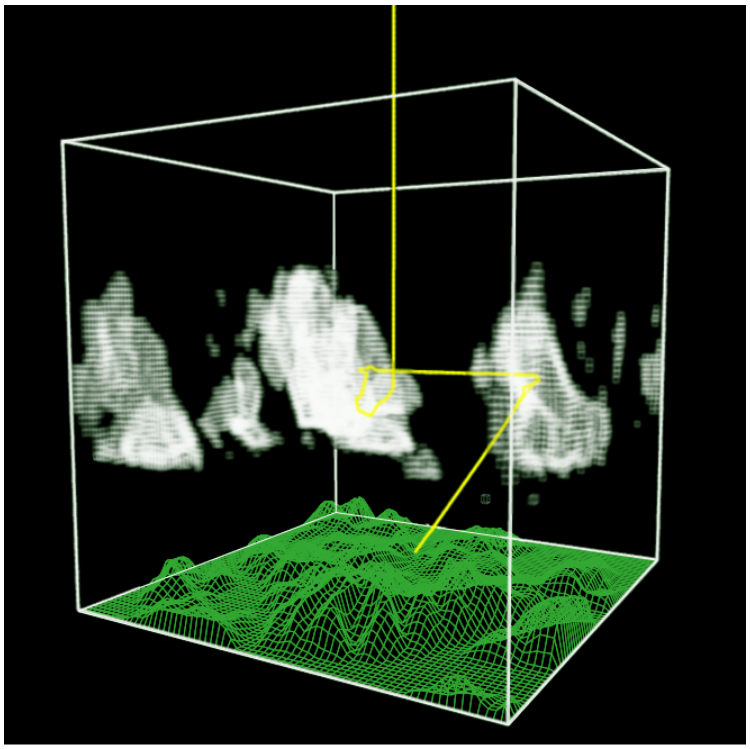
*Edition for libRadtran version 1.6-beta*

# libRadtran user's guide

Bernhard Mayer, Arve Kylling, Claudia Emde,
Ulrich Hamann, and Robert Buras



March 3, 2011

# Contents

# Chapter 1

# Preface

*libRadtran* is a library of radiative transfer routines and programs. The central program of the *libRadtran* package is the radiative transfer tool *uvspec*. The `uvspec` model was originally designed to calculate spectral irradiance and actinic flux in the ultraviolet and visible parts of the spectrum (Kylling, 1992) (initially the package was called `uvspec` and the executable still carries this name). Over the years, `uvspec` has undergone numerous extensions and improvements. The `uvspec` program now includes the full solar and thermal spectrum, currently from 120 nm to 100 $\mu$m. It has been designed as a user-friendly and versatile tool which provides a variety of options to setup and modify an atmosphere with molecules, aerosol particles, water and ice clouds, and a surface as lower boundary. One of the unique features of `uvspec` is that it includes not only one but a selection of about ten different radiative transfer equation solvers, fully transparent to the user, including the widely-used DISORT code by Stamnes et al. (1988), a fast two-stream code (Kylling et al., 1995), a polarization-dependent code polRadtran (Evans and Stephens, 1991) .

*libRadtran* also provides related utilities, like e.g. a Mie program (`mie`), some utilities for the calculation of the position of the sun (`zenith`, `noon`, `sza2time`), a few tools for interpolation, convolution, and integration (`spline`, `conv`, `integrate`), and several other small tools for setting up `uvspec` input and postprocessing `uvspec` output.

Further general information about *libRadtran* including examples of use may be found in the reference publication (Mayer and Kylling, 2005).

It is expected that the reader is familiar with radiative transfer terminology. In addition, a variety of techniques and parameterizations from various sources are used. For more information about the usefulness and applicability of these methods in a specific context, the user is referred to the referenced literature.

*Please note that this document is by no means complete. It is under rapid development and major changes will take place.*

## Acknowledgements

Many people have already contributed to *libRadtran*'s development. In addition to Bernhard Mayer (bernhard.mayer (at) dlr.de), Arve Kylling (arve.kylling (at) gmail.com), Ulrich Hamann (ulrich.hamann (at) dlr.de), Claudia Emde (claudia.emde (at) lmu.de), and Robert Buras (robert.buras (at) lmu.de), the following people have contributed to *libRadtran* or helped out in various other ways (the list is almost certainly incomplete – please let us know if we forgot somebody):

- The disort solver was developed by Knut Stamnes, Warren Wiscombe, S.C. Tsay, and K. Jayaweera
- disort was converted from Fortran to C by Tim Dowling (dowling (at) louisville.edu) which greatly enhances the performance of the solver in libRadtran
- Warren Wiscombe provided the Mie code MIEV0, and the routines to calculate the refractive indices of water and ice, REFWAT and ICEWAT.
- Seiji Kato (kato (at) aerosol.larc.nasa.gov) provided the correlated-k tables described in Kato et al. (1999).
- Tom Charlock (t.p.charlock (at) larc.nasa.gov), Quiang Fu (qfu (at) atm.dal.ca), and Fred Rose (f.g.rose (at) larc.nasa.gov) provided the most recent version of the Fu and Liou code.
- David Kratz (kratz (at) aquila.larc.nasa.gov) provided the routines for the simulation of the AVHRR channels described in Kratz (1995).
- Frank Evans (evans (at) nit.colorado.edu) provided the polradtran solver.
- Ola Engelsen provided data and support for different ozone absorption cross sections.
- Albano Gonzales (aglezf (at) ull.es) included the Yang et al. (2000), Key et al. (2002) ice crystal parameterization.
- Tables for the radiative properties of ice clouds for different particle "habits" were obtained from Jeff Key and Ping Yang, Yang et al. (2000), Key et al. (2002). In addition, Ping Yang and Heli Wei kindly provided a comprehensive database of particle single scattering properties which we used to derive a consistent set of ice cloud optical properties for the spectral range 0.2 - 100 micron following the detailed description in Key et al. (2002). A comprehensive dataset including the full phase matrices has been generated and provided by Hong Gang.
- Paul Ricchiazzi (paul (at) icess.ucsb.edu) and colleagues allowed us to include the complete gas absorption parameterization of their model SBDART into uvspec.

- Luca Bugliaro (luca.bugliaro (at) dlr.de) wrote the analytical TZS solver (thermal, zero scattering).
- Sina Lohmann (sina.lohmann (at) dlr.de) reduced the "overhead time" for reading the Kato et al. tables dramatically which resulted in a speedup of a factor of 2 in a twostr solar irradiance calculation.
- Detailed ice cloud properties were provided by Bryan Baum (bryan.baum (at) ssec.wisc.edu).
- Yongxiang Hu (yongxiang.hu-1 (at) nasa.gov) provided the delta-fit program used to calculate the Legendre coefficients for ic_properties baum_hufit.
- UCAR/Unidata for providing the *netCDF* library.
- Many unnamed users helped to improve the code by identifying or fixing bugs in the code.

# Chapter 2

# Radiative transfer theory

## 2.1 Overview

Radiative transfer in planetary atmospheres is a complex problem. The best tool for the solution may vary depending on the problem. The *libRadtran* package contains numerous tools that handle various aspects of atmospheric radiative transfer. The main tools will be presented later in chapter 3. To give the user a background for the problem to be solved, the theory behind will briefly be presented below. The radiative transfer equation is presented first, and solution methods and approximations are outlined afterwards.

The number of equations in this chapter may be intimidating even for the brave-hearted. If you just want to get things done and wonder if the *libRadtran* package includes tools that may be used for your problem, jump directly to chapter 3. Another good starting point is to try the examples available through the Graphical User Interface to the `uvspec` tool.

## 2.2 The radiative transfer equation

Quite generally, the distribution of photons in a dilute gas may be described by the Boltzmann equation[1]

$$\frac{\partial f}{\partial t} + \nabla_{\mathbf{r}}(\mathbf{v}\ f) + \nabla_{\mathbf{p}}(\mathbf{F}\ f) = Q(\mathbf{r}, \hat{n}, \nu, t). \tag{2.1}$$

Here, the photon distribution function $f(\mathbf{r}, \hat{n}, \nu, t)$ varies with location ($\mathbf{r}$), direction of propagation ($\hat{n}$), frequency ($\nu$) and time ($t$). It is defined such that

$$f(\mathbf{r}, \hat{n}, \nu, t)\ c\ \hat{n} \cdot d\mathbf{S}\ d\Omega\ d\nu\ dt \tag{2.2}$$

represents the number of photons with frequency between $\nu$ and $\nu + d\nu$ crossing a surface element $d\mathbf{S}$ in direction $\hat{n}$ into solid angle $d\Omega$ in time $dt$ (Stamnes 1986). The units of

[1] For a derivation of the Boltzmann equation see a textbook on statistical mechanics, for example Reif (1965). Also note that the Boltzmann equation is not a fundamental equation. For a derivation of the radiative transfer equation from the Maxwell equations see Mishchenko (2002).

$f(\mathbf{r}, \hat{n}, \nu, t)$ are $cm^{-3}\ sr^{-1}\ Hz^{-1}$ and $c$ is the speed of light. Furthermore, $\nabla_{\mathbf{r}}$ and $\nabla_{\mathbf{p}}$ are the divergence operators in configuration and momentum space, respectively. The photons may be subject to an external force $\mathbf{F}(\mathbf{r}, \hat{n}, \nu, t)$ and there may be sources and sinks of photons due to collisions and/or 'true' production and loss, which are represented by $Q(\mathbf{r}, \hat{n}, \nu, t)$.

In the absence of relativistic effects $\mathbf{F} = 0$, and the photons propagate in straight lines with velocity $\mathbf{v} = c\ \hat{n}$ between collisions. Using the relation

$$\nabla_{\mathbf{r}}(\mathbf{v}\ f) = f\ \nabla_{\mathbf{r}}\mathbf{v} + \mathbf{v} \cdot \nabla f = \mathbf{v} \cdot \nabla f, \tag{2.3}$$

where $\mathbf{r}$ and $\mathbf{v}$ are independent variables, Eq. 2.1 may be written as

$$\frac{\partial f}{\partial t} + c\ (\hat{n} \cdot \nabla)\ f = Q(\mathbf{r}, \hat{n}, \nu, t) \tag{2.4}$$

where the $\mathbf{r}$ subscript on the gradient operator $\nabla$ has been omitted.

The differential energy associated with the photon distribution is

$$dE = c\ h\nu\ f\ \hat{n} \cdot d\mathbf{S}\ d\Omega\ d\nu\ dt. \tag{2.5}$$

The specific intensity of photons $I(\mathbf{r}, \hat{n}, \nu, t)$ is defined such that ($\hat{n} \cdot d\mathbf{S} = \cos\theta\ dS$)

$$dE = I(\mathbf{r}, \hat{n}, \nu, t)\ dS\ \cos\theta\ d\Omega\ d\nu\ dt, \tag{2.6}$$

which gives

$$I(\mathbf{r}, \hat{n}, \nu, t) = c\ h\nu\ f(\mathbf{r}, \hat{n}, \nu, t). \tag{2.7}$$

In a steady state situation Eq. 2.4 may then be written as

$$(\hat{n} \cdot \nabla)\ I(\mathbf{r}, \hat{n}, \nu) = h\nu\ Q(\mathbf{r}, \hat{n}, \nu). \tag{2.8}$$

Eq. 2.8 may be interpreted as the radiative transfer equation in a general geometry. However, as long as the source term $Q(\mathbf{r}, \hat{n}, \nu)$ is not specified it is of little use. First, however, the two most common geometries for radiative transfer in planetary atmospheres will be described.

### 2.2.1 The streaming term

The streaming term $\hat{n} \cdot \nabla$ defines the geometry. In planetary atmospheres the cartesian and spherical geometries are most common. In cartesian geometry the plane-parallel approximation is often used while in spherical geometry the pseudo-spherical and spherical shell approximations are popular.

### Cartesian geometry - plane-parallel atmosphere

In a Cartesian coordinate system the streaming term may be written (Rottmann, 1991; Kuo et al., 1996)

$$\begin{aligned}\hat{n}\cdot\nabla &= n_x\frac{\partial}{\partial x}+n_y\frac{\partial}{\partial y}+n_z\frac{\partial}{\partial z}\\ &= \cos\phi\sqrt{1-\mu^2}\frac{\partial}{\partial x}+\sin\phi\sqrt{1-\mu^2}\frac{\partial}{\partial y}+\mu\frac{\partial}{\partial z},\end{aligned} \tag{2.9}$$

where $(n_x, n_y, n_z)$ are the components of the unit vector, $\mu = \cos\theta$ and $\phi$ is the azimuth angle.

In a plane-parallel geometry (Flat Earth approximation) the atmosphere is divided into parallel layers of infinite extensions in the $x$- and $y$-directions. This implies that there are no variation in the $x$- and $y$-directions. Hence, for this approximation the streaming term becomes

$$\hat{n}\cdot\nabla = \mu\frac{\partial}{\partial z}. \tag{2.10}$$

This approximation is used by numerous radiative transfer solvers, including the much used DISORT solver (Stamnes et al., 1988).

### Spherical geometry - pseudo-spherical atmosphere

In spherical geometry the streaming term becomes[2]

$$\begin{aligned}\hat{n}\cdot\nabla &= \mu\frac{\partial}{\partial r}+\frac{1-\mu^2}{r}\frac{\partial}{\partial\mu}\\ &+\frac{\sqrt{1-\mu^2}\sqrt{1-\mu_0^2}}{r}\left[\cos(\phi-\phi_0)\frac{\partial}{\partial\mu_0}+\frac{\mu_0}{1-\mu_0^2}\sin(\phi-\phi_0)\frac{\partial}{\partial(\phi-\phi_0)}\right].\end{aligned} \tag{2.11}$$

In a spherically symmetric (=spherical shell) atmosphere the streaming term reduces to

$$\hat{n}\cdot\nabla = \mu\frac{\partial}{\partial r}+\frac{1-\mu^2}{r}\frac{\partial}{\partial\mu}. \tag{2.12}$$

Dahlback and Stamnes (1991) has shown that for mean intensities it is sufficient to include only the first term in Eq. 2.12 for solar zenith angles up to 90°. Thus,

$$\hat{n}\cdot\nabla = \mu\frac{\partial}{\partial r}. \tag{2.13}$$

For this to hold the direct beam must be calculated in spherical geometry. This is the so-called pseudo-spherical approximation. It may work well for irradiances, mean intensities and nadir and zenith radiances. For irradiances in off-zenith and off-nadir directions it must be shown the angle derivatives are indeed negligible. This is rarely done in practice.

[2] A derivation is provided in Appendix O of Thomas and Stamnes (1999). The appendix is available from http://odin.mat.stevens-tech.edu/rttext/.

### 2.2.2 The source term

The source term on the right hand side of Eq. 2.8 includes all losses and gains of radiation in the direction and frequency of interest. For photons in a planetary atmosphere the source term may be written as[3]

$$\begin{aligned} h\nu\, Q(r,\hat{n},\nu) &= h\nu\, Q(r,\theta,\phi,\nu) = -\beta^{ext}(r,\nu)\, I(r,\theta,\phi,\nu) \\ &\quad + \frac{1}{4\pi}\int_0^{\infty} \beta^{sca}(r,\nu,\nu') \int_0^{2\pi} d\phi' \int_0^{\pi} d\theta' p(r,\theta,\phi;\theta',\phi') I(r,\theta',\phi',\nu') d\nu' \\ &\quad + \beta^{abs}(r,\nu)\, B[T(r)]. \end{aligned} \tag{2.14}$$

The first term represents loss of radiation due to absorption and scattering (=extinction) out of the photon beam. The second term (multiple scattering term) describes the number of photons scattered into the beam from all other directions and frequencies, finally, the third term gives the amount of thermal radiation emitted in the frequency range of interest.

The lower part of the Earth's atmosphere, may to a good approximation, be assumed to be in local thermodynamic equilibrium [4]. Thus, the emitted radiation is proportional to the Planck function, $B[T(r)]$, integrated over the frequency or wavelength region of interest. Furthermore, by Kirchhoff's law the emissivity coefficient $\beta^{emi}$ is equal to the absorption coefficient $\beta^{abs}$.

The absorption, scattering and extinction coefficients are defined as (Stamnes, 1986)

$$\beta^{abs}(r,\nu) = \sum_i \beta_i^{abs}(r,\nu), \qquad \beta_i^{abs}(r,\nu) = n_i(r)\sigma_i^{abs}(\nu) \tag{2.15}$$

$$\beta^{sca}(r,\nu) = \sum_i \beta_i^{sca}(r,\nu), \qquad \beta_i^{sca}(r,\nu) = n_i(r)\sigma_i^{sca}(\nu) \tag{2.16}$$

$$\beta^{ext}(r,\nu) = \beta^{abs}(r,\nu) + \beta^{sca}(r,\nu)$$

where $n_i(r)$ is the density of the atmospheric molecule species $i$ and $\sigma_i^{abs}(\nu)$ and $\sigma_i^{sca}(\nu)$ are the corresponding absorption and scattering cross sections. The phase function is defined as

$$p(r,\theta,\phi;\theta',\phi',\nu) = \frac{\sum_i \beta_i^{sca}(r,\nu) p_i(\theta,\phi;\theta',\phi',\nu)}{\sum_i \beta_i^{sca}(r,\nu)}$$

where the phase function for each species

$$p_i(\theta,\phi;\theta',\phi',\nu) = p_i(\cos\Theta,\nu) = \frac{\sigma_i^{sca}(\nu,\cos\Theta)}{\int_{4\pi} d\Omega\, \sigma_i^{sca}(\nu,\cos\Theta)}$$

[3] For a derivation of the individual terms see e.g. Chandrasekhar (1960).

[4] The hypothesis of local thermodynamic equilibrium (LTE) makes the assumption that all thermodynamic properties of the medium are the same as their thermodynamic equilibrium (T.E.) values at the local $T$ and density. Only the radiation field is allowed to depart from its T.E. value of $B[T(r)]$ and is obtained from a solution of the transfer equation. Such an approach is manifestly *internally inconsistent*. ... 'However, if the medium is subject only to *small* gradients over the mean free path a photon can travel before it is destroyed and thermalized by a collisional process, then the LTE approach is valid.' (adapted from Mihalas (1978, p. 26))

and the scattering angle $\Theta$ is related to the local polar and azimuth angles through

$$\cos\Theta = \cos\theta \ \cos\theta' + \sin\theta \ \sin\theta' \ \cos(\phi - \phi').$$

The temperature profile, the densities and absorption and scattering cross sections are all needed to solve the radiative transfer equation. Temperatures and densities may readily be obtained from measurements or atmospheric models. Cross sections are taken from measurements, from theoretical models or a combination of both.

### 2.2.3 The radiative transfer equation in 1D

In plane-parallel geometry the monochromatic[5] radiative transfer equation 2.8 is written by combining Eq. 2.10 and Eq. 2.14

$$\begin{aligned} -\mu\frac{dI(z,\mu,\phi)}{\beta^{ext}dz} &= I(z,\mu,\phi) \\ &\quad -\frac{\omega(z)}{4\pi}\int_0^{2\pi} d\phi' \int_{-1}^{1} d\mu' p(z,\mu,\phi;\mu',\phi') I(z,\mu',\phi') \\ &\quad -(1-\omega(z))B[T(z)] \end{aligned} \tag{2.17}$$

where the single scattering albedo

$$\omega(z) = \omega(z,\nu) = \frac{\beta_i^{sca}(z,\nu)}{\beta_i^{ext}(z,\nu)} = \frac{\beta_i^{sca}(z,\nu)}{\beta_i^{abs}(z,\nu) + \beta_i^{sca}(z,\nu)}.$$

Formally the pseudo-spherical radiative transfer equation is similar to Eq. 2.17, but with $z$ replaced by $r$.

### 2.2.4 Polarization - scalar versus vector

The intensity or radiance $I$, solved for in the above equations have a magnitude, a direction and a wavelength. In addition to this light also possesses a property called polarization. When assuming randomly oriented particles the radiative transfer equation formally does not change when including polarization. However, the scalar radiance $I$ is replaced with the vector quantity $\mathbf{I}$

$$\mathbf{I} = (I, Q, U, V), \tag{2.18}$$

where $I$, $Q$, $U$ and $V$ are the so-called Stokes parameters (see e.g. Bohren and Huffman (1998)). Furthermore, the phase function $p(r,\theta,\phi;\theta',\phi')$ is replaced by the $4 \times 4$ phase matrix $\mathbf{P}(r,\theta,\phi;\theta',\phi')$, and if thermal radiation is under consideration the Stokes emission vector must also be accounted for.

[5] Frequency redistribution is required if Raman scattering is included in the calculation. For many applications Raman scattering is negligible and the photons are assumed not to change frequency. They are monochromatic. Thus, all frequency dependence have been suppressed in Eq. 2.17.

The degree of polarization $p$ is defined as

$$p = \frac{\sqrt{Q^2 + U^2 + V^2}}{I}. \tag{2.19}$$

For completely polarized radiation, $Q^2 + U^2 + V^2 = I^2$, thus $p = 1$, and for unpolarized radiation, $Q = U = V = 0$, thus $p = 0$.

In addition to the degree of polarization, $p$, the degree of linear polarization is defined as

$$p_{lin} = \frac{\sqrt{Q^2 + U^2}}{I}, \tag{2.20}$$

and the the degree of circular polarization is defined as

$$p_{circ} = \frac{V}{I}. \tag{2.21}$$

Polarization is often ignored in radiative transfer calculations both due to the complexity involved in the solution of the RTE including polarization and the higher demand on computer resources by these solution methods. Also, for many applications polarization may be ignored. If you are concerned about your specific application, `uvspec` makes it easy to change solvers and thus readily allows comparisons to be made between scalar and vector calculations.

## 2.3 General solution considerations

A multitude of methods exist to solve the radiative transfer equation 2.8. Most methods have some commonalities and they are briefly described below.

### 2.3.1 Direct beam/diffuse radiation splitting

The integro-differential radiative transfer equation 2.8 gives the radiance field when solved with appropriate boundary conditions, that is, the radiation incident at the bottom and the top of the atmosphere. At the bottom of the atmosphere the Earth partly reflects radiation and also emits radiation as a quasi-black-body. At the top of the atmosphere ($z = z_{\text{toa}}$) a parallel beam of sunlight with magnitude $I^0$ in the direction $\mu_0$ may be present

$$I(z_{toa}, \mu) = I^0 \delta(\mu - \mu_0), \tag{2.22}$$

where $\delta(\mu - \mu_0)$ is the Dirac delta-function. It is akward to use a delta function for a boundary condition. However, a homogeneous differential equation with inhomogeneous boundary conditions may always be turned into an inhomogeneous differential equation with homogeneous boundary conditions. Since the integro-differential equation 2.8 is already inhomogeneous, the addition of another inhomogeneous term does not necessarily complicate the problem. Hence the intensity field is written as the sum of the direct (dir) and the scattered (sca)(or diffuse) radiation

$$I(z, \mu, \phi) = I^{\text{dir}}(z, \mu_0, \phi_0) + I^{\text{sca}}(z, \mu, \phi), \tag{2.23}$$

where $\mu_0$ and $\phi_0$ are the solar zenith and azimuth angles respectively. Inserting Eq. 2.23 into Eq. 2.8 it is seen that the direct beam satisfies

$$-\mu\frac{dI^{\text{dir}}(z,\mu_0,\phi_0)}{\beta^{ext}dz} = -\mu\frac{dI^{\text{dir}}(z,\mu_0,\phi_0)}{d\tau} = I^{\text{dir}}(z,\mu_0,\phi_0) \tag{2.24}$$

where the optical depth is defined as $d\tau = \beta^{ext}dz$. The scattered intensity satisfies in 1D (the sca superscript is omitted)

$$\begin{aligned} -\mu\frac{dI(\tau,\mu,\phi)}{d\tau} &= I(\tau,\mu,\phi) \\ &\quad -\frac{\omega(r)}{4\pi}\int_0^{2\pi}d\phi'\int_{-1}^{1}d\mu' p(\tau,\mu,\phi;\mu',\phi')I(\tau,\mu',\phi) \\ &\quad -(1-\omega(\tau))B[T(\tau)] \\ &\quad -\frac{\omega(\tau)I^0}{4\pi}p(\tau,\mu,\phi;\mu_0,\phi_0)e^{-\tau/\mu_0}. \end{aligned} \tag{2.25}$$

Solution of Eq. 2.24 for the direct beam yields the Beer-Lambert-Bouguer law

$$I^{dir}(\tau,\mu_0) = I^0\ e^{-\tau/\mu_0}. \tag{2.26}$$

The popular **disort** solver (Stamnes et al., 1988, 2000) solves Eqs. 2.24-2.25.

### 2.3.2 Pseudo-spherical approximation

In the pseudo-spherical approximation the extinction path $\tau/\mu_0$ in Eqs. 2.25 and 2.26 is replaced by the Chapman function, $ch(r,\mu_0)$ (Rees, 1989; Dahlback and Stamnes, 1991)

$$ch(r_0,\mu_0) = \int_{r_0}^{\infty}\frac{\beta^{ext}(r,\nu)\ dr}{\sqrt{1-\left(\frac{R+r_0}{R+r}\right)^2(1-\mu_0^2)}}. \tag{2.27}$$

Here $R$ is the radius of the earth and $r_0$ the distance above the earth's surface. The Chapman function describes the extinction path in a spherical atmosphere.

Thus, in the pseudo-spherical approximation the direct beam is correctly described by

$$I^{dir}(r,\mu) = I^0\ e^{-ch(r,\mu_0)} \tag{2.28}$$

and the diffuse radiation is approximated by replacing the plane-parallel direct beam source in Eq. 2.25 with the corresponding direct beam source in spherical geometry

$$\begin{aligned} -\mu\frac{dI(\tau,\mu,\phi)}{d\tau} &= I(\tau,\mu,\phi) \\ &\quad -\frac{\omega(r)}{4\pi}\int_0^{2\pi}d\phi'\int_{-1}^{1}d\mu' p(\tau,\mu,\phi;\mu',\phi')I(\tau,\mu',\phi) \\ &\quad -(1-\omega(\tau))B[T(\tau)] \\ &\quad -\frac{\omega(\tau)I^0}{4\pi}p(\tau,\mu,\phi;\mu_0,\phi_0)e^{-ch(\tau,\mu_0)}. \end{aligned} \tag{2.29}$$

The **sdisort** solver included in the libRadtran software package (Mayer and Kylling, 2005) solves Eqs. 2.28-2.29.

### 2.3.3 Boundary conditions

The diffuse radiative transfer Eq. 2.25 is solved subject to boundary conditions at the top and bottom of the atmosphere. At the top boundary there is no incident diffuse intensity[6] ($\mu \geq 0$)

$$I(\tau = 0, -\mu, \phi) = 0. \tag{2.30}$$

The bottom boundary condition may quite generally be formulated in terms of a bidirectional reflectivity, $\rho(\mu, \phi; -\mu', \phi')$, and directional emissivity, $\epsilon(\mu)$,

$$\begin{aligned} I(\tau = \tau_g, \mu, \phi) &= \epsilon(\mu) B[T(\tau_g)] + \frac{1}{\pi}\mu_0 I_0 e^{-\tau_g/\mu_0}\rho(\mu, \phi; -\mu', \phi') \\ &\quad + \frac{1}{\pi}\int_0^{2\pi} d\phi' \int_0^1 \rho(\mu, \phi; -\mu', \phi') I(\tau, -\mu', \phi')\mu' d\mu', \end{aligned} \tag{2.31}$$

where $T(\tau_g)$ is the temperature of the bottom boundary, here the Earth's surface.

In the case of a Lambertian reflecting bottom boundary with albedo $\rho(\mu, \phi; -\mu', \phi') = A$, Eq. 2.31 simplifies to

$$\begin{aligned} \pi I(\tau_L, \mu) &= \pi\, \epsilon\, B[T(\tau_g)] + \mu_0\, A\, I^0 e^{-\tau_g/\mu_0)} \\ &\quad + 2\pi\, A \int_0^{2\pi} d\phi' \int_0^1 \mu I(\tau_L, -\mu, \phi) d\mu. \end{aligned} \tag{2.32}$$

The albedo, $A$, gives the fraction of reflected light under the assumption that the surface reflects radiation isotropically (Lambert reflector). The emissivity $\epsilon = 1 - A$, by Kirchhoff's law. In both Eqs. 2.31 and 2.32 the first term on the right hand side is the thermal radiation emitted by the surface. The second term is due to reflection of the direct beam that has penetrated through the whole atmosphere and the last term is reflection of downward diffuse radiation

### 2.3.4 Separation of the azimuthal $\Phi$-dependence, Fourier decomposition

For scattering processes in the atmosphere the scattering phase function depends only on the angle $\Theta$ between the incident and scattered beams. This may be used to seperate out the $\Phi$-dependence in Eqs. 2.25 and 2.29 as follows. The phase function is first expanded as a series of Legendre polynomials

$$p(\tau, \mu, \phi; \mu', \phi') = p(\tau, \Phi) = \sum_{l=0}^{2M-1} (2l+1) g_l(\tau) p_l(\cos\Phi) \tag{2.33}$$

[6] The DISORT type RTE-solvers, **disort 1.3**, **disort 2.0**, **sdisort** and **twostr**, may include a diffuse radiation source at the top boundary. This may be of interest when for example modelling the aurora.

where the phase function moments $g_l$ are given by

$$g_l(\tau) = \frac{1}{2}\int_{-1}^{+1} p_l(\cos\Phi)p(\tau,\Phi)d(\cos\Phi). \tag{2.34}$$

The $g_1$ term is called the "asymmetry factor", and $g_0 = 1$ due to normalization of the phase function. Applying the addition theorem for spherical harmonics to Eq. 2.33 gives

$$p(\tau,\Phi) = \sum_{l=0}^{2M-1}(2l+1)g_l(\tau)\left\{p_l(\mu)p_l(\mu') + 2\sum_{m=1}^{l}\Lambda_l^m(\mu)\Lambda_l^m(\mu')\cos m(\phi-\phi')\right\} \tag{2.35}$$

where the normalized associated Legendre polynomials are defined as

$$\Lambda_l^m(\mu) = \sqrt{\frac{(l-m)!}{(l+m)!}}P_l^m(\mu), \tag{2.36}$$

and $P_l^m(\mu)$ are the standard Legendre polynomials. The cosine dependence of the phase function, Eq. 2.35, suggests that cosine expansion of the intensity may be fruitful. Expanding the intensity as a cosine Fourier series:

$$I(\tau,\mu,\phi) = \sum_{l=0}^{2M-1} I^m(\tau,\mu)\cos m(\phi_0-\phi) \tag{2.37}$$

and inserting into Eqs. 2.25 and 2.29 gives $2M$ independent integro-differential equation (only the plane-parallel version is shown here)

$$\begin{aligned} -\mu\frac{dI^m(\tau,\mu)}{d\tau} &= I^m(\tau,\mu) \\ &\quad -\frac{\omega(r)}{2}\int_{-1}^{1}d\mu'\sum_{l=m}^{2M-1}(2l+1)g_l(\tau)\Lambda_l^m(\mu)\Lambda_l^m(\mu')I^m(\tau,\mu') \\ &\quad -\delta_{m0}(1-\omega(\tau))B[T(\tau)] \\ &\quad -\frac{\omega(\tau)I^0}{4\pi}(2-\delta_{m0})\sum_{l=m}^{2M-1}(2l+1)g_l(\tau)\Lambda_l^m(\mu)\Lambda_l^m(\mu')e^{-\tau/\mu_0}. \end{aligned} \tag{2.38}$$

where

$$\delta_{m0} = \begin{cases} 1 & \text{if } m = 0 \\ 0 & \text{if } m \neq 0 \end{cases}$$

### 2.3.5 Calculated quantities

Solution of the radiative transfer equation generally yields the diffuse radiance

$$I(\tau, \mu, \phi) \tag{2.39}$$

and the direct radiance

$$I^{\mathrm{d}ir}(\tau, \mu_0, \phi_0). \tag{2.40}$$

For the solvers that include polarization the vector quantities of the above quantities are calculated. From these quantities the upward, $E^+(\tau)$, and downward, $E^-(\tau)$, fluxes, or irradiances, are calculated

$$E^+(\tau) = \int_0^{2\pi} d\phi \int_0^1 \mu I(\tau, \mu, \phi) d\mu \tag{2.41}$$

$$E^-(\tau) = \mu_0 I_0 e^{-\tau/\mu_0} + \int_0^{2\pi} d\phi \int_0^1 \mu I(\tau, -\mu, \phi) d\mu. \tag{2.42}$$

Furthermore, the mean intensity

$$\overline{I}(\tau) = \frac{1}{2\pi}\left[I_0 e^{-\tau/\mu_0} + \int_0^{2\pi} d\phi \int_0^1 I(\tau, -\mu, \phi) d\mu + \int_0^{2\pi} d\phi \int_0^1 I(\tau, \mu, \phi) d\mu\right], \tag{2.43}$$

is related to the actinic flux (Madronich, 1987), $F$, used for the calculation of photolysis (or photodissociation) rates

$$F(\tau) = 4\pi \overline{I}(\tau). \tag{2.44}$$

Finally, heating rates may be calculated from either the flux differences or the mean intensity.

$$\frac{\partial T}{\partial t} = -\frac{4\pi}{c_p \rho_m}\frac{\partial E}{\partial z} = -\frac{4\pi}{c_p \rho_m}(1 - w)(\overline{I} - B)\frac{\partial \tau}{\partial z}. \tag{2.45}$$

Note that the partial derivative of $\tau$ with respect to $z$ is needed since optical properties and $\overline{I}$ are calculated as functions of $\tau$.

The various radiative transfer equation solvers included in the `uvspec` tools in the *libRadtran* package, have different capabilities to calculate the above radiative quantities. The user is referred to section 3.2 for an overview of the different solvers included in the `uvspec` program and their respective capabilities. For a complete description of all solvers with options section 6.1 should be consulted. Finally, there is nothing to complement a thorough understanding of the problem at hand, the theory behind the chosen solution and a little reading of the code itself.

### 2.3.6 Lidar equation

The lidar equation can be written as (see e.g. Weitkamp (2005))

$$\frac{\mathrm{d}N(r)}{\mathrm{d}r} = \frac{E_0}{E_{\mathrm{phot}}} A_{\mathrm{det}} \eta \frac{O(r)}{4\pi r^2} p(\cos\pi) \beta^{sca}(r) \exp\left(-2\int_0^r dr' \beta^{ext}(r')\right), \tag{2.46}$$

where $N(r)$ is the number of detected photons, $E_0$ is the energy per laser pulse, $E_{\mathrm{phot}}$ is the energy per photon, $A_{\mathrm{det}}$ is the detector area, $\eta$ is the detector efficiency, $O(r)$ is the overlap function, $r$ is the range, and $p(\cos\pi)$ is the scattering phase function in backward direction. Note that the nomenclature here is consistent with the libRadtran documentation and differs from that in most lidar papers and books.

The lidar equation is a solution of the RTE for the special problem of a lidar signal, and is a single scattering approximation to the real world. Nevertheless it is applicable for many cases of interest. For space-borne lidars it should not be used.

Many lidarists are also interested in the lidar ratio, which is defined as

$$S(r) = \frac{4\pi}{p(\cos\pi)\omega(r)}. \tag{2.47}$$

For the special case of Lambertian surface reflection, the signal is

$$N_{\mathrm{surf}}(r_{\mathrm{surf}}) = \frac{E_0}{E_{\mathrm{phot}}} A_{\mathrm{det}} \eta \frac{O(r_{\mathrm{surf}})}{4\pi r_{\mathrm{surf}}^2} 4a \cos\theta_{\mathrm{refl}} \exp\left(-2\int_0^{r_{\mathrm{surf}}} dr' \beta^{ext}(r')\right), \tag{2.48}$$

where $r_{\mathrm{surf}}$ is the range of the surface, $a$ is the surface albedo, and $\theta_{\mathrm{refl}}$ is the inclination with which the laser beam hits the surface.

### 2.3.7 Verification of solution methods

To solve the radiative transfer equation involves complex numerical procedures that are difficult both to develop and to implement. Great care must be taken during implementation to assure that the numerical procedure is stable for any values and combinations of the input parameters, i.e. optical depth, single scattering albedo, phase function and boundary conditions. The testing of new solvers are typically done by the developers against analytical solutions which are available for a few special cases. Furthermore, tests and comparisons are made against other models and measurements. The reader is referred to the individual papers describing the various solvers for more information.

The input quantities needed by the solvers are optical depth, single scattering albedo, phase function and boundary conditions. These are calculated from atmospheric profiles of molecular density, trace gas species, water and ice clouds and aerosols. In addition, the absorption and scattering properties of the various species are taken from measurements or model calculations. The calculation of the optical properties are compared against other models and measurements during code development.

# Chapter 3

# Radiative transfer simulations - `uvspec`

The `uvspec` program calculates the radiation field in the Earth's atmosphere. Input to the model are the constituents of the atmosphere including various molecules, aerosols and clouds. The absorption and scattering properties of these constituents may either be taken from the algorithms and databases provided with *libRadtran* and `uvspec` or be provided by the user. Boundary conditions are the solar spectrum at the top of the atmosphere and the reflecting surface at the bottom. Several extraterrestrial solar spectra are provided with *libRadtran* and various surface models are also included.

The `uvspec` program is structured into the following three essential parts: (1) An atmospheric shell which converts atmospheric properties like ozone profile, surface pressure, or cloud microphysical parameters into optical properties required as input to (2) the radiative transfer equation solver which calculates radiances, irradiances, actinic fluxes and heating rates for the given optical properties; and (3) post-processing of the solver output including multiplication with the extraterrestrial solar irradiance correction of Earth-Sun distance, convolution with a slit-function, or integration over wavelength (depending on the choice of the user). For an overview see Figure 3.1.

The core of all radiative transfer models is a method to calculate the radiation field for a given distribution of optical properties by solving the radiative transfer equation. To solve the radiative transfer equation discussed in Chapter 2 the `uvspec` program has the unique feature of giving the user a choice of various radiative transfer solvers (table 3.2). This implies that for the radiative transfer problem at hand an appropriate solver may be chosen, e.g. a fast two-stream code to calculate approximate irradiance or a discrete ordinate code to accurately simulate radiances, with or without polarization.

Below the basic usage of `uvspec` is described first followed by a general description of the `uvspec` input file. The `uvspec` input file may either be generated manually using any text editor capable of saving files in ASCII (plain text) format. Or it may be generated by the `uvspec` Graphical User Interface found in the `GUI` folder. The input file description is followed by several examples of usage of `uvspec`. Finally the radiative transfer equation solvers available in `uvspec` are briefly described.

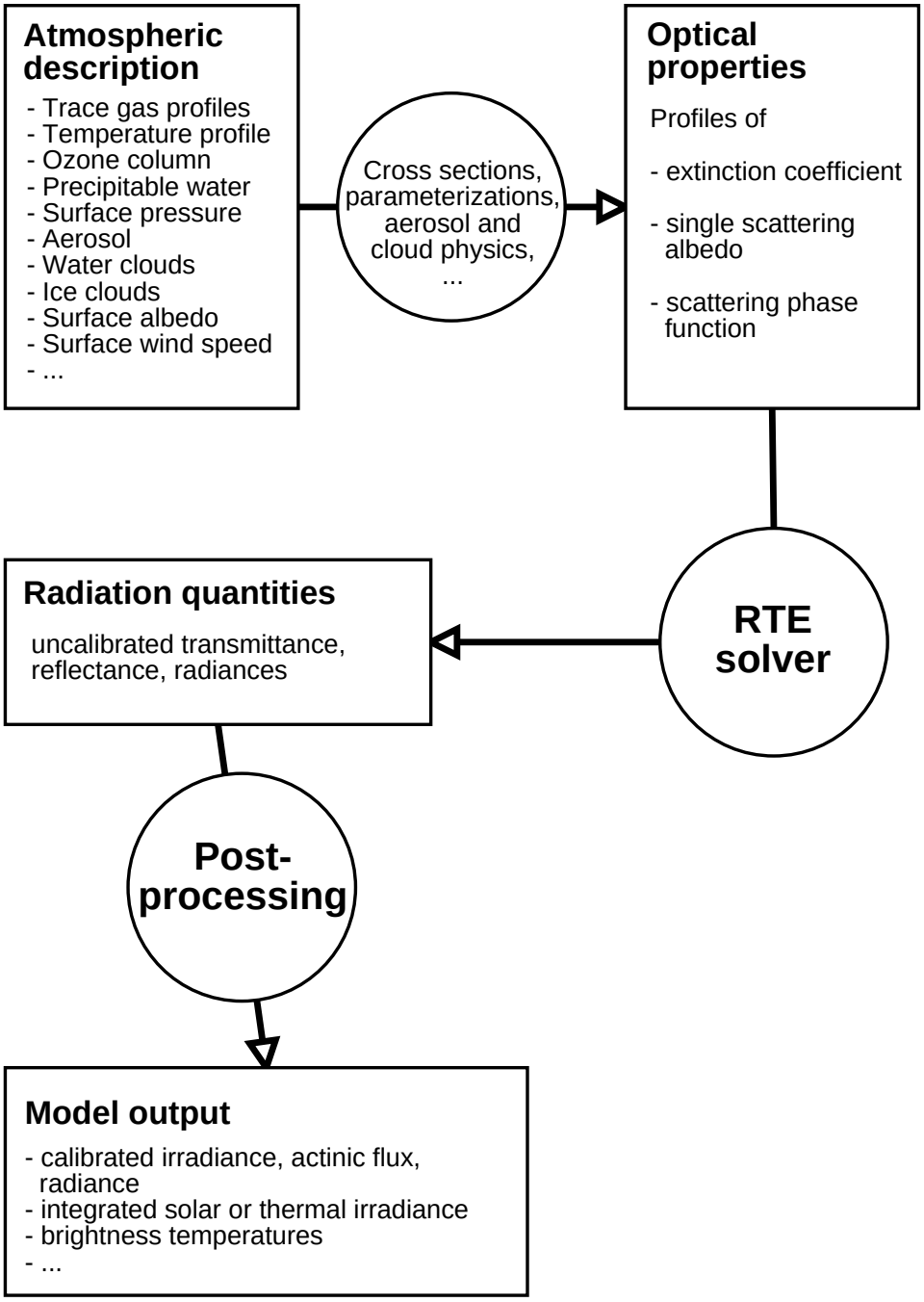



Figure 3.1: Structure of the `uvspec` model

## 3.1 Basic usage

### 3.1.1 Running `uvspec`

The `uvspec` program reads input from standard input, and outputs to standard output. It is normally invoked in the following way[1]:

```
uvspec < input_file > output_file
```

The formats of the input and output files are described below. Several realistic examples of input files are given in section 3.3.

The `uvspec` program may produce a wealth of diagnostic messages and warnings, depending on your use of `verbose` or `quiet`. Diagnostics, error messages, and warnings are written to `stderr` while the `uvspec` output is written to `stdout`. To make use of this extra information, you may want to write the standard `uvspec` output to one file and the diagnostic messages to another. To do so, try `(./uvspec < uvspec.inp > uvspec.out) >& verbose.txt`. The irradiances and radiances will be written to `uvspec.out` while all diagnostic messages go into `verbose.txt`. This method can also be used to collect `uvspec` error messages.

**Warning:** Please note the error checking on input variables is not complete at the moment. Hence, if you provide erroneous input, the outcome is unpredictable.

### 3.1.2 The `uvspec` input file

The `uvspec` program is controlled in a user-friendly way. The control options are named in a (hopefully) intuitive way.

The `uvspec` input file consists of single line entries, each making up a complete input to the `uvspec` program. First on the line comes the option name, followed by one or more parameter values. The option name and the parameter values are separated by white space. Filenames are entered without any surrounding single or double quotes. Comments are introduced by a `#`. Blank lines are ignored. The order of the lines is not important, with one exception: if the same input option is used more than once, the second one will usually over-write the first one. Be aware that also options in another included input file will overwrite options specified before.

### 3.1.3 How to setup an input file for your problem (checklist)

There are several steps to consider when setting up an input file for your specific problem. First of all we strongly recommend that you read a radiative transfer textbook to become familiar with what is required for your problem. Below is a short checklist including the steps you need to consider for each problem:

[1] The Graphical User Interface to `uvspec` provides another convenient way. The `uvspec` program may also be called as a function from another C program. See `src/worldloop.c` for an example.

1. **Wavelength grid / band parameterization**

   First you need to think about the spectral range and spectral resolution required for your calculation. As long as you stay in the ultraviolet or the lower visible spectral range you don't need to consider anything. Molecular absorption varies smoothly with wavelength in this range and a calculation with 0.5 or 1 nm step width should be sufficient. Above 500nm, however, absorption by water vapour, oxygen, and other trace gases starts; these absorption lines are very narrow, and a spectral calculation which resolves all lines is not feasible for most applications (such a line-by-line calculation is possible, however, if you provide your own spectral absorption cross sections). For most applications you need to select a correlated k-distribution, e.g. `correlated_k lowtran` which allows pseudo-spectral calculations (meaning that you still can calculate radiation at any wavelength you want, but the gas absorption is provided only at limited resolution - if you select the wavelengths too close, you will see the steps in your spectrum). For a spectral or pseudo-spectral calculation, you may define your own wavelength grid with `transmittance_wl_file` and we recommend to do that because otherwise you get the default 1nm step which might be too expensive for your application. Finally, in order to calculate integrated shortwave or integrated longwave radiation, please choose one of the pre-defined correlated-k distributions, e.g. `correlated_k kato2` or `correlated_k fu` because these are not only much more accurate but also much faster than a pseudo-spectral calculation. Please read the respective sections in the manual to become familiar with the `correlated_k` options.

2. **Quantities**

   The next point one needs to consider is the desired radiation quantity. Per default, `uvspec` provides direct, diffuse downward and diffuse upward solar irradiance and actinic flux at the surface. Thermal quantities can be calculated with `source thermal` - please note that `uvspec` currently does either solar or thermal, but not both at the same time. If both components are needed (e.g. for calculations around 3$\mu$m) then `uvspec` needs to be called twice. To calculate radiances in addition to the irradiances, simply define `umu`, `phi`, and `phi0` (see next section).

3. **Geometry**

   Geometry includes the location of the sun which is defined with `sza` (solar zenith angle) and `phi0` (azimuth). The azimuth is only required for radiance calculations. Please note that not only the solar zenith angle but also the sun-earth-distance change in the course of the year which may be considered with `day_of_year` (alternatively, `latitude`, `longitude`, and `time` may be used). The altitude of the location may be defined with altitude which modifies the profiles accordingly. Radiation at locations different from the surface may be calculated with `zout` which gives the sensor altitude above the ground. For satellites use `zout TOA` (top of atmosphere). For radiance calculations define the cosine of the viewing zenith angle `umu` and the sensor azimuth `phi` and don't forget to also specify the solar azimuth `phi0`. `umu`>0 means sensor looking downward (e.g. a satellite), `umu`<0 means looking upward. `phi` = `phi0` indicates that the sensor looks into the direction of the sun, `phi`-`phi0` = 180° means that the sun is in the back of the sensor.

4. **What do you need to setup the atmosphere?**

   To define an atmosphere, you need at least an `atmosphere_file` which usually contains profiles of pressure, temperature, air density, and concentrations of ozone, oxygen, water vapour, carbon dioxide, and nitrogen dioxide. The set of six standard atmospheres provided with libRadtran is usually a good start: `afglms` (mid-latitude summer), `afglmw` (mid-latitude winter), `afglss` (sub-arctic summer), `afglsw` (sub-arctic winter), `afglt` (tropical), and `afglus` (US standard). If you don't define anything else, you have an atmosphere with Rayleigh scattering and molecular absorption, but neither clouds, nor aerosol.

   (a) **Trace gases?**

   Trace gases are already there, as stated above. But sometimes you might want to modify the amount. There is a variety of options to do that, e.g. `ozone_column` which modifies the ozone column, or `co2_mixing_ratio`,

   (b) **Aerosols?**

   If you want aerosol, switch it on with `aerosol_default` and use either the default aerosol or one of the many `aerosol_` options to setup whatever you need.

   (c) **Clouds?**

   `uvspec` allows water and ice clouds. Define them with `wc_file` and `ic_file` and use one of the many `wc_` or `ic_` options to define what you need. Please note that for water and ice clouds you also have a choice of different parameterizations, e.g. `ic_properties fu`, `yang`, `baum`, - these are used to translate from liquid/ice water content and droplet/particle radius to optical properties. You need some experience with clouds to define something reasonable. Here are two typical choices for a `wc_file`

```
# z[km] LWC[g/m3] Reff[um]
    2         0         0
    1       0.1        10
```

   and an `ic_file`

```
# z[km] IWC[g/m3] Reff[um]
   10         0        0
    9     0.015       20
```

   The first is a water cloud with effective droplet radius of 10$\mu$m between 1 and 2 km, and an optical thickness of around 15; the second is an ice cloud with effective particle radius 20$\mu$m between 9 and 10 km and an optical thickness of about 1.

   (d) **Surface properties?**

   Per default, the surface albedo is zero - the surface absorbs all radiation. Define your own monochromatic albedo, a spectral `albedo_file` or a BRDF, e.g. for a water surface which is mainly determined by the wind speed `cox_and_munk_u10`.

5. **Choice of the radiative transfer equation (RTE) solver**

   The RTE-solver is the engine, or heart, in any radiative transfer code. All RTE-solvers involve some approximations to the radiative transfer equations, or the solution has some uncertainties due to the computational demands of the solution method. The choice of RTE-solver depends on your problem. For example, if your calculations involves a low sun you should not use a plane-parallel solver, but one which somehow accounts for the spherical shape of the Earth. You may choose between many RTE-solvers in `uvspec`. The default solution method to the radiative transfer is the discrete ordinate solver `disort2` which is the method of choice for most applications. There are other solvers like `rte_solver twostr` (faster but less accurate), `rte_solver polradtran` (polarization-dependent solver), or `rte_solver sdisort` (pseudo-spherical) . Even lidars can be simulated using `rte_solver sslidar`.

6. **Postprocessing**

   The spectral grid of the output is defined by the extraterrestrial spectrum. If you want spectrally integrated results, use either `correlated_k kato2` and `output sum` or `correlated_k lowtran` and `output integrate`. Check also other options like `filter_function_file`, `brightness`, etc. Instead of calibrated spectral quantities you might also want `transmittance` or `reflectivity`.

7. **Check your input**

   Last but not least, make always sure that `uvspec` actually does what you want it to do! A good way to do that is to use `verbose` which produces a lot of output. To reduce the amount, it is a good idea to do only a monochromatic calculation. Close to the end of the verbose output you will find profiles of the optical properties (optical thickness, asymmetry parameter, single scattering albedo) which give you a pretty good idea, e.g. if the clouds which you defined are already there, where the aerosol is, etc. As a general rule, never trust your input, but always check, play around, and improve. For if thou thinkest it cannot happen to me and why bother to use the verbose option, the gods shall surely punish thee for thy arrogance!

### 3.1.4 Output from `uvspec`

The `uvspec` output depends on the radiative transfer solver. The output formats are described in the following. The meaning of the symbols is described in Table 3.1. The output may be user controlled to some degree using the option `output_user`.

#### fdisort1, sdisort and spsdisort

For the fdisort1, sdisort and spsdisort solvers `uvspec` outputs one block of data to standard output (stdout) for each wavelength.

If `umu` is not specified the format of the block is

```
lambda edir edn eup uavgdir uavgdn uavgup
```

If umu is specified the format of the block is

```
lambda edir edn eup uavgdir uavgdn uavgup umu(0)
u0u(umu(0)) umu(1) u0u(umu(1)) .  .  .  .
```

If both umu and phi are specified the output format of each block is

```
lambda edir edn eup uavgdir uavgdn uavgup
                            phi(0)     ...     phi(m)
umu(0) u0u(umu(0)) uu(umu(0),phi(0)) ... uu(umu(0),phi(m))
umu(1) u0u(umu(1)) uu(umu(1),phi(0)) ... uu(umu(1),phi(m))
  .          .           .                        .
  .          .           .                        .
umu(n) u0u(umu(n)) uu(umu(n),phi(0)) ... uu(umu(n),phi(m))
```

and so on for each wavelength.

### twostr, twostrpp, and rodents

The format of the output line for the twostr solver is

```
lambda edir edn eup uavg
```

for each wavelength.

### polradtran

The output from the polradtran solver depends on the number of Stokes parameters, polradtran_nstokes.

If phi is not specified the output block is for each wavelength

```
lambda down_flux(1) up_flux(1) ...  down_flux(is) up_flux(is)
```

Here is is the number of Stokes parameters specified by polradtran_nstokes.

If phi and umu are specified the block is

```
lambda down_flux(1) up_flux(1) ...  down_flux(is) up_flux(is)
                              phi(0) ...  phi(m)
Stokes vector I
umu(0) u0u(umu(0)) uu(umu(0),phi(0)) ... uu(umu(0),phi(m))
umu(1) u0u(umu(1)) uu(umu(1),phi(0)) ... uu(umu(1),phi(m))
  .      .             .                    .
  .      .             .                    .
umu(n) u0u(umu(n)) uu(umu(n),phi(0)) ... uu(umu(n),phi(m))
Stokes vector Q
 .      .
 .      .
```

Note that `polradtran` outputs the total (=direct+diffuse) downward flux. Also note that `u0u` is always zero for `polradtran`.

#### sslidar

The format of the output line for the sslidar solver is

```
center-of-range number-of-photons lidar-ratio
```

for each range bin.

#### Description of symbols

The symbols used in section 3.1.4 are described in table 3.1.

The total downward irradiance is given by

```
irr_down = edir + edn
```

The total mean intensity is given by

```
uavg = uavgdir + uavgdn + uavgup
```

If `deltam` is on it does not make sense to look at the direct and diffuse contributions to `uavg` separately since they are delta-M scaled (that is, the direct would be larger than expected and the diffuse would be smaller).

## 3.2 RTE solvers included in `uvspec`

The `uvspec` tool includes numerous radiative transfer equation solvers. Below their capabilities and limitations are briefly described. A complete technical description of all solvers is far beyond the scope of the present document. The reader is referred to the individual papers describing the specific solver (see references for each solver). The solvers as they are named in the `uvspec` input files are written in **bold**. They also appear within the parenthesis in the subsection heads below. A list of all the solvers is provided in Table 3.2.

### 3.2.1 DIScrete ORdinate Radiative Transfer solvers (DISORT)

The discrete ordinate method was developed by Chandrasekhar (1960) and Stamnes et al. (1988). It solves the radiative transfer in 1-D geometry and allows accurate calculations of radiance, irradiance, and actinic flux. The standard DISORT solver developed by Stamnes et al. (1988, 2000) is probably the most versatile, well-tested and mostly used 1D radiative transfer solver on this planet.

| Symbol | Description |
|---|---|
| cmu | Computational polar angles from polradtran. |
| down_flux, up_flux | The total (direct+diffuse) downward (down_flux) and upward (up_flux) irradiances. Same units as extraterrestrial irradiance ( e.g mW/(m$^2$ nm) if using the atlas3 spectrum in the data/solar_flux directory.) |
| lambda | Wavelength (nm) |
| edir | Direct beam irradiance (same unit as extraterrestrial irradiance). |
| edn | Diffuse down irradiance, i.e. total minus direct beam (same unit as edir). |
| eup | Diffuse up irradiance (same unit as edir). |
| uavg | The mean intensity. Proportional to the actinic flux: To obtain the actinic flux, multiply the mean intensity by $4\pi$ (same unit as edir). |
| uavgdir | Direct beam contribution to the mean intensity (same unit as edir). |
| uavgdn | Diffuse downward radiation contribution to the mean intensity (same unit as edir). |
| uavgup | Diffuse upward radiation contribution to the mean intensity (same unit as edir). |
| u0u | The azimuthally averaged intensity at numu user specified angles umu (units of e.g. mW/(m$^2$ nm sr) if using the atlas3 spectrum in the data/solar_flux directory.) |
| uu | The radiance (intensity) at umu and phi user specified angles (unit e.g. mW/(m$^2$ nm sr) if using the atlas3 spectrum in the data/solar_flux directory.) |
| uu_down, uu_up | The downwelling and upwelling radiances (intensity) at cmu and phi angles (unit e.g. mW/(m$^2$ nm sr) if using the atlas3 spectrum in the data/solar_flux directory.) |

Table 3.1: Description of symbols used in the description of the model output.

Table 3.2: The radiative transfer equation solvers currently implemented in *libRadtran*.

| RTE solver | Geometry | Radiation quantities | Reference | Comments |
|---|---|---|---|---|
| DISORT 1.3 | 1D, PP | E, F, L | Stamnes et al. (1988) | discrete ordinate |
| DISORT 2.0 | 1D, PP | E, F, L | Stamnes et al. (2000) | discrete ordinate |
| cdisort | 1D, PP, PS | E, F, L | Buras et al. (submitted) | discrete ordinate |
| polradtran | 1D, PP | **E, F, L** | Evans and Stephens (1991) | polarization included |
| twostr | 1D, PS | E, F | Kylling et al. (1995) | two-stream; pseudo-spherical correction |
| rodents | 1D, PP | E,F | Zdunkowski et al. (2007) | Note that the reference contains errors (see next section) |
| sdisort | 1D, PS | E, F, L | Dahlback and Stamnes (1991) | pseudo-spherical correction, double precision, customized for airmass calculations |
| qdisort | 1D, PS | E, F, L | Kylling and Stamnes (1992) | based on sdisort, includes extra source term used for Raman scattering |
| spsdisort | 1D, PS | E, F, L | Dahlback and Stamnes (1991) | pseudo-spherical correction, single precision, not suitable for cloudy conditions |
| tzs | 1D, PP | L(TOA) | | thermal, zero scattering |
| sss | 1D, PP | L(TOA) | | solar, single scattering |
| sslidar | 1D, PP | $*$ | | |

$^{(a)}$ not included in the free package; available in joint projects

| | | |
|---|---|---|
| Explanation: | PP, plane-parallel | E, irradiance |
| | PS, pseudo-spherical | F, actinic flux |
| | SP, fully spherical | L, radiance |
| | 1D, one-dimensional | L(TOA), radiance at top of atmosphere |
| | 3D, three-dimensional | $*$ sslidar: see section 3.1.4. |

Bold face **E**, **F**, and **F** indicate vector quantities.

The uvspec model includes the standard DISORT solvers which are available from ftp://climate1.gsfc.nasa.gov/wiscombe/Multiple_Scatt/. In addition, a number of special purpose disort-family solvers are included.

From a historic point of view it is of interest to note that the first version of uvspec was based on the DISORT solver.

#### DISORT solvers (fdisort1, fdisort2, cdisort)

This group of solvers solve the 1D plane-parallel radiative transfer equation 2.25. A very complete and thorough description of the nitty-gritty details of the standard DISORT solver has been provided by Stamnes et al. (2000). The theory behind is clearly elucidated by Thomas and Stamnes (1999). Three versions of the DISORT solver are included in uvspec.

**fdisort** The original DISORT version 1.3.

**fdisort2** The DISORT version 2.0, with several improvements.

**cdisort** The C version of DISORT version 2.0, can also be used in pseudo-spherical mode.

The major changes between version 1.3 and 2.0 includes improved treatment for peaked phase functions and a realistic handling of the bidirectional reflectance function (BRDF). The modified version **disort2** included in uvspec further improves the treatment of peaked phase functions.

**cdisort** is the C version of the fdisort2. The C version runs in double precision, produces less instabilities, and is slightly faster. Further, it can be used in pseudo-spherical mode.

If you are in doubt, use the modified version 2.0. The default RTE solver in uvspec is **cdisort**. If you are worried about spherical effects please use the additional option cdisort_pseudospherical.

#### Pseudo-spherical DISORT (sdisort, spsdisort)

Dahlback and Stamnes (1991) extended the DISORT version 1.3 solver to pseudo-spherical geometry by solving equation 2.25. The **sdisort** solver includes further improvements, for instance the possibility to include 2D density profiles of trace gases. This option is of importance for air mass factor (AMF) calculations relevant for analysis of DOAS measurements. The **sdisort** solver dose not include the improvements of DISORT version 2.0.

Note that **sdisort** is not a fully spherical solver and may thus not be used for limb geometry.

The **spsdisort** solver is a single precision version of **sdisort**. Unless you have a 64-bit processor with compilers that do the numerics using all 64-bits we do not recommend that you use it because of numerical instabilities caused by the limited numerical resolution of 32-bits CPUs.

#### General source term (qdisort)

The **qdisort** solver is similar to **sdisort** with the addition of a general source term to the right in Eq. 2.29. It is only used when Raman scattering is included the calculation.

#### Two-stream solvers (twostr, twostrpp, rodents)

The DISORT solver are multi-stream solvers and thus not optimized for fast two-stream calculations. The **twostr** solver was developed by Kylling et al. (1995) and solves equation 2.25. Being a two-stream solution, **twostr** can not calculate radiances. Furthermore, based on the accuracy requirements of the specific application, the user is encouraged to make sample sensitivity test of **twostr** results versus for example **sdisort**.

The **twostrpp** solver is simply **twostr** run in plane-parallel geometry.

The **rodents** solver is the delta-Eddington twostream method presented in Zdunkowski et al. (2007), Sect. 6. Note that the equations (6.50) and (6.88) in the reference are wrong. Also note that the thermal radiation is not implemented as described on page 178 of the reference, but in analogy to the solar radiation. For more information, you are invited to visit our master course "Advanced Atmospheric Physics", next start in October 2011. The solver was implemented by Robert Buras, hence the name "ROberts' Delta-EddingtoN Two-Stream".

### 3.2.2 Polarization (polradtran)

The **polradtran** solver developed by Evans and Stephens (1991) solves the plane-parallel RTE including polarization in 1D. It should be noted that `polradtran` is not accurate for strongly peaked phase functions that are typical for water and ice cloud scattering in the shortwave spectral region.

### 3.2.3 Thermal zero scattering (tzs)

The **tzs** solver calculates the thermal radiance at the top of the atmosphere for a non-scattering atmosphere. In this case, the radiative transfer equation reduces to

$$-\mu \frac{dI(z,\mu,\phi)}{\beta^{ext} dz} = I(z,\mu,\phi) - (1-\omega(z))B[T(z)] \tag{3.1}$$

where $I(z,\mu,\phi) = I(z,\mu,\phi,\nu)$ represents the spectral radiance at the wavenumber $\nu$, $\omega(z) = \omega(z,\nu)$ is the single scattering albedo, $\beta^{ext} = \beta^{ext}(\nu)$ the exctinction coefficient and $B[T(z)] = B[T(z),\nu]$ is Planck's function for temperature $T$. This local problem can be solved by assuming a one-dimensional atmosphere that is split into a number of isothermal layers.

### 3.2.4 sslidar

The solver basically returns the solution of the lidar equation (2.46) and the lidar ratio, Eq. (2.47). The overlap function is set to 1. Input parameters for this solver are:

**sslidar_area** Detector area in units of $m^2$ (default: $1m^2$)

**sslidar_E0** Energy of laser pulse in units of J (default: 0.1J)

**sslidar_eff** Detector efficiency (default: 0.5)

**sslidar_nranges** Number of range bins (default: 100)

**sslidar_position** Altitude of position of lidar in units of km (default: 0km)

**sslidar_range** width of range bin in units of km (default: 0.1km)

Also, the cosine of the nadir angle into which the lidar is shooting/looking can be set using the option `umu` (default: 0).

The result is evaluated in the center of each range bin, i.e. the extinction from one range bin to the next is integrated correctly up to the middle of the range bin, where the backscatter coefficient is evaluated. This is then multiplied with the width of the range bin in order to get the number of photons detected in this range bin. The lidar ratio is also evaluated in the center of each range bin.

## 3.3 Examples

In the following sections, several examples are given, how to create an input file, how to define a cloudless sky atmosphere, how to add aerosols and clouds, etc. All examples are taken from the libRadtran examples directory and are part of the `uvspec` self-check. For a complete listing and explanation of all input options, have a look at section 6.1. More examples of `uvspec` input files (extension `.INP`) are found in the `examples` directory. Several examples are also availabe through the `uvspec` Graphical User Interface (see `GUI` directory).

### 3.3.1 Cloudless, aerosol-free atmosphere

The simplest possible input file contains only a few lines:

```
                          # Location of atmospheric profile file.
atmosphere_file ../data/atmmod/afglus.dat

                          # Location of the extraterrestrial spectrum
solar_file ../data/solar_flux/atlas_plus_modtran

wavelength 310.0 310.0    # Wavelength range [nm]

quiet
```

The first two statements define the location of some data files: the the atmospheric profile (atmosphere_file), and the extraterrestrial spectrum (solar_file). The third line defines the desired wavelength range which is a monochromatic data point in this example. All other data which are not explicitely mentioned assume a default value which is ”0” in most cases. Here, the solar zenith angle is 0, the surface albedo is 0, and the atmosphere does not contain clouds nor aerosols. Pressure, temperature, ozone concentration, etc. are read from atmosphere_file.

An example of a more complete input file for a clear sky atmosphere is:

```
                          # Location of atmospheric profile file.
atmosphere_file ../data/atmmod/afglus.dat
                          # Location of the extraterrestrial spectrum
solar_file ../data/solar_flux/atlas_plus_modtran
ozone_column  300.        # Scale ozone column to 300.0 DU
day_of_year 170           # Correct for Earth-Sun distance
albedo 0.2                # Surface albedo
sza 32.0                  # Solar zenith angle
rte_solver disort         # Radiative transfer equation solver; formerly fdisort1
deltam  on                # delta-M scaling on
nstr  6                   # Number of streams
wavelength 299.0 341.0    # Wavelength range [nm]
slit_function_file ../examples/TRI_SLIT.DAT
                          # Location of slit function
spline 300 340 1          # Interpolate from first to last in step

quiet
```

The atmosphere model, i.e. pressure, temperature, and ozone concentration profiles are read from ../data/atmmod/afglus.dat. The extraterrestrial solar flux is read from the file ../data/solar_flux/atlas_plus_modtran.

A wavelength dependent **surface albedo** may be specified using albedo_file instead of albedo. Non-Lambertian surface reflectance (BRDF) for vegetation and water may also be defined (please note that these require the use of rte_solver cdisort. The BRDF of vegetation is specified using rpv_rho0, rpv_k, and rpv_theta, following the definition of Rahman et al. (1993b). Wavelength-dependent BRDF for vegetation can be defined with rpv_file. The BRDF of water surfaces is parameterized following Cox and Munk (1954a,b) and Nakajima and Tanaka (1983). The respective parameters are the wind speed cox_and_munk_u10, the pigment concentration cox_and_munk_pcl, and the salinity cox_and_munk_sal. A complete description of these parameters is given in section 6.1.

It is helpful to know some details about the **input/output wavelength resolution** in uvspec and how it can be influenced by the user. Basically there are three independent wavelength grids, the **input grid**, the **internal grid**, and the **output grid**. The essential thing to know is that the internal grid is chosen by uvspec itself in a reasonable way, if not explicitely defined in the input file with transmittance_wl_file or molecular_tau_file. The output grid is completely independent of the internal grid and is entirely defined by the solar_file. The wavelength grid of all other input data

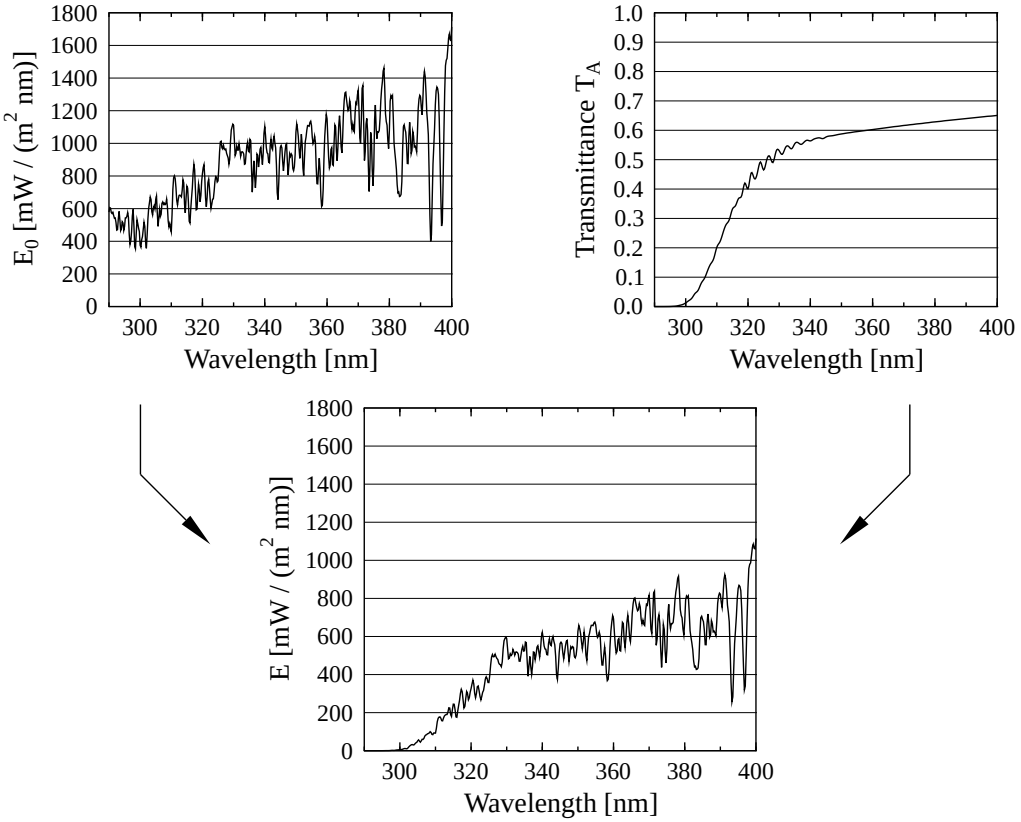



Figure 3.2: `uvspec` calculation of spectral irradiance in the ultraviolet range. (Top left) Low-resolution atmospheric transmittance for US standard atmosphere, solar zenith angle 0°. (Top right) High-resolution extraterrestrial irradiance Kurucz (1992), averaged over 0.1 nm intervals. (Bottom) Product of both: spectral irradiance.

(e.g. albedo, optical properties of aerosols and clouds, etc) is also completely independent. These data are automatically interpolated to the resolution of the internal wavelength grid. Hence, only two constraints are set to the gridding of the input data: (1), the wavelength range has to cover all internal grid points; and (2), it should be chosen in a reasonable manner to allow reasonable interpolation (which essentially means, dense enough).

In the ultraviolet/visible, `uvspec` uses an internal grid with a step with of 0.5nm below 350nm and 1nm above 350nm. This is a conservative choice which fully resolves the broad ozone absorption bands and the slowly varying Rayleigh, aerosol, and cloud extinctions. The idea is outlined in figure 3.2 which is taken from Mayer et al. (1997).

The transmittance (or reflectance) is calculated on a moderate resolution grid which reduces the number of calls to the `rte_solver` and hence the computational time. Then, the transmittance is interpolated to the wavelengths in the `solar_file` (which is usually defined with higher spectral resolution), multiplied with the extraterrestrial irradiance, and possibly post-processed. Hence, the wavelength in the output spectrum are those contained in the `solar_file` which has two important implications: (1) Only those wavelengths

are output that are contained in the solar_file. If e.g. a monochromatic calculation is defined by setting 'wavelength 327.14', there will only be output if the wavelength 327.14 is explicitely listed in solar_file; (2) this is also true at thermal wavelengths where the extraterrestrial irradiance is zero; hence, even for a calculation in the thermal range a solar_file can be specified which defines the output grid in the first column and arbitrary values in the second column. Keeping these points in mind, solar_file is a convenient way to define an arbitrary output grid. solar_file may be omitted for thermal radiation calculations (source thermal) as well as for transmittance and reflectivity calculations. If omitted, the output grid equals the internal wavelength grid.

If required, a **user-defined internal grid** can be specified with transmittance_wl_file or molecular_tau_file. Note that this is a way to speed up the calculation considerably. E.g., for some applications the internal grid in the UV-A and visible can be set to 10nm which would reduce computational times by up to a factor of 10.

Things are completely different if one of the correlated_k parameterizations is selected (see below). In this case all flexibility is taken away from the user which is an inherent feature of the k distribution method. Internal grid as well as the extraterrestrial file are in this case defined by the choice of the parameterization itself.

### 3.3.2 Spectral resolution

uvspec offers four different ways of spectral calculations:

1. **Spectrally resolved calculation** in the UV and visible spectral ranges;
2. **Line-by-line calculation** with user-defined molecular absorption data;
3. **The correlated-k method**.
4. **Pseudo-spectral calculation** with exponential-sum-fit, from LOWTRAN; code adopted from SBDART (Ricchiazzi et al., 1998);

The choice of the method is determined by the problem and the decision is therefore entirely up to the user. The spectrally resolved calculation and the line-by-line calculation are more or less exact methods while the correlated-k distribution and the pseudo-spectral calculation are approximations that provide a compromise between speed and accuracy. In the following it is briefly described which method fits which purpose:

A **spectrally resolved calculation** is the most straightforward way, and will be the choice for all users interested in the ultraviolet and visible spectral ranges. In the UV/vis gas absorption generally occurs in broad bands with only slow spectral variation, the most important of these being the Hartley, Huggins, and Chappuis bands of ozone. Hence, a radiative transfer calculation every 1nm usually is sufficient to fully resolve any spectral variation using the method described in the last section. Absorption cross sections for various species are included, among them the most important O3 and NO2.

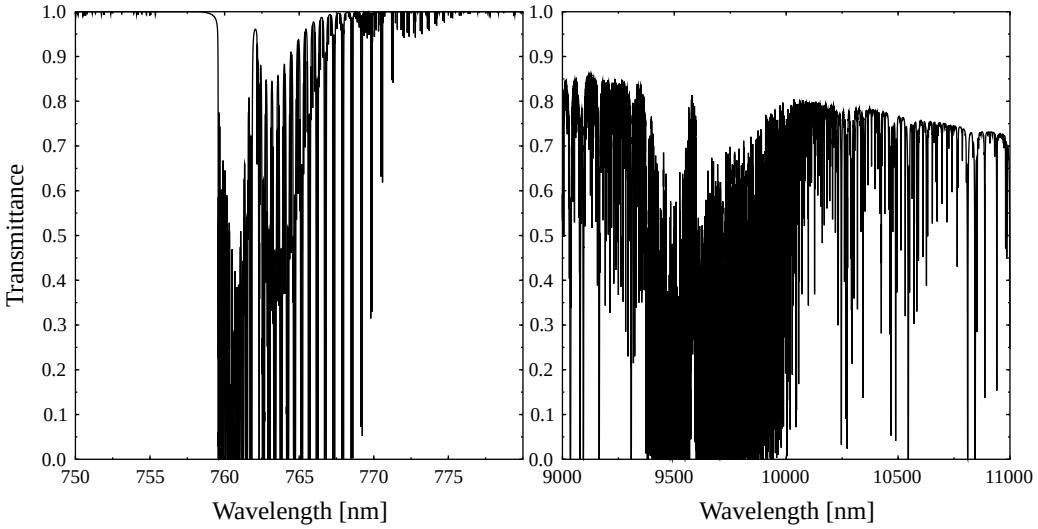



Figure 3.3: Line-by-line calculation of the atmospheric transmittance in two selected solar and thermal spectral ranges, the O2A-absorption band around 760 nm and a region within the infrared window around 10 $\mu$m.

In the infrared, however, molecular absorption spectra are characterized by thousands of narrow absorption lines. There are two ways to treat these, either by highly resolved spectral calculations, so-called **line-by-line** calculations, or by a band parameterization. Concerning line-by-line, `uvspec` offers the possibility to define a spectrally resolved absorption cross section profile using `molecular_tau_file`. There is no option in libRadtran to generate such a `molecular_tau_file`, because (1) the HITRAN database which forms the basis for such calculations amounts to about 100 MByte which are updated continuously; and (2), there are sophisticated line-by-line programs available, like e.g. `genln2` Edwards (1992). Using `genln2` it is straightforward to create the input for `uvspec` line-by-line calculations. line-by-line cross sections available for the six standard profiles that come with libRadtran are also available on request. Figure 3.3 shows an example of a line-by-line calculation of the atmospheric transmittance in two selected solar and thermal spectral ranges, the O2A-absorption band around 760 nm and a region within the infrared window around 10 $\mu$m.

All spectral lines in the left figure are due to absorption by oxygen, while the ones in the right figure are due to ozone, water vapour, and CO2. Line-by-line is obviously the exact way for radiation calculations. For most applications, however, line-by-line is far too slow. Here one needs a band parameterization, and the most accurate of these is the so-called **correlated-k approximation**. `uvspec` contains several correlated-k parameterizations which are invoked with `correlated_k`, in particular Kato et al. (1999); Fu and Liou (1992); Kratz and Varanasi (1995), as well as the possibility to specify a user-defined one. Kato et al. (1999) is a accurate parameterization for the solar spectral range. `uvspec` contains three different versions:

**Kato**

The original tables provide by Seiji Kato which should correspond to the full ver-

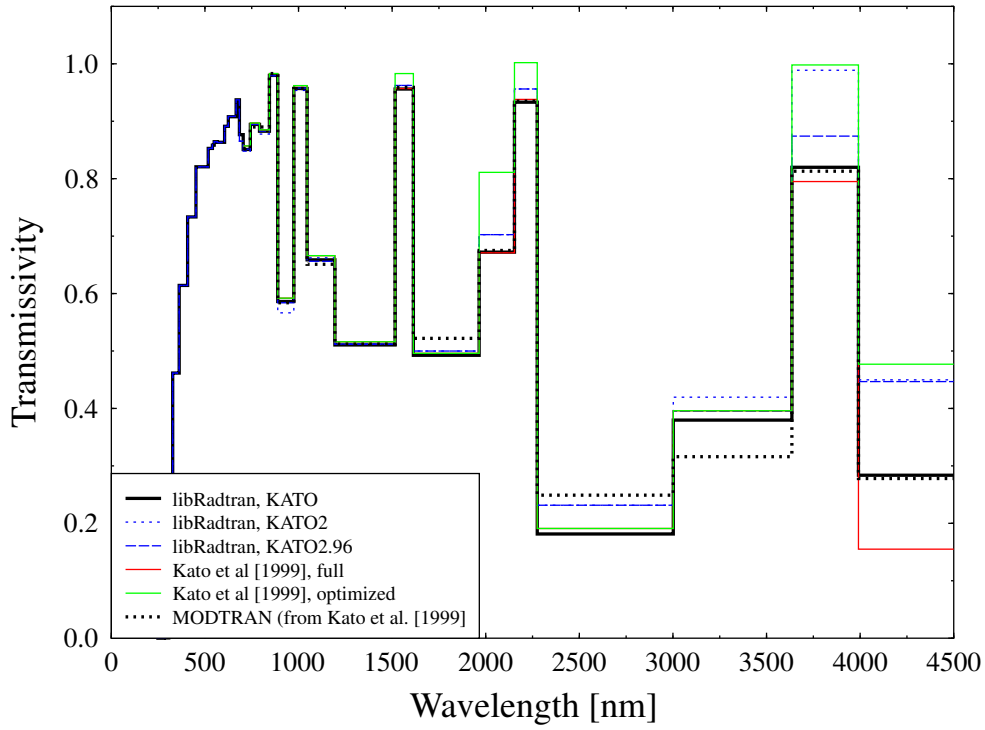



Figure 3.4: Comparison between the three parameterization which are part of `uvspec` and the data from Figure 3 by Kato et al. (1999).

sion described in Kato et al. (1999); 575 subbands total, that is, 575 calls to the `rte_solver`

**Kato2**
A new, optimized version of the tables, provided by Seiji Kato, 2003, with only 148 subbands (that is, calls to the `rte_solver`); the uncertainty is only slightly higher than `Kato`; the absorption coefficients are based on HITRAN 2000.

**Kato2.96**
Similar to `Kato2` but based on HITRAN96.

Figure 3.4 shows a comparison between the three parameterization which are part of libRadtran and the data from Figure 3 by Kato et al. (1999). It is immediately obvious that the uncertainty is high for all bands above 2.5 micrometer which is probably due to the treatment of band overlap. For this reasons, the results for the indiviudal bands should not be trusted while the integrated shortwave radiation (the sum of all 32 bands) is calculated with high accuracy because (1) the bands above 2.5 micrometer contribute only little to the integrated irradiance; and (2) errors are random and cancel each other to some degree.

For more information on these parameterizations please refer to the mentioned publications. Correlated-k is a powerful way to calculate spectrally integrated quantities, however, it takes

away some flexibility. In particular, this means that the wavelength grid is no longer chosen by the user but by the parameterization, that is, by `uvspec`. The `uvspec` output is then no longer spectral quantities, e.g. W/(m$^2$nm), but integrated over the spectral bands, e.g. W/m$^2$.

A simple but complete example for a correlated-k approximation of the solar spectrum:

```
# Conditions for the calculation of Figure 3 in
# Kato et al., JQSRT 62, 109-121, 1999.
# To compare the data, the direct irradiance calculated
# by uvspec has to be divided by cos(30 deg) because
# Kato et al. plot direct normal irradiance.

                          # Location of atmospheric profile file.
atmosphere_file ../examples/AFGLMS50.DAT
                          # Location of the extraterrestrial spectrum

albedo 0.2                # Surface albedo
sza 30.0                  # Solar zenith angle
rte_solver twostr         # Radiative transfer equation solver

correlated_k KATO         # Correlated-k by Kato et al. [1999]

output sum                # Calculate integrated solar irradiance

quiet
```

Here, the solar spectrum is split up into 32 bands according to Kato et al. (1999). In order to calculate integrated shortwave irradiance, simply sum the outputs, or even simpler, add `output sum` to the input file.

For **pseudo-spectral calculations** in the whole spectral range, we have adopted the molecular absorption parameterization from LOWTRAN/SBDART by Ricchiazzi et al. (1998). The respective section of this paper says:

> SBDART relies on low-resolution band models developed for the LOWTRAN 7 atmospheric trans-mission code (Pierluissi and Peng, 1985). These models provide clear-sky atmospheric transmission from 0 to 50000 cm-1 and include the effects of all radiatively active molecular species found in the earth s atmosphere. The models are derived from detailed line-by-line calculations that are degraded to 20 cm-1 resolution for use in LOWTRAN. This translates to a wavelength resolution of about 5 nm in the visible and about 200 nm in the thermal infrared. These band models represent rather large wavelength bands, and the transmission functions do not necessarily follow Beers Law. This means that the fractional transmission through a slab of material depends not only on the slab thickness, but also on the amount of material penetrated before entering the slab. Since the radiative transfer equation solved by SBDART assumes Beers Law behavior, it is necessary to express the transmission as the sum of several exponential functions (Wiscombe and Evans, 1977). SBDART uses a three-term exponential fit, which was also obtained from LOWTRAN 7. Each

term in the exponential fit implies a separate solution of the radiation transfer equation. Hence, the RT equation solver only needs to be invoked three times for each spectral increment. This is a great computational economy compared to a higher order fitting polynomial, but it may also be a source of significant error.

The LOWTRAN/SBDART gas parameterization is invoked with `correlated_k LOWTRAN`. The spectral resolution may be arbitrarily chosen by the user. If not explicitely defined with `transmittance_wl_file`, an internal grid with a step width of 0.5nm below 350nm and 1nm above 350nm is chosen which is practically overkill for most applications in the infrared. An extraterrestrial spectrum covering the complete solar range is provided at two different resolutions, `data/solar_flux/kurudz_1.0nm.dat` and `data/solar_flux/kurudz_0.1nm.dat`. An example for the solar range is shown in `examples/UVSPEC_LOWTRAN_SOLAR.INP`:

```
atmosphere_file ../data/atmmod/afglus.dat
solar_file ../data/solar_flux/kurudz_1.0nm.dat

albedo 0.2               # Surface albedo
sza 30.0                 # Solar zenith angle

rte_solver twostr        # Radiative transfer equation solver
wavelength 250.0 2500.0  # Wavelength range

correlated_k LOWTRAN      # select LOWTRAN molecular absorption

aerosol_default
aerosol_visibility 20

quiet
```

while `examples/UVSPEC_LOWTRAN_THERMAL.INP` shows how to do a thermal calculation:

```
# uvspec data files
data_files_path ../data/
atmosphere_file ../examples/AFGLUS.70KM
solar_file ../examples/UVSPEC_LOWTRAN_THERMAL.TRANS

source thermal

rte_solver twostr     # Radiative transfer equation solver
transmittance_wl_file ../examples/UVSPEC_LOWTRAN_THERMAL.TRANS

correlated_k LOWTRAN   # select LOWTRAN molecular absorption

output per_nm

quiet
```

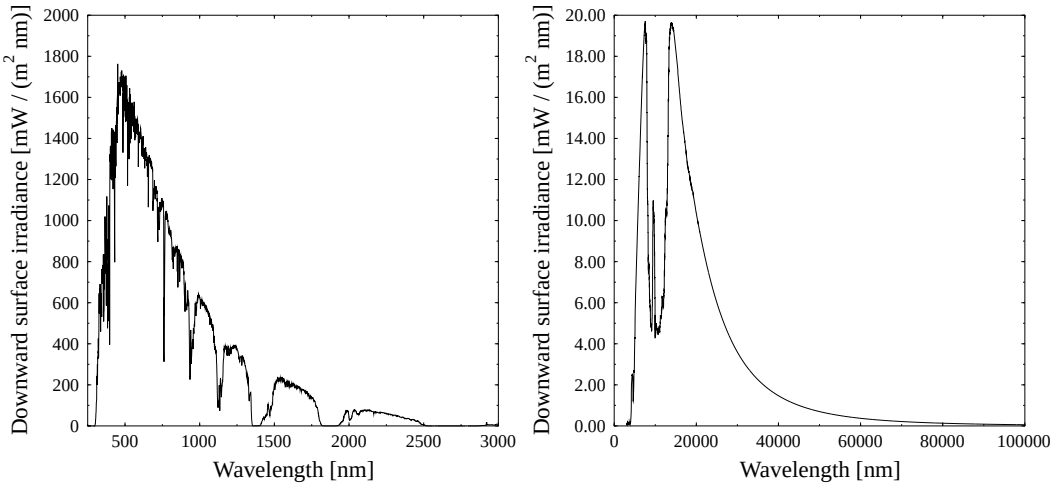



Figure 3.5: Results of the solar and thermal calculations using `correlated_k LOWTRAN`.

Figure 3.5 shows the results of the solar and thermal calculations. The water vapour absorption bands in the solar range are clearly visible, as is the absorption window around 10 micrometer in the thermal. Please note the following points:

- Thermal radiation is per default output in W/(m$^2$cm$^{-1}$), if the bandwidth is equal to 1 cm$^{-1}$ (default for `correlated_k LOWTRAN` calculations). Otherwise the output is the integrated flux over the wavenumber interval specified by `thermal_bandwith`, `thermal_bands_file`, or by the `correlated_k` option (Kato, Kato2, Kato2.96, Fu, AVHRR_KRATZ, or Generic). To convert e.g. to W/(m2 nm) use `output per_nm` or multiply with k/lambda where k is the wavenumber [cm$^{-1}$] and lambda is the wavelength [nm]. To calculate band-integrated thermal quantities please consider `thermal_bands_file`.
- Even though no extraterrestrial irradiance is required, a `solar_file` may be specified for the thermal case. The reason is that, as explained initially, the `solar_file` defines the output grid. The second column in `solar_file` can be chosen arbitrarily in this case because it is ignored.
- For the choice of the wavelength grid for the calculation (`transmittance_wl_grid`) please consider that the resolution of the absorption parameterization is 5 cm$^{-1}$ which translates to 0.3nm at 750 nm and to 50 nm at 10 $\mu$m. Choosing higher resolutions for the internal wavelength grid (`transmittance_wl_file`) is usually a waste of computational time.
- Please also make sure to choose a fine enough spectral resolution in order to capture all absorption features.

Figure 3.6 shows two selected wavelength intervals of the solar and thermal range, to demonstrate the spectral resolution of the LOWTRAN/SBDART absorption parameteriza-

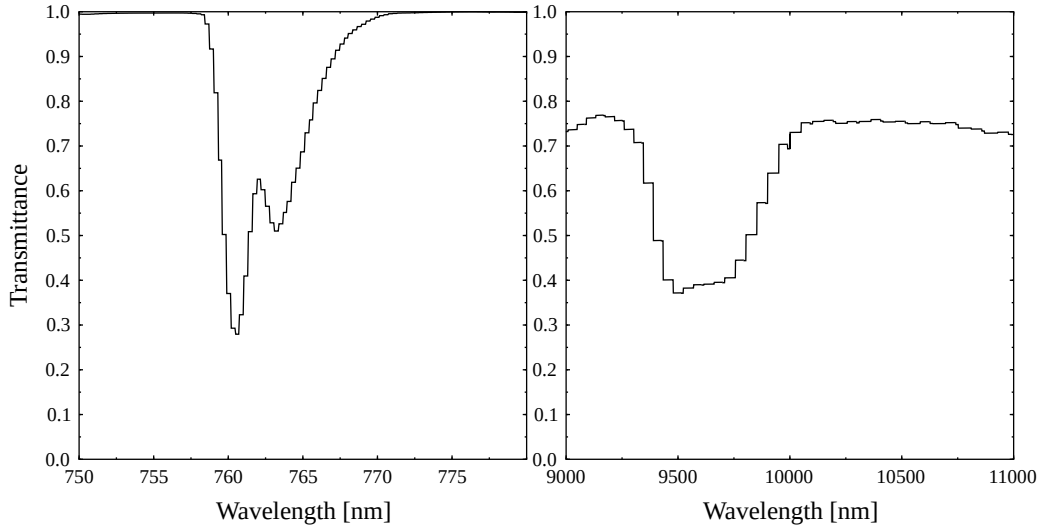



Figure 3.6: Two selected wavelength intervals of the solar and thermal range, to demonstrate the spectral resolution of the LOWTRAN absorption parameterization.

tion.

The resolution is about 5 $cm^{-1}$ which translates to about 0.3 nm in the left figure (oxygen A-band) and 50nm in the right figure (ozone absorption band in the atmospheric window). Compare this figure to the above line-by-line example to get an impression about the differences between both methods.

### 3.3.3 Aerosol

All options to set up and modify aerosol properties start with `aerosol_`. Aerosols may be specified in a hierarchical way. The most simple way to define an aerosol is by the command `aerosol_default` which will set up the aerosol model by Shettle (1989). The default properties are a rural type aerosol in the boundary layer, background aerosol above 2km, spring-summer conditions and a visibility of 50km. These settings may be modified with `aerosol_haze`, `aerosol_vulcan`, `aerosol_season`, and `aerosol_visibility`. More information can be introduced step by step, overwriting the default parameters. `aerosol_tau_file`, `aerosol_ssa_file`, and `aerosol_gg_file`, can be used to define the profiles of optical thickness, single scattering albedo, and asymmetry parameter. The integrated optical thickness can be set to a constant value using `aerosol_set_tau` or scaled with `aerosol_scale_tau`. The single scattering albedo may be scaled by `aerosol_scale_ssa` or set to a constant value by `aerosol_set_ssa`. The aerosol asymmetry factor may be set by `aerosol_set_gg`. The wavelength dependence of the aerosol optical depth is specified using the `aerosol_angstrom` parameter. `aerosol_moments_file` allows specification of the scattering phase function. If microphysical properties are available these may be introduced by defining the complex index of refraction `aerosol_refrac_index` or `aerosol_refrac_file` and the size distribution `aerosol_sizedist_file`. Fi-

nally, one may define the aerosol optical properties of each layer explicitely using `aerosol_files`.

The following list is an overview of some aerosol description parameters. The entries are arranged in a way that a parameter 'overwrites' all values higher up in the list.

**aerosol_default**
Generate default aerosol according to Shettle (1989).

**aerosol_vulcan, aerosol_haze, aerosol_season, aerosol_visibility**
Set Shettle (1989) aerosol properties (aerosol type, visibility)

**aerosol_files**
Specify optical properties of each layer explicitly, that is, extinction coefficient, single scattering albedo, and the moments of the phase function (everything as a function of wavelength).

**aerosol_tau_file , aerosol_ssa_file, aerosol_gg_file**
Overwrite profiles of optical thickness, single scattering albedo, and asymmetry parameter

**aerosol_moments_file**
Specify a phase function to be used instead of the Henyey-Greenstein phase function

**aerosol_refrac_index, aerosol_refrac file, aerosol_sizedist_file**
Calculate optical properties from size distribution and index of refraction using Mie theory. Here is an exception from the rule that ALL values defined above are overwritten because the optical thickness profile is re-scaled so that the optical thickness at the first internal wavelength is unchanged. It is done that way to give the user an easy means of specifying the optical thickness at a given wavelength.

**aerosol_species_file**
Define profiles of OPAC aerosol types.

**aerosol_set_gg, aerosol_set_ssa, aerosol_scale_ssa, aerosol_set_tau, aerosol_scale_tau**
Overwrite profiles of asymmetry parameter and single scattering albedo

**aerosol_angstrom**
Overwrite the integrated optical thickness (profiles are not changed).

An example for a `uvspec` aerosol description is

```
include ../examples/UVSPEC_CLEAR.INP

aerosol_vulcan 1          # Aerosol type above 2km
aerosol_haze 6            # Aerosol type below 2km
aerosol_season 1          # Summer season
aerosol_visibility 20.0   # Visibility
aerosol_angstrom 1.1 0.2  # Scale aerosol optical depth
                          # using Angstrom alpha and beta
                          # coefficients
aerosol_scale_ssa 0.85    # Scale the single scattering albedo
                          # for all wavelengths
aerosol_set_gg 0.70       # Set the asymmetry factor
aerosol_tau_file ../examples/AERO_TAU.DAT
                          # File with aerosol optical depth profile
```

By combining this with the clear sky example given above a complete `uvspec` input file including aerosol is constructed.

### 3.3.4 Water clouds

All options to set up and modify water cloud properties start with `wc_`.

The easiest way to define a water cloud is to specify a `wc_file` which defines the liquid water content and effective droplet radius at each model layer or level. By combining the following lines with the clear sky example given above a complete `uvspec` input file including water clouds is constructed.

```
include ../examples/UVSPEC_CLEAR.INP

wc_file ../examples/WCSIMPLE.DAT  # Location of water cloud file
wc_set_tau 15.                    # Set total water cloud optical depth
```

A typical example for a wc_file looks like:

```
# z LWC R_eff
# (km) (g/m3) (um)
5.000 0 0
4.000 0.2 12.0
3.000 0.1 10.0
2.000 0.1 8.0
```

The three columns are the level altitude [km], the liquid water content [g/m3], and the effective droplet radius [micrometer]. Per default (since version 1.4), these quantities are interpreted as layer quantities, and in the above example, the cloud would extend from 2 to 5 km, with e.g. a LWC of 0.2 g/m3 for the layer between 4 and 5 km. Before version 1.4 the `wc_file` was interpreted as level quantities (unless `wc_layer` was specified). That is, the value 0.2 g/m3 referred to altitude 4.0 km, as e.g. in a radiosonde profile. The properties

of each layer were calculated as average over the adjacent levels. E.g. the single scattering properties for the model layer between 3 and 4 km were obtained by averaging over the two levels 3 km and 4 km. To allow definition of sharp cloud boundaries, clouds were only formed if both liquid water contents above and below the respective layer were larger than 0. Hence, in the above example, the layers between 2 and 3 as well as between 3 and 4 km were cloudy while those between 1 and 2km and between 4 and 5 km were not. To switch to the old behaviour, use `wc_level`.

To make sure that the clouds really look as you want them to look, it is recommended to use the `verbose` option. This option shows not only where the cloud is actually placed, it rather tells the user exactly how LWC and effective radius are translated into optical properties, depending on the choice of parameterisation. Please also note that the definition of the empty top level at 5km is important to tell `uvspec` where the cloud ends. If omitted, the cloud would extend all the way to the top of the atmosphere.

There are different ways to convert the microphysical properties to optical properties. Either a parameterization is used, like the one by Hu and Stamnes (1993) (which is the default), or by Mie calculations. The latter are very time-consuming, hence we decided not to include these online into `uvspec` but rather have an option to read in pre-calculated Mie tables. The option `wc_properties` controls the method: `hu` selects the Hu and Stamnes (1993) parameterization, `mie` selects pre-calculated Mie tables which are available at `http://www.libradtran.org`. If `wc_properties mie` is selected, the model expects one or more Mie cloud property files including each internal wavelength which is useful for the fixed wavelength grids used by the correlated-k parameterisations `correlated_k kato`, `correlated_k fu`, etc. For a spectral calculation with free wavelength grid, there is also the possibility to use a pre-defined set of Mie tables (available at the web site) and to define `wc_properties_interpolate` to automatically interpolate the Mie properties to the internal wavelength grid. Although this is an extremely useful option, please use it careful because it might consume enourmous amounts of memory. Finally, there is the option to define an arbitrary file which can be generated using the `mie` tool (see section 4).

As for the aerosol, there are several options to modify the optical properties of the clouds. And of course there is also the option of defining all cloud properties explicitely using `wc_files`.

### 3.3.5 Ice clouds

Ice clouds are generated in a similar way to water clouds. All options to set up and modify ice cloud properties start with `ic_`. The main difference between water and ice clouds is that the latter usually consist of non-spherical particles. Hence, the conversion from microphysical to optical properties is much less defined, and several parameterizations are available. Please note in addition that there are different definitions of the effective radius. E.g. the parameterizations by Key et al. (2002) and Baum et al. (2005b, 2007) use the same definition whereas Fu (1996) actually uses another definition (see explanation of `ic_properties`). Finally, the sharp forward peak which is typical for ice particles can also be treated differently: E.g., Fu (1996) provides delta-scaled optical properties while Key et al. (2002) uses unscaled parameters (see explanation of `ic_fu_tau`). Figure 3.7 illustrates the implica-

tions. Plotted are extinction coefficient, asymmetry parameter, and single scattering albedo for ice clouds with an effective radius of 25 micrometers as a function of wavelength. If treated consistently, all parameterizations Key et al. (2002), Fu (1996), and Baum et al. (2005b, 2007) provide nearly identical results (solid lines, default settings in `uvspec`). If the definition of effective radius by Fu (1996) and delta-scaling is applied the optical properties look different. The effect of delta scaling on a radiative transfer calculation is that the direct irradiance in increased and the diffuse irradiance is decreased, whereas the global irradiance remains unchanged. The definition of the effective radius has a smaller effect but it modifies also the global irradiance. Note that the parameterization by Baum et al. (2005b, 2007) is plotted only up to 2200 nm. The reason is that it does not cover the full spectral region, it is available for two spectral regions (from 0.4–2.2 $\mu$m and from about 3–100 $\mu$m). For the calculation of radiances one should use either `ic_properties baum` or `ic_properties hey`, because these parameterizations include complete scattering phase functions and do not use approximations like the Heney-Greenstein phase function. `ic_properties hey` can also be used for polarized radiative transfer.

### 3.3.6 Calculation of radiances

To calculate radiances the following lines will do the job when combined with the clear sky example above

```
include ../examples/UVSPEC_AEROSOL.INP # Include's may be nested.

rte_solver disort  # This override what is specified in above file
                   # and files included in that file etc.

phi0 40.0                  # Solar azimuth angle
umu  -1.0 -0.5  -0.2 -0.1 # Output cosine of polar angle
phi 0.0 45. 90. 135. 180.0 225. 270.0 # Output azimuth angles
```

In this example radiances are calculated for the specified directions, where `umu` are the cosines of the viewing zenith directions and `phi` are the viewing azimuth angles.

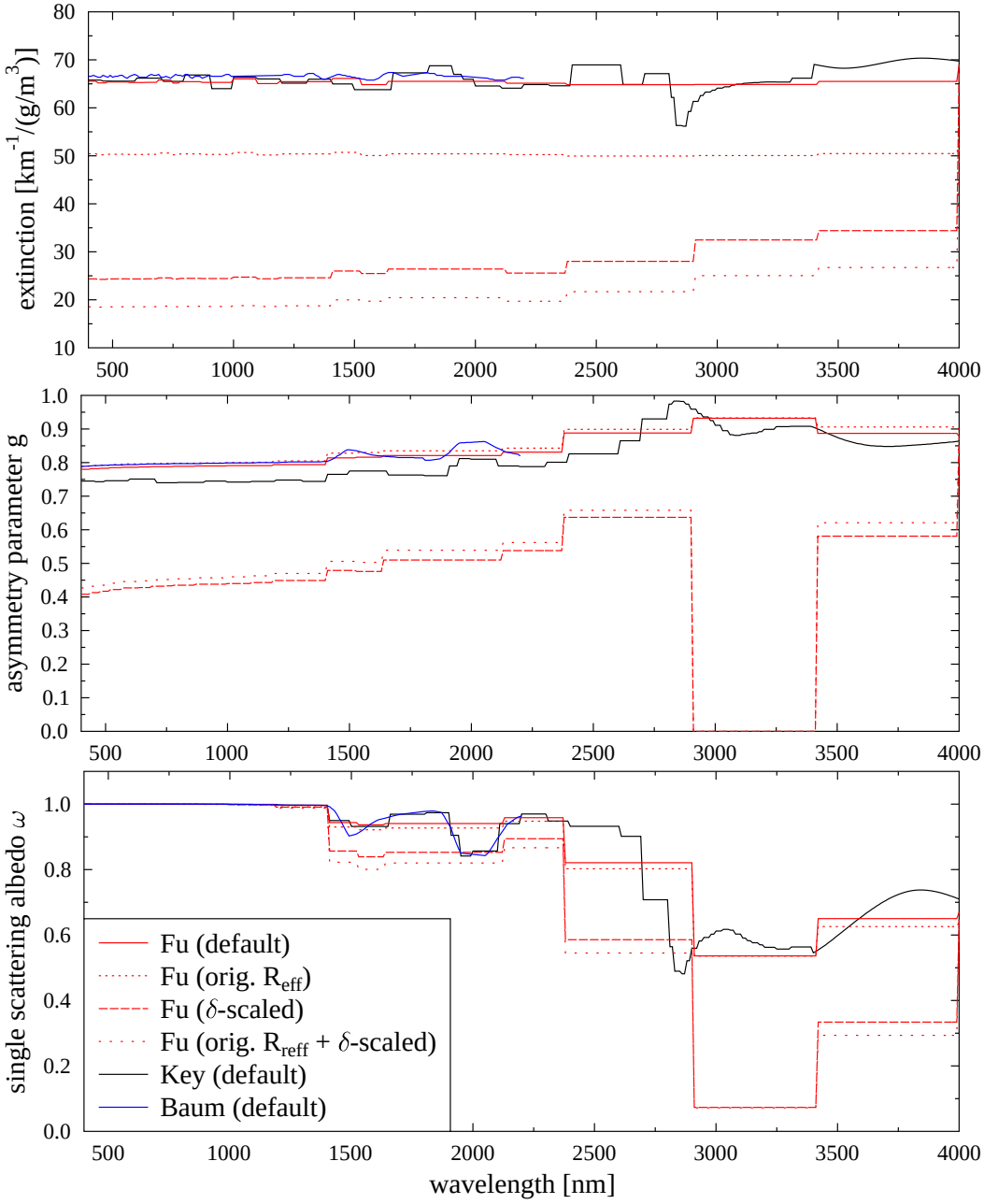



Figure 3.7: Extinction coefficient, asymmetry parameter, and single scattering albedo for ice clouds with an effective radius of 25 $\mu$m as a function of wavelength for various parameterizations.

# Chapter 4

# Calculation of optical properties - `mie`

*libRadtran* includes the tool `mie` to calculate optical properties of spherical particles. Two different efficient and well tested Mie codes are implemented: The one by Wiscombe (1980) and the one by Bohren and Huffman (1998). Scattering phase matrices and corresponding Legendre polynomials can currently only be calculated using the code by Wiscombe (1980).

## 4.1 Basic usage

### 4.1.1 Running `mie`

Mie scattering calculations are performed for a specified wavelength interval. The `mie` program reads input from standard input, and outputs to standard output or to a file. If `output_user netcdf` is specified `mie` generates a file that can be used for radiative transfer calculations with `uvspec`.

The `mie` tool is normally invoked in the following way:

```
mie < input_file > output_file
```

**Warning:** Please note the error checking on input variables is very scarce at the moment. Hence, if you provide erroneous input, the outcome is unpredictable.

### 4.1.2 The `mie` input file

The `mie` input file consists of single line entries, each making up a complete input to the mie program. First on the line comes the parameter name, followed by one or more parameter values. The parameter name and the parameter values are seperated by white space.

Filenames are entered without any surrounding single or double quotes.

Comments are introduced by a `#`. Blank lines are ignored.

### 4.1.3 Model output

The standard output (stdout) of the mie program is one line for each wavelength and each effective radius. The format of the output line is

```
  lambda refrac_real refrac_imag qext omega gg spike pmom
```

The keywords here are the same as in input option `output_user`.

If `output_user netcdf` is specified the output is written to a netcdf file in the format that is required by `uvspec`.

## 4.2 Examples

### 4.2.1 Calculation for one particle

The following example shows a Mie calculation for a single spherical particel with a radius of 200 $\mu$m. Thew refractive index is specified by the user. The calculation is performed for wavelengths from 200 to 5000 nm in 5 nm steps.

```
mie_program MIEV0              # Select Mie code by Wiscombe
refrac user 1.75  0.16         # Specify refractive index
r_eff 200.                     # Specify particle radius
wavelength  280. 5000.         # Define wavelengths
wavelength_step 5.
```

### 4.2.2 Calculation for a size distribution

Not all cloud droplets are of one specific size. The cloud droplet size spectrum may be represented for instance by a gamma distribution. Gamma distributions can easily be specified using the option `distribution gamma` as demonstrated in the following example:

```
mie_program MIEV0           # Select Mie code by Wiscombe
refrac water                # Use refractive index of water
r_eff 4 12 1                # Specify effective radius grid
distribution gamma 7        # Specify gamma size distribution (alpha=7)
wavelength  1600 1600       # Define wavelength
nstokes 4                   # Calculate all phase matrix elements
nmom 500                    # Number of Legendre terms to be computed
nthetamax 1000              # Maximum number of scattering angles to be
                            #    used to store the phase matrix
output_user netcdf          # Write output to netcdf file
verbose                     # Print verbose output
```

The refractive index of water is taken for this calculation. In order to generate input for `uvspec` Legendre polynomials and from those the phase matrices need to be calculated. The option `nmom` specifies how many Legendre polynomials shall be computed. If the

selected number is too small for an accurate representation of the phase matrix, a warning is given. If `output_user netcdf` is specified the corresponding phase matrices are calculated from the Legendre moments. The scattering angle grid is optimized so that the phase matrix is sampled as accurate as possible. The option `nthetamax` can be used to set an upper limit of scattering angle grid points to be used. This example generates a netcdf file which can directly be used in `uvspec` with the options `wc_properties` or `ic_properties`.

# Chapter 5

# Further tools

Besides `uvspec` and `mie` *libRadtran* provides several small tools related to radiative transfer in the atmosphere. These tools can be found in the `bin` directory. Some of the tools are described in this chapter.

Help for all tools can be obtained on the command line using the option `-h`.

## 5.1 General tools

### 5.1.1 Integration - `integrate`

`integrate` calculates the integral between limits $x_{\min}$ and $x_{\max}$ by interpolating the data points (x[i], y[i]) with natural cubic splines or linear interpolation. $x_{\min}$ and $x_{\max}$ are the minimum and maximum values of the first column in the input file. The x-values in the first column must be in ascending order.

The different options to `integrate` are displayed when executing:

```
integrate -h
```

### 5.1.2 Interpolation - `spline`

`spline` interpolates discrete data points using natural cubic splines or linear interpolation. The x-values in the first column must be in ascending order.

The different options to `spline` are displayed when executing:

```
spline -h
```

### 5.1.3 Convolution - `conv`

`conv` convolutes a spectrum with a given filter function.

The different options to `conv` are displayed when executing:

```
conv -h
```

### 5.1.4 Add level to profile - `addlevel`

`addlevel` is a simple shell script to add a level to one of the existing standard profiles.

The different options to `addlevel` are displayed when executing:

```
addlevel -h
```

### 5.1.5 Numerical difference between two files -`ndiff`

The Perl script `ndiff` calculates the relative difference between two files containing columns of numbers (file1/file0). The first column is not included. The calculated differences are output to stdout. If limit is different from 0.0, the number of differences greater than abs(maxdiff) are printed to stdout. The `ndiff` script is extensively used by the `test/test.pl` script invoked by `make check`.

The `ndiff` script is invoked by

```
ndiff [options] file0 file1
```

The script understands the following options

**–limit** <**value**> The minimum value in file0 considered when counting the number of differences between file0 and file1. Default is 0.0.

**–maxdiff** <**value**> The maximum relative difference allowed between file0 and file1. Defaut is 0.0.

**–sub** Subtract file1 - file0 instead of division

**–nox** First column is included

**–quiet** The differences are not output, but the number of differences are still printed.

**–help** Print help message.

## 5.2 Tools to generate input data to and analyse output data from `uvspec`

### 5.2.1 Calculate albedo of snow - `Gen_snow_tab`, `snowalbedo`

The `Gen_snow_tab.pl` script and the `snowalbedo` program may be used to calculate the diffuse and direct albedo of snow as formulated by Warren and Wiscombe (1980).

First a table of various snow optical properties must be generated. This is done by the PerlGen_snow_tab.pl script. The resulting tables will be read by the snowalbedo program which will calculate the wanted surface albedo quantities.

Generating the tables by the Gen_snow_tab.pl script is straightforward as the script only takes one argument, namely the name of the file body (It will also print a small help message if --help is given to it). The script will generate three files with extensions .gg, .qext and .ssa.

```
perl Gen_snow_tab.pl --file <name>
```

The generated tables is read by the snowalbedo program which requires the following options:

**-l** Equivalent depth of liquid water in snowpack (g cm -2)

**-r** mean grain radius ($\mu$m)

**-u** cosine of solar zenith angle

The options below are optional

**-a** albedo of underlying surface, default 0.03

**-p** turn of printing of messages

**-h** Print help message.

A typical usage of snowalbedo is (Gen_snow_tab.pl --file ../examples/MIE_ICE_TAB has been executed first)

```
snowalbedo ../examples/MIE_ICE_TAB -l 0.05 -r 50 -u 0.5 -p
```

This will produce the following output (only two first output lines shown)

```
290.0 2.00893  0.9999776000 0.88037   0.9728   0.9689
291.0 2.01212  0.9999782400 0.88064   0.9731   0.9693
```

Here, the various columns have the following content

1. wavelength (nm)
2. Q_ext
3. Single scattering albedo
4. Asymmetry parameter
5. Direct albedo
6. Diffuse albedo

### 5.2.2 Calculate cloud properties - `cldprp`

`cldprp` calculates wavelength-dependent cloud properties using one of several parameterizations.

The different options to `cldprp` are displayed when executing:

```
cldprp -h
```

### 5.2.3 Solar zenith and azimuth angle - `zenith`

The `zenith` tool calculates the solar zenith and azimuth angle for a given time and location. Output is to stdout and is self-explanatory (unless the `-q` option is used).

The solar zenith and azimuth angles are calculated using the algorithm of Blanco-Muriel et al. (2001). If the `-S` option is invoked the Spencer (1971) algorithm is used.

The `zenith` tool is invoked by

```
zenith [options] <day> <month> <hour> <min> [sec]
```

where the various options are

**-a** <latitude> Latitude (North positive)

**-o** <longitude> Longitude (West positive)

**-s** <std. long> Standard Longitude (West positive) this is the longitude to which the time zone refers (-15 deg for central Europe, corresponds to UTC+1).

**-l** <location> Instead of `-a`, `-o` and `-s` define a location. possible locations are ifu, dlrop.

**-y** <yyyy> year; not used if -S specified, default: 2003.

**-S** Use the Spencer algorithm.

**-e** Calculate eccentricity.

**-t** <UTC + x> Time zone; e.g. -t2 means UTC + 2.

**-q** Be quiet.

**-h** Print help message.

The options below apply if the solar zenith angle is wanted as a function of wavelength. This is useful for simulation of scanning spectroradiometer measurements. Output is two columns with wavelength and solar zenith angle. All options must be specfied. However <hour> and <min> should not be specified. To avoid too much output use the `-q` option.

**-B** start_time (decimal hours of Greenwich time)

**-E** end_time (decimal hours of Greenwich time)

**-u** start_wavelength (nanometers)

**-v** end_wavelength (nanometers)

**-w** step_wavelength (nanometers)

The following invocation of `zenith` calculates the solar zenith and azimuth angles at the time and location of the writing of this text

```
zenith -a 62.462052 -o -6.303358 -s -15 -y 2010 4 3 9 35
```

### 5.2.4 Local noon time - `noon`

The `noon` tool calculates the local noon time given a location in terms of longitude and latitude or a location name using the `-l` option. Output is to stdout and is self-explanatory.

The local noon time is calculated using the algorithm of Blanco-Muriel et al. (2001). If the `-S` option is invoked the Spencer (1971) algorithm is used.

The `noon` tool is invoked by

```
noon [options] <day> <month>
```

where the various options are

**-a** <latitude> Latitude (North positive)

**-o** <longitude> Longitude (West positive)

**-s** <std. long> Standard Longitude (West positive) this is the longitude to which the time zone refers (-15 deg for central Europe, corresponds to UTC+1).

**-l** <location> Instead of `-a`, `-o` and `-s` define a location. possible locations are ifu, dlrop.

**-y** <yyyy> year; not used if -S specified, default: 2003.

**-S** Use the Spencer algorithm.

**-h** Print help message.

The following invocation of `noon` calculates the noon time at the home location of one of the *libRadtran* developers for his wedding date.

```
noon -a 62.462052 -o -6.303358 -s -15 -y 1992 29 2
```

### 5.2.5 Angular response and tilted surfaces - `angres`

The `angres` tool takes a precalculated radiance field and integrates it over a given angular area using any angular response. Typical usages of `angres` are calculation of radiation on tilted surfaces and estimation of effects of imperfect angular response functions.

The `angres` tool is invoked as follows:

```
angres angres_file raddis_file
```

The two required input files will be read by the `angres` tool.

**angres_file** is a two column file with the first column holding the angle and the second column the angular response, e.g. a measured cosine response. To generate standard angular response function see the `make_angres_func` tool.

**raddis_file** holds the radiance distribution as output from uvspec with the disort solvers for one single wavelength.

After reading the two input files the angular response will be tilted and rotated if specified with the `-t` and `-r` options respectively. Finally the product of the resulting angular response and radiance distribution field are integrated using Monte Carlo methods to yield the effective response. The integration is done for the diffuse radiation field only. To include the direct contribution the -s and -z options must be set to give the direction of the sun.

Output is 3 numbers:

1. The integral of the diffuse radiation field times angular response.
2. Estimated absolute error of the above integral.
3. The integral of the diffuse+direct radiation field times angular response (requires that `-s` and `-z` are specified, otherwise same as first number

The angles in the `angres_file` must be in radians if not the `-a` option is used. The `raddis_file` must contain output from `uvspec` run for one single wavelength with one of the disort solvers and with `phi` and `umu` set. Note that the angles in the `angres_file` must follow the same conventions as for the disort algorithm. This is different from that typically used when reporting measurements of the angular response.

The `angres` tool accepts the following command line options:

**-h** show this page.

**-c** number of random points used for Monte Carlo integration.

**-i** The diffuse radiation is assumed to be isotropic.

**-a** angular response angle given in degrees and not cosine of angle.

**-r** rotation angle in degrees.

**-t** tilt angle in degrees.

**-s** solar zenith angle in degrees.

**-z** solar azimuth angle in degrees.

**-p** pgm files are made of the angular response before and after tilt and rotate.

Sample `angres` input and output files are found in the `examples` directory. The following

```
angres examples/ANGRES_1_ANG.DAT  \
       examples/ANGRES_RADDIS_1.DAT -a -t -r 0 -s 32 -z 0
```

calculates the radiation on a horisontal surface given the angular response in `examples/ANGRES_1_ANG.DAT`. The input used to calculate the radiance file is given in the start of `examples/ANGRES_RADDIS_1.DAT`.

An example of the use of `angres` together with `uvspec` is given in Mayer and Kylling (2005, section 4.6).

### 5.2.6 Angular response function - `make_angresfunc`

The `make_angresfunc` tool calculates various angular response functions to be used by for example the `angres` tool. All output is to stdout in two column format. The first column is the angle and the second column contains the corresponding value for a given angular response. The output angles follow `disort` conventions.

The `make_angresfunc` tool is invoked on the command line as

```
    make_angresfunc [-hart]
```

where the various options are

**-t** type of angular response

1. cosine (default)
2. 2pi actinic flux
3. 4pi actinic flux

**-a** angular output format

1. angles (default)
2. cosine of angle

**-r** resolution, in degrees

**-h** Print help message.

The following invocation of `make_angresfunc` calculates the angular response for a perfect cosine detector. The output is found in the `examples/ANGRES_1_ANG.DAT`.

```
make_angresfunc -t 1 -r 1
```

### 5.2.7 Slit function generator - `make_slitfunction`

To generate standard slit functions to be used by `uvspec` the `make_slitfunction` tool may be used. For a given set of input it outputs to stdout in two column format the wavelength and corresponding value for the wanted slit function.

The `make_slitfunction` tool is invoked on the command line as

```
make_angresfunc [-hrtf]
```

where the various options are

**-t** type of slitfunction

1. triangular (default)
2. rectangular

**-f** full width at half maximum, in nm

**-r** resolution, in nm

**-h** Print help message.

The following invocation of `make_slitfunction` calculates the a triangular slit function with FWHM of 0.75 nm and a resolution of 0.01 nm. The output is found in the `examples/TRI_SLIT.DAT`.

```
make_slitfunction -f 0.75 -r 0.01 -t 1
```

### 5.2.8 Calculate phase function from Legendre polynomials - `phase`

The `phase` tool takes a Legendre series as input and calculates the corresponding phase function.

The program is invoked as follows:

```
phase [options] <filename>
```

The following optional arguments may be specified:

**-h** Display help message.

**-c** 1-column input.

**-b** Binary (netcdf) input.

**-d** Use scattering angle in degrees instead of the cosine of the scattering angle $\mu$.

**-s** <**step**> Step width for evaluation (default: 0.01).

**-o** <number of digits> Optimize scattering angle grid.

**-x** <**filename**> File containing $\mu$-values to be interpolated.

**-n** Normalize phase function.

**-f** Use delta scaling.

The format of the input file is as generated by the `mie` program. the first 7 columns are ignored, the following columns are assumed to hold the moments of the phase function. If option `-c` is specified, the input file is considered a one column file holding one moment per line.

### 5.2.9 Perform Legendre decomposition of phase function - `pmom`

The `pmom` tool calculates the Legendre moments of a given phase function. The input must be provided as 2-column file, containing the scattering angle grid in the first column and the phase function value in the second column. The output of `pmom` are the Legendre moments.

The `pmom` tool is invoked on the command for instance as

```
pmom [options] <filename>
```

The following optional arguments may be specified:

**-h** Display help message.

**-l** <**number**> Number of Legendre moments to be computed. In order to obtain an accurate decomposition of the phase function, the last terms of the Legendre series should approach 0.

**-r** <**grid**> Specify scattering angle grid which is used internally (see below for more explanation).

**-c** Calculate coefficients instead of polynomials (these include the factor $(2l+1)$. `uvspec` requires Legendre coefficients.

**-n** Normalize the phase function before computing the Legendre moments.

You may specify the number of moments using the option `-l`. Different scattering angle grid resolutions can be chosen using the option `-r`. For moderate forward peaks, the standard grid (`-r 1` - equidistant, 0.01 degrees step width) should be sufficient. For phase functions with a very strong forward peak, e.g. ice particle phase functions, the finest grid resolution (`-r 2` - equidistant, 0.001 degrees step width) should be specified. If the grid of the input file should be used for the Legendre decomposition, please use `-r 3`; this option uses the new speedy and exact method for Legendre decomposition (Buras, Dowling, Emde 201X). Default. You may test `-r 4` and `-r 5`, in this case non-equdistant grids with a finer resolution around the forward peak are used.

You may test the accuracy of the Legendre decomposition by using the tool `phase`:

```
phase -c -d -s 1 pmom_outfile.dat
```

## 5.3 Other useful tools

### 5.3.1 Stamnes tables for ozone and cloud optical depth

Stamnes et al. (1991) devised a method to derive the total ozone column and cloud optical depth from global irradiance measurements. For ozone column retrieval this method requires a table of irradiance ratios as a function of solar zenith angle and ozone column. The irradiance ratio is taken as the ratio of irradiances at non-absorbing and ozone-absorbing wavelengths. The cloud optical depth is retrieved from tables of cloud/cloudless irradiance ratios as a function of solar zenith angle and water cloud optical depth.

The *libRadtran* package comes with three tools for calculation and reading of these so-called Stamnes tables. The Perl script `Gen_o3_tab.pl` is used to generate a matrix of ozone values for solar zenith angle versus a chosen ratio of global irradiances at different wavelengths. For cloud optical depths the Perl script `Gen_wc_tab.pl` may be used to generate a matrix of cloud optical depth for solar zenith angle versus a chosen global irradiance at a selected wavelength. Both tables may be read by the C program `read_Stamnes_tab` which, for a solar zenith angle and a measured ratio, returns the overhead ozone column or cloud optical depth. The Perl scripts `Gen_o3_tab.pl` and `Gen_wc_tab.pl` and the C program are briefly described below. For example of their use please see Mayer et al. (1998); Kylling et al. (2005); Mayer and Kylling (2005).

#### Generation of the Stamnes ozone column table- `Gen_o3_tab`

The Perl script `Gen_o3_tab.pl` is used to generate a matrix of ozone values for solar zenith angle versus a chosen ratio of global irradiance at different wavelengths. The table is read by the C program `read_Stamnes_tab` which, for a solar zenith angle and a measured irradiance ratio, returns the overhead ozone column. The following options are understood by `Gen_o3_tab.pl`:

**–absolute** The wavelengths in the bandpass files are in absolute units. Default is relative units.

**–albedo** <**value**> Lambertian surface albedo. Default is 0.0.

**–alpha** <**value**> Angstrom alpha coefficient. Default is 0.0.

**–beta** <**value**> Angstrom beta coefficient. Default is 0.0.

**–altitude** <**value**> Altitude above sea level [km]. Default is 0.0.

**–atmmod** <**name**> Name of atmosphere file. Default atmmod/afglus.dat.

**–help** Prints help message.

**–o3_crs** <**name**> Name of o3 cross section to use. Default is Molina. See `uvspec` documentation for other options.

**–slitfunction** <**name**> Name of slitfunction file.

**–bandpasslower** <**name**> Name of file holding bandpass for lower wavelength.

**–bandpassupper** <**name**> Name of file holding bandpass for upper wavelength.

**–file** <**name**> Name of file where the table will be stored.

**–lower_lambda** <**value**> Value for lower wavelength, in nm.

**–upper_lambda** <**value**> Value for upper wavelength, in nm.

**–zenith** Calculate zenith sky radiance table.

Two different types of tables may be generated depending on the measurement type and the preferred analysis method.

**Simple wavelength ratios with `Gen_o3_tab`** The simplest type of table is made of ratios of the global irradiance at two single wavelengths. This is the type of table described by Stamnes et al. (1991) and it is typically used to analyse measurements of the global irradiance from spectroradiometers. It is generated by the following command ( is line continuation character)

```
perl Gen_o3_tab.pl --slitfunction slitfncfile --lower_lambda 305. \
--upper_lambda 340. --file table.dat
```

Here `slitfncfile` is the name of the slit function file. It is a two column file where the first column is the wavelength (nm, in relative units) and the second column holds the slit function. The slit function must be normalized to unity at the center wavelength.

The generated table `table.dat` is read by `read_Stamnes_tab` for a measured ratio, `-r 10.0`, and solar zenith angle, `-s 30.0`, corresponding to the modelled ratio in the table

```
read_Stamnes_tab -r 10.0 -s 30.0 table.dat
```

**Bandpassed wavelength ratios with `Gen_o3_tab`** Instead of using single wavelengths it may be of advantage to use ratios of irradiances covering a certain wavelength range and weighted with a bandpass function. This approach may reduce problems due to changes in cloud cover and experimental uncertainties. This approach is also suitable to calculate ozone columns from multichannel, moderate bandwidth filter instruments (Dahlback, 1996). Such tables are generated by

```
perl Gen_o3_tab.pl --slitfunction slitfncfile  --lower_lambda 305.0 \
                   --upper_lambda 320.0 --file table.dat \
                   --bandpasslower bplow.dat --bandpassupper bpupp.dat
```

Here `bplow.dat` and `bpupp.dat` are the bandpass function of the lower and upper wavelength region respectively. The bandpass files have two columns. The first column is the wavelength in nm and relative units to `--lower_lambda` and `--upper_lambda`. If absolute units are specified as for filter instruments, use the `--absolute option`. The second column is the bandpass function.

The tables are read in the same way as the simple wavelength ratio tables.

### Generation of the Stamnes cloud optical thickness table - `Gen_wc_tab`

The Perl script `Gen_wc_tab.pl` is used to generate a matrix of cloud optical depth for solar zenith angle versus a chosen global irradiance at a selected wavelength. The wavelength should be chosen such that it is not affected by ozone, e.g. 380 nm. The table is read by the C program `read_Stamnes_tab` which, for a solar zenith angle and a measured irradiance, returns the overhead cloud optical depth. The available options are

**–absolute** The wavelengths in the bandpass file are in absolute units. Default is relative units.

**–albedo** <**value**> Lambertian surface albedo. Default is 0.0.

**–alpha** <**value**> Angstrom alpha coefficient. Default is 0.0.

**–beta** <**value**> Angstrom beta coefficient. Default is 0.0.

**–altitude** <**value**> Altitude above sea level [km]. Default is 0.0.

**–atmmod** <**name**> Name of atmosphere file. Default atmmod/afglus.dat.

**–help** Prints help message.

**–o3_crs** <**name**> Name of o3 cross section to use. Default is Molina. See `uvspec` documentation for other options.

**–slitfunction** <**name**> Name of slitfunction file.

**–bandpass** <**name**> Name of file holding bandpass for chosen wavelength.

**–file** <**name**> Name of file where the table will be stored.

**–lambda** <**value**> Value of chosen wavelength, in nm.

**–wc_file** <**name**> Name of water cloud file. Default none. Must be specified.

The following different types of tables may be generated.

**Simple wavelength ratios with `Gen_wc_tab`** The simplest type of table is made of the global irradiance at a single wavelength. This is the type of table described by Stamnes et al. (1991). This type of table is typically used to analyse measurements of the global irradiance from spectroradiometers. It is generated by the following command ( is line continuation character)

```
perl Gen_wc_tab.pl --slitfunction slitfncfile --lambda 380. \
                   --file table.dat --wc_file ../examples/WC.DAT
```

Here `slitfncfile` is the name of the slit function file. It is a two column file where the first column is the wavelength (nm, in relative units) and the second column holds the slit function. The slit function must be normalized to unity at the center wavelength.

The generated table `table.dat` is read by `read_Stamnes_tab` for a measured global irradiance, `-r 10.0`, and solar zenith angle, `-s 30.0`, corresponding to the modelled ratio in the table. The table must be corrected for the Earth–Sun distance for the day of the measurement. This is achieved by specifying `-d 170`, where `170` is the day number. The table is generated for day 1.

```
read_o3_tab -r 10.0 -s 30.0 -d 170 table.dat
```

**Bandpassed wavelength ratios with Gen_wc_tab** Instead of using a single wavelength it may be of advantage to use irradiances covering a certain wavelength range and weighted with a bandpass function. This approach may reduce problems due to changes in cloud cover and experimental uncertainties. This approach is also suitable to calculate cloud optical depth from multichannel, moderate bandwidth filter instruments (Dahlback, 1996). Such tables are generated by

```
perl Gen_wc_tab.pl --slitfunction slitfncfile  --lambda 380.0 \
                   --file table.dat --bandpass bp.dat
```

Here `bp.dat` is the bandpass function of the wavelength region. The bandpass file have two columns. The first column is the wavelength in nm and relative units to –lambda. If absolute units are specified as for filter instruments, use the –absolute option. The second column is the bandpass function.

The tables are read in the same way as the simple wavelength irradiance tables.

# Chapter 6

# Complete description of input options

## 6.1 Radiative transfer tool - `uvspec`

The `uvspec` input file consists of single line entries, each making up a complete input to the `uvspec` program. First on the line comes the parameter name, followed by one or more parameter values. The parameter name and the parameter values are separated by white space. Filenames are entered without any surrounding single or double quotes. Comments are introduced by a `#`. Blank lines are ignored. The order of the lines is not important, with one exception: if the same input option is used more than once, the second one will usually over-write the first one. Be aware that also options in another `included` input file will overwrite options specified before.

The various input parameters are described in detail below.

**absorption**

Switch off absorption by individual minor trace gases which are currently only included when `correlated_k lowtran` is chosen. The syntax is

```
absorption species on/off
```

where species may be one of O4, N2, CO, SO2, NH3, NO, HNO3. By default all are switched on.

**aerosol_angstrom**

Scale the aerosol optical depth using the Ångström formula. Specify the Ångström alpha and beta coefficients by

```
aerosol_angstrom alpha beta
```

The optical thickness defined here is the integral from the user-defined `altitude` to TOA (top of atmosphere).

**aerosol_default**

Set up a default aerosol according to Shettle (1989). The default properties are a rural type aerosol in the boundary layer, background aerosol above 2km, spring-summer conditions and a visibility of 50km. These settings may be modified with `aerosol_haze`, `aerosol_vulcan`, `aerosol_season`, and `aerosol_visibility`.

**aerosol_files**

A way to specify aerosol optical depth, single scattering albedo, and phase function moments for each layer. The file specified by `aerosol_files` by

```
aerosol_files file_name
```

has two columns where column 1 is the altitude in km. The second column is a the name of a file which defines the optical properties of the layer starting at the given altitude. The files specified in the second column must have the following format:

**Column 1:**
The wavelength in nm. These wavelengths may be different from those in `solar_file`. Optical properties are interpolated to the requested wavelengths.

**Column 2:**
The extinction coefficient of the layer in units km-1.

**Column 3:**
The aerosol single scattering albedo of the layer.

**Column 4-(nmom+4):**
The moments of the aerosol phase function.

For some simple examples see the files `examples/AERO_*.LAYER`. Note that if using the `rte_solver cdisort` it makes good sense to make the number of moments larger than `nstr`. For `rte_solver fdisort1` and `rte_solver polradtran` the number of moments included in the calculations will be `nstr`+1. Higher order moments will be ignored for these solvers. Please note that the uppermost line of the `aerosol_files` denotes simply the top altitude of the uppermost layer. The optical properties of this line are consequently ignored. There are two options for this line: either an optical property file with zero optical thickness is specified or "NULL" us used.

**aerosol_gg_file**

Location of aerosol asymmetry parameter file specified by

```
aerosol_gg_file file_name
```

The file must have two columns. Column 1 is the altitude in km. Column 2 is the asymmetry parameter of each layer. The asymmetry parameter defined with this option is constant with wavelength. If you require spectral dependence please use `aerosol_files`. Comments start with `#`. Empty lines are ignored.

**aerosol_haze**

Specify the aerosol type in the lower 2 km of the atmosphere as

```
aerosol_haze type
```

where `type` is an integer identifying the following aerosol types:

**1** Rural type aerosols.

**2** Maritime type aerosols.

**5** Urban type aerosols.

**6** Tropospheric type aerosols.

For a description of the different aerosol types see Shettle (1989).

**aerosol_moments_file**

Set the aerosol phase function moments to the values specified in the aerosol moments file

```
aerosol_moments_file file_name
```

where the file contains one column with arbitrary number of Legendre terms of the phase function. The phase function $p(\mu)$ is

$$p(\mu) = \sum_{m=0}^{\infty} (2m+1) \cdot k_m \cdot P_m(\mu) \tag{6.1}$$

where $k_m$ is the m'th moment and $P_m(\mu)$ is the m'th Legendre polynomial. If not specified, a Henyey-Greenstein phase function is assumed where the asymmetry parameter g is either a default value depending on the aerosol type or it may be specified using `aerosol_set_gg`. The phase function will be the same for all altitudes and wavelengths. See `aerosol_files` if more flexibility is wanted. May only be used together with the `cdisort` or `fdisort2` solver in combination with the option `disort_icm moments`.

**aerosol_no_scattering**

Switch off scattering by aerosols.

**aerosol_refrac_file**

The command line

```
aerosol_refrac_file file_name
```

specifies the file containing the wavelength-dependent refractive index of the aerosol. Three columns are expected: wavelength [nm] and the real and imaginary parts of the refractive index. Together with `aerosol_sizedist_file` this forms the input to Mie calculations of the aerosol optical properties. Please note that only the single-scattering albedo, the scattering phase function, and the wavelength-dependence of the extinction coefficient are affected by the Mie calculation while

the absolute value of the extinction coefficient is taken from other sources; generally, the extinction coefficient at the first *internal* wavelength is taken from whatever is available (either default Shettle (1989) or user-defined); the extinction at all other wavelengths is scaled according to the Mie calculation. For this reason, the absolute numbers are not relevant - only the shape of the size distribution matters. In detail: If the aerosol properties are defined using the refractive index and the size distribution, the wavelength dependence of the optical properties is determined by Mie theory. At present there are at least three ways to define the absolute value of the optical thickness: (1) `visibility` defines the profile at the first *internal* wavelength; for a monochromatic calculation and in correlated-k mode, the first *internal* wavelength equals the first wavelength output by `uvspec`; for spectral calculations, the first wavelength might be a little bit smaller than the first wavelength output by `uvspec`; (2) `aerosol_tau_file` defines the optical thickness profile at the first *internal* wavelength; or (3) absolute optical thickness and wavelength-dependence are defined by `aerosol_angstrom`. *It is recommended to avoid this option and rather to calculate the aerosol optical properties externally e.g. with mie and to pass them to `uvspec` with `aerosol_files`.*

**aerosol_refrac_index**

Wavelength-independent refractive index of the aerosol; if wavelength-dependence is required, use `aerosol_refrac_file` instead. Together with `aerosol_sizedist_file` this forms the input to Mie calculations of the aerosol optical properties. Please see the description of `aerosol_refrac_file` to learn how the optical properties are set up. *It is recommended to avoid this option and rather to calculate the aerosol optical properties externally e.g. with mie and to pass them to `uvspec` with `aerosol_files`.*

**aerosol_scale_ssa**

Scale the aerosol single scattering albedo for all wavelengths and altitudes with a positive number. If the resulting scaled single scattering albedo is larger than 1 it is set to 1.

```
aerosol_scale_ssa value
```

**aerosol_scale_tau**

Scale the aerosol extinction for all wavelengths and altitudes with a positive number.

```
aerosol_scale_tau value
```

**aerosol_set_gg**

Set the aerosol asymmetry parameter for all wavelengths and altitudes to a constant value between -1.0 and 1.0. Please note that this option is only applied if a Henyey-Greenstein phase function is used but not if an explicit phase function is defined e.g. with a aerosol_files. It doesn't make sense to modify only the first moment of an

explicit phase function.

```
aerosol_set_gg value
```

**aerosol_set_ssa**
Set the aerosol single scattering albedo for all wavelengths and altitudes to a constant value between 0.0 and 1.0.

```
aerosol_set_ssa value
```

**aerosol_set_tau**
Set the aerosol optical thickness for all wavelengths and altitudes to a constant value. The optical thickness defined here is the integral from the user-definded `altitude` to TOA (top of atmosphere).

```
aerosol_set_tau value
```

**aerosol_set_tau550**
Set the aerosol optical thickness at 550nm. Other wavelengths are scaled accordingly. Note that this option requires for technical reasons that the wavelength interval defined by `wavelength` does contain 550nm. The optical thickness defined here is the integral from the user-definded `altitude` to TOA (top of atmosphere).

```
aerosol_set_tau550 value
```

**aerosol_season**
Specify season to get appropriate aerosol profile.

```
aerosol_season season
```

where `season` is either `1` or `2`:

**1** Spring-summer profile.

**2** Fall-winter profile.

**aerosol_sizedist_file**
Aerosol size distribution. Two columns are expected: The radius [micrometer] and the particle number. Together with `aerosol_refrac_index` or `aerosol_refrac_file` this forms the input to Mie calculations of the aerosol optical properties. Please note that only the single-scattering albedo, the scattering phase function, and the wavelength-dependence of the extinction coefficient are affected by the Mie calculation while the absolute value of the extinction coefficient is taken from

other sources; generally, the extinction coefficient at the first *internal* wavelength is taken from whatever is available (either default Shettle (1989) or user-defined); the extinction at all other wavelengths is scaled according to the Mie calculation. For this reason, the absolute numbers are not relevant - only the shape of the size distribution matters. For details see also the description of `aerosol_refrac_file`. *It is recommended to avoid this option and rather to calculate the aerosol optical properties externally e.g. with mie and to pass them to* `uvspec` *with* `aerosol_files`.

**aerosol_species_file**

Specify mass density profiles of a mixture of aerosol types.

```
aerosol_species_file profile [aero_1 aero_2 ...  aero_n]
```

where `aero_1` to `aero_n` are the aerosol species to be included. For each of these species, the optical properties are read from the `aerosol_species_library`, e.g. the OPAC data set provided with libRadtran. The profile file needs to include vertical profiles for each of these species. This file can be either in *netCDF*-format (automatically recognized filename extension `.nc` or `.cdf`) or in ASCII format. The format of the ASCII file is:

```
 z1 dens(aero_1, z1) dens(aero_2, z1) ...  dens(aero_n, z1)
z2 dens(aero_1, z2) dens(aero_2, z2) ...
.  .  .
.  .  .
```

where `z` is the height in km, and `dens` are the aerosol mass densities in g/m3. Please make sure to include one column for each of the species `aero_1` to `aero_n` listed after `aerosol_species_file`. For netCDF input it is also possible to specify the unit 'kg kg$^{-1}$'; the data are then automatically converted to g/m$^3$.

Some default aerosol mixtures are provided, corresponding to the definitions in Hess et al. (1998). They can simply be invoked by

```
aerosol_species_file mixture_name
```

where `mixture_name` can be one of the following:

```
 continental_clean
continental_average
continental_polluted
urban
maritime_clean
maritime_polluted
maritime_tropical
desert
antarctic
```

**aerosol_species_library**

With this option the *directory* is specified where the optical property files for all aerosols species used in the `aerosol_species_file` are expected: For each species defined in `aerosol_species_file`, *netCDF*-file *species_name*`.nc`, (e.g. `INSO.nc`), which contains the optical properties of the aerosol species, has to be provided. The netcdf format is the one produced by the *libRadtran* `mie` tool.

At the libRadtran webpage we provide the OPAC data set (Hess et al., 1998) which can be directly used with `uvspec`:

```
aerosol_species_library OPAC
```

OPAC contains following aerosol species:

```
 INSO insoluble
WASO water_soluble
SOOT soot
SSAM sea_salt_accumulation_mode
SSCM sea_salt_coarse_mode
MINM mineral_nucleation_mode
MIAM mineral_accumulation_mode
MICM mineral_coarse_mode
MITR mineral_transported
SUSO sulfate_droplets
```

**aerosol_ssa_file**

Location of aerosol single scattering albedo file.

```
aerosol_ssa_file file
```

The file must have two columns. Column 1 is the altitude in km. The altitude grid must be exactly equal to the altitude grid specified in the file `atmosphere_file`. Column 2 is the single scattering albedo of each layer. The single scattering albedo defined with this option is constant with wavelength. If you require spectral dependence please use `aerosol_files`. Comments start with `#`. Empty lines are ignored.

**aerosol_tau_file**

Location of aerosol optical depth file.

```
aerosol_tau_file file
```

The file must have two columns. Column 1 is the altitude in km. The altitude grid must be exactly equal to the altitude grid specified in the file `atmosphere_file`. Column 2 is the aerosol optical depth of each layer. To allow wavelength-dependent aerosol optical thickness please use either `aerosol_angstrom` or `aerosol_files`. Comments start with `#`. Empty lines are ignored.

**aerosol_visibility**

Horizontal visibility in km. Affects the profile according to Shettle (1989) and the

optical thickness.

```
aerosol_visibility value
```

**aerosol_vulcan**

Aerosol situation above 2 km as defined in Shettle (1989).

```
aerosol_vulcan value
```

where `value` is an integer choosing between the following models

**1** Background aerosols.

**2** Moderate vulcanic aerosols.

**3** High vulcanic aerosols.

**4** Extreme vulcanic aerosols.

**albedo**

The Lambertian surface albedo

```
albedo value
```

where `value` is a number between 0.0 and 1.0, constant for all wavelengths. For wavelength dependent surface albedo use `albedo_file`. The default albedo is 0.0.

**albedo_file**

Location of surface albedo file for wavelength dependent surface albedo.

```
albedo_file file
```

The file must have two columns. Column 1 is the wavelength in nm, and column 2 the corresponding Lambertian surface albedo. An arbitrary wavelength grid may be chosen as the albedo will be interpolated linearely to the wavelength grid used for the radiation calculation. Comments start with `#`. Empty lines are ignored. A large collection of spectral albedos are available e.g. at http://speclib.jpl.nasa.gov/ (Baldridge et al., 2009).

**albedo_library**

Albedo libraries are a collection of spectral albedos of different surface types. This option must be used either with `surface_type` or `surface_type_map`, in order to select the specific surface type. There are two possibilities for libraries: the built-in IGBP library or a user defined albedo library.

The built-in library of the International Geosphere Biosphere Programme is selected with

```
albedo_library IGBP
```

The IGBP library contains 20 surface types which are set by `surface_type`:

```
 1 evergreen_needle_forest
2 evergreen_broad_forest
3 deciduous_needle_forest
4 deciduous_broad_forest
5 mixed_forest
6 closed_shrub
7 open_shrubs
8 woody_savanna
9 savanna
10 grassland
11 wetland
12 cropland
13 urban
14 crop_mosaic
15 antarctic_snow
16 desert
17 ocean_water
18 tundra
19 fresh_snow
20 sea_ice
```

Surface types 1 - 17 are defined by the International Geosphere Biosphere Programme (IGBP); additionally there are tundra, fresh_snow, and sea_ice surface types. The spectral albedo of the ground is determined as a function of solar zenith angle, precitable water, and clouds. The spectral resolution equals the grid of the correlated-k Fu/Liou parameterisation. This library originates from the NASA CERES/SARB Surface Properties Project, see Belward and Loveland (1996).

For creating your own albedo library use `albedo_library path`, where `path` is the path of the directory where the albedo data is stored. The files are expected to have the names `albedo_01.dat, albedo_02.dat, ...` If `surface_type 1` is specified the albedo from `albedo_01.dat` will be used, and so on. Each file is required to have two columns: Column 1 is the wavelength in nm, and column 2 the corresponding Lambertian surface albedo. The wavelength grid may be freely set. The albedo will be interpolated linearely to the wavelength grid used for the radiation calculation. Comments start with `#`. Empty lines are ignored. This option is similar to `albedo_file`, except that it offers an easy way to use the option `surface_type_map` in combinition with albedo files.

**albedo_map**

*This option is preliminary and still subject to change (no wavelength dependency yet)!* It gives the possibility to specify a wavelength independent albedo with the help of a *netCDF* file, which is used in combinition with the options `latitude`, `longitude`, and `time`.

```
albedo_map file [variable_name]
```

Here `file` is the location of the *netCDF* file, The optional argument allows

the name of the albedo variable in the *netCDF* file to be specified (the default name is `AL`). The albedo must be provided as function of `latitude` and `longitude` `AL(lat, lon)`, and may also depend on time `AL(time, lat, lon)`. The latitude, longitude, and time grids must be provided as doubles `double lat(lat)`, `double lon(lon)`, and `double time(time)`. `uvspec` reads the value at the nearest pixel to the given `latitude` and `longitude`. No spatial interpolation or averaging of the values are performed. If a time-dependent albedo is provided, the albedo data nearest to the specified time will be selected (or linear interpolated if `time_interpolate` is switched on).

**altitude**

Set the bottom level in the model atmosphere provided in `atmosphere_file` to be at the given altitude above sea level (km).

```
altitude 0.73 # Altitude of IFU, Garmisch-Partenkirchen
```

The profiles of pressure, temperature, molecular absorbers, ice and water clouds are cut at the specified altitude. The aerosol profile is not affected by `altitude` but starts right from the model surface. This is a convenient way for the user to calculate the radiation at other altitudes than sealevel. Note that `altitude` is very different from `zout` where the radiation is calculated at an altitude of zout above the surface. E.g. to calculate the radiation field 1 km above the surface at a location at 0.73 km above sealevel, one would specify '`altitude` 0.73' and '`zout` 1.0'. If an altitude is specified which is below the lowest level in the `atmosphere_file`, the atmospheric profiles are extrapolated assuming a constant gradient for temperature and mixing ratios. A second optional argument may be given to `altitude` as e.g.

```
altitude 0.73 0.5
```

Here the bottom level will be at 0.73 km and the vertical resolution of the model atmosphere will be redistributed to have a spacing between levels specified by the second number, here 0.5 km, starting however from 0km. (Levels 0.73, 1., 1.5 ... will be added to the original atmosphere grid and optical properties are devided into the new layers. In order to use interpolated properties use `zout_interpolate`. See verbose output for details.) Be aware that specifying a fine vertical spacing will produce many layers thus increasing the computing time. Also the radiative transfer equation solvers implemented in Fortran 77 might need to have some array sizes increased (see `src_f/DISORT.MXD`).

**altitude_map**

Specifies an altitude map which is used in combinition with `latitude`, `longitude` in order to select the altitude for the simulation. No interpolation is done between the pixels of the map. The format of the call is:

```
altitude_map file [variable_name]
```

where `file` is the location of the altitude map file. The map is expected to be in

*netCDF* format. The file must contain `double lat(lat)`, `double lon(lon)`, and the altitude variable, where `variable_name` is the name of the surface elevation variable in the *netCDF* file. The default name is Z. The altitude variable must be `altitude(lat, lon)`. For format discribtion see also the example map included in *libRadtran*, `data/altitude/ELEVATION_GTOPO_10min.cdf`. To use this map in `uvspec`, you may also use `altitude_map GTOPO`. This map has a resolution of 10 arc minutes and the unit of the altitude is meter. Please note that this resolution might not ne adequate for your application. If an altitude in the map is below the lowest level of the `atmosphere_file`, the atmospheric profiles are extrapolated assuming a constant gradient for temperature and mixing ratios.

**angstrom**

Deprecated option. Same as `aerosol_angstrom`.

**atmosphere_file**

Location of the atmospheric data file.

```
atmosphere_file file
```

The file must have at least three columns containing the altitude, pressure, and temperature. Missing profiles are filled with 0 (e.g., if you did not specify the ozone profile, there will be no ozone absorption!), with exception of the air density which is calculated from pressure and temperature. Other traces gases may be set by `dens_file`. The columns are interpreted as follows:

**1** Altitude above sea level in km

**2** Pressure in hPa

**3** Temperature in K

**4** air density in $\mathrm{cm}^{-3}$

**5** Ozone density in $\mathrm{cm}^{-3}$

**6** Oxygen density in $\mathrm{cm}^{-3}$

**7** Water vapour density in $\mathrm{cm}^{-3}$

**8** CO2 density in $\mathrm{cm}^{-3}$

**9** NO2 density in $\mathrm{cm}^{-3}$

The atmosphere is specified top-down, that is, the top level is the first line in the file, the bottom (surface) level the last line. All properties refer to model *level* z, not to model *layer*. It is important that the correct units are used, otherwise unpredictable results are guaranteed. Comments start with `#`. Empty lines are ignored. Please note that there is some redundancy: For air as an ideal gas the density $\rho$, can be calulated from pressure and temperature, $\rho = p/kT$. `uvspec` will check if this relation is fulfilled and will stop if it is not. *libRadtran* provides the six standard atmospheres by Anderson et al. (1986):

**afglt** Tropical (`tropics`)

**afglms** Midlatitude Summer (`midlatitude_summer`)

**afglmw** Midlatitude Winter (`midlatitude_winter`)

**afglss** Subarctic Summer (`subarctic_summer`)

**afglsw** Subarctic Winter (`subarctic_winter`)

**afglus** U.S. Standard (`US-standard`)

which may be chosen by for example

```
atmosphere_file tropics
```

or by specifying the the full file name. These atmosphere files are found in `data/atmmod`. If no `atmosphere_file` is defined, `uvspec` will automatically select one. If the information `time`, `latitude` and `longitude` are provided in the input file `uvspec` will choose from the first 5 files, otherwise it takes the U.S. Standard atmosphere.

**atm_z_grid**
With this option the vertical resolution of the `atmosphere_file` data is changed to the levels (in km above sea surface) given as argument. This might be useful in oder to reduce the number of levels (save computational time) .

```
atm_z_grid 0 2 4 6 8 10 20 30 ...
```

**brdf_ambrals**
AMBRALS BRDF, a three-parameter BRDF fit for vegetated and non-vegetated surfaces (Wanner et al., 1997).

```
brdf_ambrals iso_value vol_value geo_value
```

Specify iso, vol, and geo. May be combined with , `cdisort`, and `fdisort2`.

**brightness**
Convert radiances / irradiances to equivalent brightness temperatures.

**cdisort_pseudospherical**
Pseudo-spherical geometry in cdisort. Default is plane-parallel.

**ch4_mixing_ratio**
The mixing ratio of CH4 in ppm (default: 1.6 ppm).

```
ch4_mixing_ratio value
```

**cloud_fraction_file**
File containing a cloud fraction profile.

```
cloud_fraction_file file
```

Two columns are expected: altitude [km] and cloud fraction, including ice and water clouds. If `cloud_fraction_file` is defined, effective cloud properties are calculated assuming either random overlap or maximum random overlap of the cloud layers (see also `cloud_overlap`). An example is provided in `examples/CF.DAT`.

**cloud_overlap**
Cloud overlap assumption.

```
cloud_overlap type
```

Following types are implemented:

**rand** Random overlap of cloud layers

**maxrand** Maximum random overlap scheme

**max** Maximum overlap scheme

**off** Turn off cloud overlap for ECMWF clouds

Per default the `cloud_overlap` scheme is switched off.

**co2_mixing_ratio**
The mixing ratio of CO2 in ppm.

```
co2_mixing_ratio value
```

The profile is scaled so that the mixing ratio at the user-defined `altitude` assumes the specified value.

**correlated_k**
To calculate integrated shortwave or longwave irradiance, or to simulate satellite instrument channels, use

```
correlated_k type
```

to choose between the following types of correlated-k schemes:

**kato** Kato et al. (1999) correlated-k distribution, shortwave; based on HITRAN 96. Please note that the bands above 2.5 micrometer are not very reliable which, however, this has only little impact on integrated shortwave radiation.

**kato2** Kato et al. (1999), shortwave; optimized version (Seiji Kato, personal communication, 2003); please note that `kato2` only has 148 subbands (that is, calls to the `rte_solver`) compared to 575 for `kato` which translates to a decrease in computational speed by up to a factor of 4 with only little increase in uncertainty. The absorption data are based on HITRAN 2000. Please note that the bands above 2.5 micrometer are not very reliable which, however, this has only little impact on integrated shortwave radiation.

**kato2.96** Kato et al. (1999), shortwave; optimized version (Seiji Kato, personal communication, 2003); similar to `kato2` but based on HITRAN96. Please note that

the bands above 2.5 micrometer are not very reliable which, however, has only little impact on integrated shortwave radiation.

**fu** Fu and Liou (1992, 1993), shortwave and longwave; fast parameterization, developed for climate models.

**avhrr_kratz** Kratz and Varanasi (1995), AVHRR instrument channels

**lowtran** Gas absorption parameterization from LOWTRAN; code adopted from SBDART (Ricchiazzi et al., 1998); please see the section on "Spectral resolution".

**sbdart** Identical to LOWTRAN.

If `correlated_k` kato/kato2/kato2.96/fu/avhrr_kratz is specified, the extraterrestrial flux is taken from internally defined files specific for each parameterization, not from `solar_file`. The output is the integrated irradiance for each band. To get e.g. integrated shortwave irradiance, simply add all bands of the Kato et al. (1999) or the Fu and Liou (1992, 1993) parameterization. The five AVHRR channels are weighted sums of the libRadtran output. Examples how to integrate the output in the `avhrr_kratz` case are included in the `uvspec` self check which is initiated with `make check`.

**cox_and_munk_pcl**

Pigment concentration for Cox and Munk (1954a,b) ocean BRDF (in mg/m$^{-3}$).

```
cox_and_munk_pcl value
```

At present only available with `rte_solver cdisort`, and `rte_solver fdisort2`. The number of streams (`nstr`) is automatically increased to 16 if `cox_and_munk` BRDF is switched on, to avoid numerical problems. The default value is 0.01 mg/m$^{-3}$. To switch on Cox and Munk BRDF, specify any of the `cox_and_munk` options and define at least `cox_and_munk_u10`.

**cox_and_munk_pcl_map**

A possibility to specify pigment concentration (in mg/m3) for the Cox and Munk ocean BRDF with the help of an *netCDF* file, which is used in combinition with options `latitude`, `longitude`, and `time`.

```
cox_and_munk_pcl_map file [variable_name]
```

where `file` is the location of the *netCDF* file. *libRadtran* reads the value at the nearest pixel to the given `latitude` and `longitude`. No spatial interpolation or averaging of the values is done.

The default name of the pigment concentration variable is `chlorophyll`, but can be changed with the optional argument `variable_name`. The pigment concentration must be provided as function of `latitude` and `longitude`, `chlorophyll(lat, lon)`, or additionally may also depend on time `chlorophyll(time, lat, lon)`. If a time-dependent pigment concentration is specified, the pigment concentration will be interpolated according to the

option time_interpolate. All grids must be provided in the file as double lat(lat), double lon(lon), and double time(time).

**cox_and_munk_sal**

Salinity for Cox and Munk (1954a,b) ocean BRDF (in "per mille", 0.1%; this unit is equivalent to the other common units for salinity, ppt - parts per thousand, psu - practical salinity unit).

```
cox_and_munk_sal value
```

At present only available with rte_solver cdisort, and rte_solver fdisort2. The number of streams (nstr) is automatically increased to 16 if cox_and_munk BRDF is switched on, to avoid numerical problems. The default value is 34.3. To switch on Cox and Munk BRDF, specify any of the cox_and_munk options and define at least cox_and_munk_u10.

**cox_and_munk_sal_map**

Specify ocean salinity (in ppt) for the Cox and Munk (1954a,b) ocean BRDF with the help of an *netCDF* file, which is used in combinition with the options latitude, longitude, and time.

```
cox_and_munk_pcl_map file [variable_name]
```

where file is the location of the *netCDF* file. *libRadtran* reads the value at the nearest pixel to the given latitude and longitude. No spatial interpolation or averaging of the values is done.

The expected name of the pigment concentration variable is per default salinity, but can be changed with the optional argument variable_name. The pigment concentration must be provided as function of latitude and longitude, salinity(lat, lon), or additionally may also depent on time salinity(time, lat, lon). If a time-dependent salinity is specified, the salinity will be interpolated according to the option time_interpolate. All grids must be provided as double lat(lat), double lon(lon), and double time(time).

**cox_and_munk_solar_wind**

Use old definition of wind direction for Monte Carlo simulations. If this switch is set, the wind azimuth is identical to the incoming photon azimuth. Else, the wind azimuth is set by cox_and_munk_uphi or is 0 by default.

**cox_and_munk_u10**

Wind speed for Cox and Munk (1954a,b) ocean BRDF (in m/s).

```
cox_and_munk_u10 value
```

At present only available with rte_solver cdisort, and rte_solver fdisort2. The wind speed is the most important parameter affecting ocean BRDF. The minimum allowed wind speed is 1 m/s because otherwise the strong specular

reflection causes numerical problems. If a lower value is specified, the wind speed is automatically set to 1m/s. Also, the number of streams (`nstr`) is automatically increased to 16 if `cox_and_munk` BRDF is switched on, to avoid numerical problems. To switch on Cox and Munk BRDF, specify any of the `cox_and_munk` options and define at least `cox_and_munk_u10`.

**cox_and_munk_u10_map**

Specify wind speed (in m/s) for the Cox and Munk (1954a,b) ocean BRDF with the help of an *netCDF* file, which is used in combinition with the options `latitude`, `longitude`, and `time`.

```
cox_and_munk_u10_map file
```

where `file` is the location of the *netCDF* file. *libRadtran* reads the value at the nearest pixel to the given `latitude` and `longitude`. No spatial interpolation or averaging of the values is done.

The file must contain the elements of the wind vector `U10` and `V10`. These must be specified as functions of `latitude` and `longitude` `U10(lat, lon)`, `V10(lat, lon)`, or additionally may also depent on time `U10(time, lat, lon)`,`V10(time, lat, lon)`. If the variable time is present in the file, the wind speed will be interpolated according to the option `time_interpolate`. All grids must be provided as `double lat(lat)`, `double lon(lon)`, and `double time(time)`.

**cox_and_munk_uphi**

Wind direction for Cox and Munk (1954a,b) ocean BRDF.

```
cox_and_munk_uphi value
```

Default value is 0 degrees, which is wind from the South. 90 degrees corresponds to wind from the West, etc. (Honestly, this was never truly validated. It could possibly be that 0 is wind from the North, 90 is wind from the East, etc.)

**crs_file**

May be used to specify cross sections of O3, NO2, BRO, OCLO, or HCHO to be used instead of those supplied with *libRadtran*. No temperature dependence may be specified. Use as follows:

```
crs_file NO2 ../examples/no2_crs.dat
```

The `NO2` or `O3`,`BRO` or `OCLO` or `HCHO` must be specified to identify the species for which the cross section applies. The cross section file has two columns:

**1** wavelength (nm)

**2** cross section ($cm^2$)

**ctwostr_pseudospherical**

Pseudo-spherical geometry in ctwostr. Default is plane-parallel.

**data_files_path**

The path to the directory where all `uvspec` internal data files live, e.g. the files that are in the subdirectories of the `data` directory that comes with the `uvspec` distribution.

```
data_files_path path
```

The default for `path` is `../data/`.

**day_of_year**

Integer, to correct the calculated radiation quantities for the Sun-Earth distance for the specified Julian day (1-365).

```
day_of_year value
```

If not specified, the Earth-Sun distance is 1 AU (i.e. equinox distance), that is, no correction is applied to the extraterrestrial irradiance `solar_file`. Alternatively `time` may be used for that purpose.

**deltam**

Turn delta-M scaling on/off. Set to either `on` or `off`. Note that for the `rte_solver cdisort` and `rte_solver fdisort2` delta-M scaling is hardcoded to be always on.

**dens_column**

Set the total column of a density profile. The column is integrated between the user-defined `altitude` and TOA (top of atmosphere). The syntax is

```
dens_column species column [unit]
```

where `species` is one of O3, O2, H2O, CO2, NO2, BRO, OCLO, or HCHO, see also `dens_file`. The second argument is the total column value, and the optional third argument is the unit, in which the column is given. The unit can be DU (Dobson units) or CM_2 (molecules/cm$^2$). The default units are DU for O3, and CM_2 for all other gases. It is possible to have several `dens_column` commands in the input file (maximum one per species). The following sets the $NO_2$ total column to 1.2 DU.

```
dens_column NO2 1.2 DU
```

**dens_file**

Specify density profiles (or matrix, see below) of various traces gases to be included in the radiative transfer calculation. The entry of the input file looks like:

```
dens_file gas_species [unit] filename
```

At the moment following gas_species are included: ozone (O3), nitrogen dioxide (NO2), water vapor (H2O), bromine oxide (BRO), chlorine dioxide (OCLO), formaldehyde (HCHO), and carbon dioxide (CO2). The gas species is identified by

their abbrevations given in the parenthesis above. `unit` is an optional argument to define the unit of the density. The profiles can be given in particles per cm3 ($cm^{-3}$), in particles per m3 ($m^{-3}$), as volume mixing ratio (vmr), as mass mixing ratio (mmr), or as relative humidity (only for water). The default for `unit` is $cm^{-3}$. The model expects a density file with two columns:

**1** Altitude above sea level in km.

**2** The density of trace gas [in the specified unit]

The altitude grid may be different from that in `atmosphere_file`. All densities inside the range of the `dens_file` are replaced. For all other altitudes the values from the `atmosphere_file` are used. If the density is specified as -1 at a level, the value from `atmosphere_file` is used.

To scale the profile to a total column value use `dens_column`.

For airmass factor calculations it is for some species necessary to account for the variation of the profile with sza. This may be accomplished by specifying a `dens_file` in the following format:

```
 0.0 SZA1 SZA2 ...
z(1) dens(1,1) ...
z(2) .  .
.  .  .
```

where z(i) are the altitude levels above sea level in km, SZA is the solar zenith angle in degrees, and dens is the density [in the specified unit] of the trace gases as function of solar zenith angle and altitude. The matrix may only be specified for one specie. It may however be combined with profiles of other species. A density matrix can only be used in connection with `rte_solver sdisort`!

**disort_icm**

Intensity correction method for `rte_solver cdisort` or `rte_solver fdisort2`. Valid options are `phase`, i.e. the phase function is used for the Nakajima intensity correction, and `moments`, i.e. the Legendre moments are used for the correction. Optionally, the option `off` turns off the intensity correction. Default is `phase`.

**earth_radius**

Specify the earth radius in km.

```
earth_radius value
```

This is needed by all solvers in spherical geometry . The default value is 6370 km.

**ECMWF_atmosphere_file**

Reads in combination with the options `latitude`, `longitude`, and `time` (all mandatory) the pressure, temperature, ozone, and water vapour from an ECMWF *netCDF* data file and will combine it with the data given by the `atmosphere_file`.

```
ECMWF_atmosphere_file file
```

No spatial interpolation of the values is done. The atmospheric data nearest to the specified `time` will be selected (or linearly interpolated if `time_interpolate` is switched on). Atmospheric profiles, which are not provided by the ECMWF file (O2, CO2, NO2) are taken from the `atmosphere_file`. Per default, also the atmosphere above the ECMWF data is taken from the `atmosphere_file`. In order to avoid this, please have a look at the option: `ECMWF_levels_only`.

**ECMWF_levels_only**

The atmosphere considered in the simulation has the same height range as the data in the `ECMWF_atmosphere_file`/`radiosonde`-file. No further levels are added above those. This option has only an effect in combination with `ECMWF_atmosphere_file` or `radiosonde` (this option is identical to `radiosonde_levels_only`).

**ECMWF_ic_file**

Reads in combination with the options `latitude`, `longitude`, and `time` (all mandatory) the pressure, temperature, and cloud ice water content (CIWC) and cloud cover (CC) from an ECMWF *netCDF* data file.

```
ECMWF_ic_file file
```

No spatial interpolation of the values is done. The data nearest to the specified `time` will be selected (or linearly interpolated if `time_interpolate` is switched on). In order to use the ECMWF data without cloud overlap assumption, use `cloud_overlap off`.

**ECMWF_ic_reff**

*This option is preliminary and still subject to change!* The ECMWF data only contains cloud water content, but no effective radius. With this option, the effective radius can be specified. There are two possibilities: For a fixed effective radius use the keyword `fixed` and specify the `reff` in micrometer.

```
ECMWF_ic_reff fixed reff
```

In order to use the parametrisation by cheng Ou and Liou (1995) use the keyword `Ou`.

```
ECMWF_ic_reff Ou
```

The default option is Ou.

**ECMWF_ozone_climatology**

The Integrated Forecast System (IFS) of the ECMWF uses a ozone climatology for radiative transfer instead of the ozone simulated by the IFS. If this option is activated the ozone profile of the `atmosphere_file` or `ECMWF_atmosphere_file` is replaced by the ozone climatology by Fortuin and Langematz (1995). (If there is also a `dens_file` for ozone, it modifies the ozone climatology profile.)

**ECMWF_wc_file**

Reads in combination with the options `latitude`, `longitude`, and `time` (all mandatory) the pressure, temperature, and cloud liqid water content (CLWC) and cloud cover (CC) from an ECMWF *netCDF* data file.

```
ECMWF_wc_file file
```

No spatial interpolation of the values is done. The data nearest to the specified `time` will be selected (or linearly interpolated if `time_interpolate` is switched on). In order to use the ECMWF data without cloud overlap assumption, use `cloud_overlap off`.

**ECMWF_wind_file**

Reads in combination with the options `latitude`, `longitude`, and `time` (all mandatory) the wind components U, V, and W from an ECMWF *netCDF* data file.

```
ECMWF_wind_file file
```

The data nearest to the specified `time` will be selected (or linearly interpolated, if `time_interpolate` is switched on).

**emissivity_map**

*This option is preliminary and still subject to change (no wavelength dependency yet)!* Specify a wavelength independent emissivity with the help of an *netCDF* file, which is used in combinition with the options `latitude`, `longitude`, and `time`.

```
emissivity_map file [variable_name]
```

where `file` is the location of the *netCDF* file. With the optional argument `variable_name` the name of the emissivity variable in the *netCDF* file can be specified. (By default the expected name is `EMIS`.) The emissivity must be specified as function of `latitude` and `longitude` `EMIS(lat, lon)`, or additionally may also depent on time `EMIS(time, lat, lon)`. All grids must be provided as `double lat(lat)`, `double lon(lon)`, and `double time(time)`. *libRadtran* reads the value at the nearest pixel to the given `latitude` and `longitude`. No spatial interpolation or averaging of the values is done. If the variable time is present in the file, the emissivity data nearest to the specified time will be selected (or interpolated if `time_interpolate` is switched on).

**f11_mixing_ratio**

The mixing ratio of F11 in ppm (default: 0.000268 ppm).

```
f11_mixing_ratio value
```

**f12_mixing_ratio**

The mixing ratio of F12 in ppm (default: 0.000503 ppm).

```
f12_mixing_ratio value
```

**f22_mixing_ratio**
The mixing ratio of F22 in ppm (default: 0.000105 ppm).

```
f22_mixing_ratio value
```

**filter_function_file**
If specified, the calculated spectrum is multiplied with a filter function defined in file.

```
filter_function_file file
```

The file must contain two columns. Column 1 is the wavelength, in nm. Column 2 is the corresponding filter function value. Comments start with #. Empty lines are ignored. In combination with `output sum` or `output integrate` this option is useful e.g. to calculate weighted irradiances or actinic fluxes or to simulate broadband or satellite observations.

**fisot**
Specifies that isotropic illumination is used at top-boundary instead of beam source. Useful for those who want to calculate the reflectance for a homogeneous or inhomogeneous atmosphere. The intensity is still set by `solar_file`.

**flexstor**
Provide output in flexstor format. Must not be combined with `header`. Also, does not currently work when `umu` and/or `phi` is specified.

**h2o_mixing_ratio**
The mixing ratio of H2O in ppm. Scale the profile so that the mixing ratio at the user-define `altitude` assumes the specified value.

```
h2o_mixing_ratio value
```

**h2o_precip**
Precipitable water in kg / m2 (which is approximately 1mm). The water vapor profile is scaled accordingly. The precipitable water is integrated from the user-defined `altitude` to TOA (top of atmosphere).

```
h2o_precip value
```

**header**

Include information on some of the input parameters in the output. May not be combined with `flexstor`. Please note that the information provided is rather incomplete because this option was introduced quite early and was never updated. For a more complete information please use the `verbose` option.

**heating_rate**

Calculation of heating rates. Output is only provided at altitudes specified by `zout`. To get heating rate profiles a number of altitudes must thus be specified by `zout`. Heating rates is the change of temperature with time in units of K/day. For spectral calculations the default output is a matrix:

```
0.0         zout1          zout2 ...
lambda1     heating_rates  ...
lambda2       .
   .          .
   .          .
```

For integrated calculations (`output sum` or `output integrate`) the default output is in two columns with column 1 being the altitude and column 2 the heating rates. The output of `heating_rate` can also be specified with the `output_user` option. Note that heating rates are only well-behaved up to altitudes for which the respective correlated-k options are valid. E.g. about 60 km for `fu` and about 80 km for `kato`, `kato2`, `kato2.96`, and `lowtran`. Attention: For spectral calculations, the extraterrestrial spectrum is assumed to be in mW/(m2 nm).

Two different methods are implemented to calculate the heating rate, which can be selected with an optional keyword:

```
heating_rate [method]
```

where `method` may be either `layer_cd` (heating rates are derived from centered differences of the flux, this is the default method) or `local` (heating rates are derived from the actinic flux). Attention: `heating_rate local` introduces new levels into the profile which slightly affects the model output. There is also a third `method` called `layer_fd`, which means that heating rates are derived from forward differences of the flux over one layer. Please be aware: If using `layer_fd`, the output is not representativ for a *level*, but for the *layer* from the z-level of the line in the output file, where it is written, up to next *output level* above!

**ic_cloudcover**

Set the fraction of the horizontal sky area which is covered by clouds.

```
ic_cloudcover value
```

When a cloud cover is specified, the result will be calculated by the independent pixel approximation (IPA), that is, as weighted average of cloudless sky and overcast sky, where the cloud properties are taken from `ic_file`, etc. Please note that, if both `wc_cloudcover` and `ic_cloudcover` are set, both must be equal.

This option is ignored, if the option `cloud_fraction_file` is used.

**ic_file**

Location of file defining ice cloud properties.

```
ic_file file
```

The file must contain three columns. Column 1 is the altitude in km, column 2 the ice water content in grams per cubic meter, and column 3 the effective particle radius in micrometer. Note that the definition of cloud altitudes in `ic_file` refers to sea level, not to altitude above ground. E.g., when `altitude` is set to 1.63km, and the first cloud level is defined at 3km, the cloud would start at 1.37km above ground. Comments start with `#`. Empty lines are ignored. An example of an ice cloud is given in `examples/IC.DAT`.

Per default the cloud properties are interpreted as layer properties. Before version 1.4 the default was level properties: The optical depth of a layer was calculated using information from the upper and lower levels defining the layer, see `ic_layer` and `ic_level`. To switch to the old behaviour, use `ic_level`. See section 3.3.5 about ice clouds for a realistic example how the contents of the `ic_file` are converted to optical properties.

**ic_files**

A way to specify ice cloud optical depth, single scattering albedo, and phase function moments for each layer.

```
ic_files file
```

The file specified by `ic_files` has two columns where column 1 is the altitude in km. The second column is the name of a file which defines the optical properties of the level starting at the given altitude. The files specified in the second column must have the following format:

**Column 1:**
The wavelength in nm. These wavelengths may be different from those in `solar_file`. Optical properties are interpolated to the requested wavelengths.

**Column 2:**
The extinction coefficient of the layer in units km-1.

**Column 3:**
The single scattering albedo of the layer.

**Column 4-(nmom+4):**
The moments of the scattering phase function.

Note that for `rte_solver cdisort` and `rte_solver fdisort2` it makes good sense to make the number of moments larger than `nstr` because all moments are used in the calculation. For `rte_solver fdisort1` and `rte_solver polradtran` the number of moments included in the calculations will be `nstr`+1. Higher order moments will be ignored for these solvers. Please note that the uppermost line of the `ic_files` denotes simply the top altitude of the uppermost layer.

The optical properties of this line are consequently ignored. There are two options for this line: either an optical property file with zero optical thickness is specified or "NULL" instead.

**ic_fu_tau**

Specify if the Fu (1996) optical properties are delta-scaled or not. With

```
ic_fu_tau scaled
```

delta-scaling is switched on, with

```
ic_fu_tau unscaled
```

it is switched off. The default is without delta-scaling. Please note that this was changed on July 22, 2008: Before, delta-scaling was switched on by default which might have caused some confusion, because irradiance calculations were not consistent with the other ice cloud parameterizations implemented in `uvspec`. Using the Fu (1996) parameterization in combination with one of `ic_set_tau/tau550/gg/ssa` or `ic_scale_gg/ssa` you now get consistent results with all other ice cloud parameterizations.

**ic_fu_reff**

If

```
ic_fu_reff fu
```

is specified, the parameterization uses the original definition of the effective radius as specified in Fu (1996); Fu et al. (1998). By default it uses the same definition of the effective radius as the Key et al. (2002), Yang et al. (2000) and Baum et al. (2005a,b) parameterizations; see discussion of `ic_properties`.

**ic_habit**

Ice crystal habit for the Yang et al. (2000), Key et al. (2002) and `hey` parameterizations, see also `ic_properties key/yang/hey`.

```
ic_habit type
```

For Key/Yang `type` may be one of `solid-column`, `hollow-column`, `rough-aggregate`, `rosette-4`, `rosette-6`, `plate`, `droxtal`, and `spheroid`. Please note that this parameterization is only valid for a restricted size range, depending on the habit (see table 1 in Key et al. (2002). Also, some of the habits are only available for wavelengths below 5 micrometer (`rosette-4`) while others are only available for wavelengths larger than 3 micrometer (`droxtal`, `spheroid`). For `hey` the following habits can be chosen: `solid-column`, `hollow-column`, `rough-aggregate`, `rosette-6`, `plate`, and `droxtal`; here all habits are available for effective radii from 5 to 90 micrometers in the wavelength region from 0.2 to 5 micrometers.

**ic_ipa_files**

A two-column file, defining ice cloud property files (see ic_file) in the first column and the correspoding weights in the second column.

```
ic_ipa_files file
```

The radiative transfer calculation is performed independently for each cloud column and the result is the weighted average of all independent columns. If ic_ipa_files and wc_ipa_files are both defined, both must have the same columns in the same order, otherwise uvspec will complain. See examples/UVSPEC_WC_IC_IPA_FILES.INP for an example.

**ic_layer**

Interpret ice cloud properties as layer properties (this is the default behaviour since version 1.4; see also ic_file). Cloud properties are assumed to be constant within each layer. The layer reaches from the level where the properties are defined in the ic_file to the level above. For example, the following lines

```
#       z      IWC     R\_eff
#     (km)   (g/m^3)  (um)
       4.000    0.0    0.0
       3.000    1.0   10.0
```

define a cloud in the layer between 3 and 4 km with sharp boundaries.

**ic_level**

Interpret cloud properties as level properties (this was the default behaviour before version 1.4; see also ic_file). If ic_level is defined, a ic_file would be interpreted as follows:

```
#       z      IWC     R\_eff
#     (km)   (g/m^3)  (um)
      5.000    0       0
      4.000    0.2     12.0
      3.000    0.1     10.0
      2.000    0.1      8.0
```

The value 0.2 g/m3 refers to altitude 4.0km, as e.g. in a radiosonde profile. The properties of each layer are calculated as average over the adjacent levels. E.g. the single scattering properties for the model layer between 3 and 4km are obtained by averaging over the two levels 3km and 4km. To allow easy definition of sharp cloud boundaries, clouds are only formed if both liquid water contents above and below the respective layer are larger than 0. Hence, in the above example, the layers between 2 and 3 as well as between 3 and 4km are cloudy while those between 1 and 2km and between 4 and 5km are not.

**ic_no_scattering**

Switch off scattering by ice clouds.

**ic_properties**

Defines how ice water content and effective particle radius are translated to optical

properties.

```
ic_properties type
```

Possible choices for `type` are

**fu** Parameterization by Fu (1996); Fu et al. (1998), see `ic_file`; this is the default setting. Note that this is a parameterization which has been created to calculate fluxes but not radiances. Note also that the optical properties in the solar range provided by Fu (1996) are delta-scaled properties (that is, the forward peak of the phase function is truncated and optical thickness, asymmetry parameter, and single scattering albedo are reduced accordingly), whereas `uvspec` uses non delta-scaled properties unless the option `ic_fu_tau scaled` is specified. By default the parameterization by Fu (1996) is treated consistently with all other ice cloud parameterizations. For wavelengths up to 4 micrometer Fu (1996) is used while for wavelengths larger than 4 micrometer Fu et al. (1998) is chosen. Please note that Fu (1996) is based on ray-tracing calculations while Fu et al. (1998) is a mixture of ray-tracing and Mie calculations (which is required for the infrared wavelengths where the geometrical assumption does not hold). Hence, both parameterizations are not fully consistent. Rather, differences of some % are to be expected in the wavelength region where both parameterizations overlap. Also, the wavelength dependence in the solar and infrared parts is treated differently: In the solar part (Fu, 1996) the optical properties are defined for wavelength bands - hence they are assumed constant within each band. In the infrared (Fu et al., 1998) they are defined at certain wavelengths and linearely interpolated in between. If you use this option, please see also the discussion of `ic_fu_tau` and `ic_fu_reff`. The allowed range for the effective radius is from 9.315 - 65.120 micrometer.

**echam4** Use the simple two-band parameterization of the ECHAM4 climate model, described in Roeckner et al. (1996); this is probably only meaningful if you want to compare your results with ECHAM4, the two bands are 0.2 - 0.68 micrometer and 0.68 - 4.0 micrometer. Within the two ECHAM4 bands, the optical properties are assumed constant.

**key** Parameterization by Key et al. (2002). This parameterization can also be used to calculate radiances because it uses a double-Henyey-Greenstein phase function which better represents both forward and backward peaks. This parameterization covers the wavelength region from 0.2 to 5.0 micrometer and is available for the following habits: solid-column, hollow-column, aggregate, rosette-4, rosette-6, and plate.

**yang** Parameterization similar to Key et al. (2002) but based on more recent single scattering calculations. Below 3.4 micrometer it actually equals the Key et al. (2002) parameterization while from 3.4 - 100 micrometer new coefficients have been calculated with much higher wavelength resolution and better accuracy. Hence, `yang` should give a reasonably consistent approximation from 0.2 - 100 micrometer, suitable for spectrally resolved calculations of radiance and irradiance. The covered range for the effective radius depends on the `ic_habit`.

(In micrometer: `solid-column` [5.96, 84.22], `hollow-column` [4.97, 70.24], `rough-aggregate` [3.55, 108.10], `rosettes-4` [2.77, 45.30], `rosettes-6` [2.85, 46.01], `plate` [4.87, 48.18], `dendrites` [0.45, 1.88], `droxtal` [9.48, 293.32], `spheroid` [6.58, 203.39]).

**baum** Use ice cloud parameterization from Baum et al. (2005a,b), http://www.ssec.wisc.edu/~baum/Cirrus/IceCloudModels.html. In combination with the radiative transfer solvers `cdisort`, and `fdisort2`, accurate phase functions are used.

**baum_hufit** Similar to the option `baum` but here the phase function is parameterized by 128 Legendre coefficients, calculated with the delta-fit method from Hu and Stamnes (2000). This parameterization covers the region from 0.4 to 2.2 micrometer. If high accuracy is needed e.g. in the vicinity of the halo, the forward peak, or the backscatter peak, `ic_properties baum` is recommended.

**hey** Use pre-calculated ice cloud optical properties including full phase matrices. This option has newly been implemented and is not yet well validated. Please check your results carefully!! The parameterization is currently only available for the spectral region from 0.2 to 5 micrometers. The single scattering properties have been been generated by Hong Gang using the models by Yang et al. (2000). The parameterization is based on simple gamma distributions

$$n(r) = n_0 r^{\alpha} \exp\left(-\frac{(\alpha+3)r}{r_e}\right), \tag{6.2}$$

where $n_0$ is found by normalization and $\alpha$ is set to 1. In case of spherical particles the parameter $r_e$ would be the effective radius. For aspherical particles, the parameter $r_e$ is found iteratively so that the size distribution yields the required effective radius. The parameterization is availabe for the following habits: `solid-column`, `hollow-column`, `rough-aggregate`, `rosette-6`, `plate`, and `droxtal`. The default habit is `solid-column`. The habit can be specified using the option `ic_habit`.

**mie** Use pre-calculated Mie tables; useful for `correlated_k`; the tables are expected in `data_files_path/correlated_k/.../`. For spectral or pseudo-spectral calculations `ic_properties_interpolate` has to be defined explicitely to initiate the interpolation of the optical properties to the internal wavelength grid. Note that a Mie calculation assumes spherical ice particles, the scattering function of which differs systematically from non-spherical particles. Hence, `ic_properties mie` is usually not representative of natural ice clouds.

**filename** Read optical properties from specified filename; file format is as produced by the `mie` tool of *libRadtran* (see `output_user netcdf`).

The default property is `fu`.

Please note also that, in contrast to spherical particles, there is no unique definition of effective size for non-spherical particles. In particular, the above parameterizations use different definitions which, however, differ only by a constant factor. Yang et al.

(2000), citeKey2002, and Baum et al. (2005a,b) use the general definition

$$r_{\text{eff}} = \frac{3}{4}\frac{\int V(h)n(h)dh}{\int A(h)n(h)dh} \tag{6.3}$$

where $h$ is the maximum dimension of an ice crystal, $n(h)$ is the number of particles with maximum dimension $h$ in the size distribution, and $V$ and $A$ are the volume and mean projected area of the particles, respectively. The volume and area are based on the spherical diameter with equivalent volume and the spherical diameter with equivalent projected area as defined by Yang et al. (2000). On the other hand, Fu (1996); Fu et al. (1998) use hexagonal columns and use the following definition

$$r_{\text{eff}} = \frac{\int D^2 L n(L)dL}{2\int (DL + \frac{\sqrt{3}}{4}D^2)n(L)dL} \tag{6.4}$$

where $D$ is the width of the ice crystal (that is, the maximum diameter of the hexagonal area) and $L$ is the length. The integrand in the numerator is proportional to the volume while that in the denominator is proportional to the projected area. Evaluating these formulas one finds that, for the same hexagonal particle, the effective radius would be $3\sqrt{3}/4 = 1.299$ times larger following the Yang et al. (2000), Key et al. (2002) definition rather than the Fu (1996); Fu et al. (1998) definition. As an example, an effective radius of $20\mu m$ with `ic_properties fu` and `ic_fu_reff fu` and $1.299 \cdot 20\mu m = 26\mu m$ with `ic_properties yang` would give comparable results for hexagonal columns. To use the original definition of the effective radius by Fu (1996); Fu et al. (1998) use `ic_fu_reff fu`!

**ic_properties_interpolate**

Interpolate ice cloud optical properties over wavelength; useful for precalculated optical property files defined with `ic_properties`. Please note that this option may be extremely memory-consuming because for each internal wavelength a full set of Legendre moments of the phase function is stored (up to several thousands).

**ic_saturate**

With this option, the relative humidity inside ice clouds can easily be adjusted to a user-defined value, e.g. saturated with respect to water. This option has one mandatory and one optional argument:

```
ic_saturate switch [relative_humidity]
```

where `switch` is `on`, `off`, or `ipa`. The second optional argument determines the relative humidity in (with respect to water!) inside the cloud. The default value is 100. `ipa` is only relevant for independent column calculations.

The follwing details apply for independent column simulations: Using `switch on`, the air in *all* columns will be saturated, if there is a cloud in at least one of the columns (this option should be used for stratiform clouds). Using `switch ipa`, only cloudy columns are affected (this option should be used for convective cloud fields.)

**ic_scale_gg**

Scale the ice cloud asymmetry factor for all wavelengths and altitudes with a float

between 0.0 and 1.0.

```
ic_scale_gg value
```

If you use this option in combination with the ice cloud properties by Fu (1996), please make sure that you understand the explanation of `ic_fu_tau`.

**ic_scale_ssa**
Scale the ice cloud single scattering albedo for all wavelengths and altitudes with a float between 0.0 and 1.0.

```
ic_scale_ssa value
```

If you use this option in combination with the ice cloud properties by Fu (1996), please make sure that you understand the explanation of `ic_fu_tau`.

**ic_set_gg**
Set the ice cloud asymmetry factor for all wavelengths and altitudes to a float between -1.0 and 1.0. Please note that this option is only applied if a Henyey-Greenstein phase function is used but not if an explicit phase function is defined e.g. with a ic_files. It doesn't make sense to modify only the first moment of an explicit phase function.

```
ic_set_gg value
```

If you use this option in combination with the ice cloud properties by Fu (1996), please make sure that you understand the explanation of `ic_fu_tau`.

**ic_set_ssa**
Set the ice cloud single scattering albedo for all wavelengths and altitudes to a value between 0.0 and 1.0.

```
ic_set_ssa value
```

If you use this option in combination with the ice cloud properties by Fu (1996), please make sure that you understand the explanation of `ic_fu_tau`.

**ic_set_tau**
Set the total ice cloud optical depth to a constant value for all wavelengths.

```
ic_set_tau value
```

The optical thickness defined here is the integral from the surface at the user-defined `altitude` to TOA (top of atmosphere). This option is useful only for monochromatic calculations or in wavelength regions where the optical properties of ice clouds can be considered constant, e.g. the ultraviolet region. If you use this option in combination with the ice cloud properties by Fu (1996), please make sure that you understand the explanation of `ic_fu_tau`.

**ic_set_tau550**

Set the ice cloud optical thickness at 550nm. Other wavelengths are scaled accordingly.

```
ic_set_tau550 value
```

The optical thickness defined here is the integral from the surface at the user-defined `altitude` to TOA (top of atmosphere). Note that this option requires for technical reasons that the wavelength interval defined by `wavelength` does contain 550nm. If you use this option in combination with the ice cloud properties by Fu (1996), please make sure that you understand the explanation of `ic_fu_tau`.

**include**

Include a file into the `uvspec` input.

```
include file
```

Works exactly like the C `#include` or the Fortran `INCLUDE` statements.

**latitude**

This option can be used to specify the latitude of the location to simulate. (This option only has an effects, if `longitude` is specified, too.)

```
latitude deg [min] [sec]
```

where `deg min sec` is the position in degrees, arc minutes, and arc seconds north. `deg` might also be a float number. `min` and `sec` may be obmitted. The `latitude` information will be used for the following:

`latitude` in combination with `longitude`, `time`, and any `map`-option is used to select the location where to read the input data.

`latitude` in combination with `longitude` and `time` is used to calculate the solar zenith angle, if no `sza` is specified (see also `time_interval`).

`latitude` in combination with `longitude` and `time` is used to choose a suitable default atmosphere file, if no `atmosphere_file` is specified.

**longitude**

This option can be used to specify the longitude of the location to simulate. (This option only has an effects, if `latitude` is specified, too.)

```
longitude deg [min] [sec]
```

where `deg min sec` is the position in degrees, arc minutes, and arc seconds east. `deg` might also be a float number. `min` and `sec` may be obmitted. The `longitude` information will be used for the following:

`longitude` in combination with `latitude`, `time`, and any `map`-option is used to select the location where to read the input data.

`longitude` in combination with `latitude` and `time` is used to calculate the solar zenith angle, if no `sza` is specified (see also `time_interval`).

`longitude` in combination with `latitude` and `time` is used to choose a suitable default atmosphere file, if no `atmosphere_file` is specified.

**molecular_tau_file**

Location of molecular absorption optical depth file.

```
molecular_file file
```

Usually, molecular absorption is calculated from trace gas concentrations provided in `atmosphere_file` (scaled with `ozone_column`, etc. Use this option only if you want to specify the optical depth directly (e.g. for a model intercomparison) or for a line-by-line calculation. If a spectral `molecular_tau_file` is specified, the wavelength grid defined there is used as the internal wavelength grid for the radiative transfer calculation, if not defined otherwise with `transmittance_wl_file`. `molecular_tau_file` can be either of the following three formats:

**Monochromatic**

Column 1 is the altitude in km Column 2 is the absorption optical depth of each layer.

**Spectral, ASCII**

The first line contains the level altitudes in decreasing order; the following lines contain the wavelength [nm] in the first column and then the absorption optical depths of each layer.

**Spectral, netcdf**

An example is available at the libRadtran homepage, the file `UVSPEC.O2A.afglms.cdf` is a line-by-line spectrum of the oxygen A-Band around 760nm, calculated for the mid-latitude summer atmosphere. The advantage of netcdf compared to ASCII is that it is much faster to read, and that the file is a self-contained, including data and a description of the variables and arrays. It is therefore particularly useful for line-by-line calculations where usually many spectral data points are involved.

Comments start with `#`. Empty lines are ignored.

**n2o_mixing_ratio**

The mixing ratio of N2O in ppm (default: 0.28 ppm).

```
n2o_mixing_ratio value
```

**no_absorption**

Switch all (molecular, aerosol, cloud, and ice cloud) absorption off. Please note that this option simply sets the absorption optical thickness to 0. If used together with `xxx_set_tau` this might be a bit confusing but probably the most logical way. E.g. when using `aerosol_default` and `aerosol_set_tau 1`, the aerosol optical thickness is set to 1, with 0.940539 scattering and 0.059461 absorption. If `no_absorption` is added, the absorption optical thickness is set to 0 while the

scattering optical thickness is preserved at 0.940539 (even though 1 was specified by the user). We find this the most logical solution of the problem because by switching `no_absorption` off and on one tests the effect of the absorber in an isolated way, rather than mixing absorption and scattering. The same is true for water and ice clouds. Note, that thermal emission of molecules is also switched off.

**no_molecular_absorption**
Switch off molecular absorption.

**no_rayleigh**
Switch Rayleigh scattering off.

**no_scattering**
Switch scattering off.

**no2_column_du**
Obsolete, use `dens_column` instead.

**no2_column_moleccm-2**
Obsolete, use `dens_column` instead.

**nscat**
The order of scattering for the `sos` radiative transfer equation solver.

```
nscat value
```

Default is 20. May also be used with the `sdisort` solver. If set to 1 `sdisort` will run in single scattering mode while if set in to 2, `sdisort` runs in full multiple scattering mode.

**nrefrac**
For the `rte_solver sdisort` refraction may be included by

```
nrefrac value
```

where `value` has the meaning

**0** No refraction, default.

**1** Refraction included using fast, but harsh method.

**2** Refraction included using slow, but accurate method.

If refraction is included also set parameter `refraction_file`.

**nstr**
Number of streams used to solve the radiative transfer equation.

```
nstr value
```

Default is 6 for fluxes and 16 for radiances. (For `rte_solver fdisort1`, `fdisort2` and `cdisort` only even nstr are possible.)

**o2_mixing_ratio**
The mixing ratio of O2 in ppm.

```
o2_mixing_ratio value
```

Scale the profile so that the mixing ratio at the user-defined `altitude` assumes the specified value.

**o3_crs**
Choose between the various ozone cross sections by

```
o3_crs type
```

where `type` is one of

**Bass_and_Paur**
Bass and Paur (1985) ozone cross section.

**Molina**
Molina and Molina (1986) ozone cross section.

**Daumont**
Ozone cross section by M. et al. (1992), Malicet et al. (1995).

Molina and Molina (1986) is default.

**optimize_fortran**
When this option is activated, the minimum dimensions of the FORTRAN arrays for the specific input conditions are determined and written to `stderr`. Using `worldloop` two tolerance levels are added to the minimum number of atmospheric levels, as this number may change during `worldloop` calculations.

**output**
Decide how the output from `uvspec` is processed:

```
output type
```

where type is one of

**sum**
Sum output over wavelength. Useful in combination with the `correlated_k` option (`kato`, `kato2`, `kato2.96`, `Fu`, `avhrr_kratz`).

**integrate**
Integrate output over wavelength for solar and over wavenumber for thermal simulations. Useful for spectral calculations and `correlated_k lowtran`.

**per_nm**
Output is given in W/(m$^2$ nm) or mW/(m$^2$ nm) (W or mW is determined by the extraterrestrial spectrum.)

**per_cm$^{-1}$**
Output is given in W/(m$^2$ cm$^{-1}$ or mW/(m$^2$ cm$^{-1}$).

**per_ck_band**
Output is given in W/m$^2$ or mW/m$^2$ per correlated-k band. (This option can not be used for spectral calculations and `correlated_k` LOWTRAN in the solar range.)

**none**
No processing - output spectral information (default).

**output_format**
Specification of the output format.

```
output_format format
```

where `format` is either `ascii` (default) or `flexstor`. There is also the possibility to write `uvspec` simulation results to an existing *netCDF* file

```
output_format format output_file
```

where `format` is `netCDF` if the `output_file` contains a lat/lon/time grid or `sat_picture` if the `output_file` contains a pixel_x/pixel_y/time grid.

**output_user**
User defined output. Here the user may specify the columns desired for output.

```
output_user format
```

where `format` is one or more of the following.

**lambda**
Wavelength in nm.

**wavenumber**
Wave number in cm$^{-1}$.

**sza**
solar zenith angle

**zout**
Output altitude in km.

**edir, eglo, edn, eup, enet, esum**
The direct, global, diffuse downward, and diffuse upward irradiance. Net is global - upward, sum is global + upward.

**uu**
Radiances: uu(umu(0),phi(0)) ... uu(umu(0),phi(m)) ... uu(umu(n),phi(0)) ... uu(umu(n),phi(m))

**fdir, fglo, fdn, fup, f**
The direct, global, diffuse downward, diffuse upward, and total actinic flux.

**uavgdir, uavgglo, uavgdn, uavgup, uavg**
The Direct, global, diffuse downward, diffuse upward, and total diffuse mean intensity (= actinic flux / $4\pi$).

**albedo**
Albedo.

**heat**
Heating rate in K/day.

It is also possible to gain some information about the atmosphere and the clouds:

**p**
pressure [hPa], ,

**T, T_d**
temperature [K], dewpoint temperature [K]

**T_sur**
surface temperature [K]

**theta**
potential temperature [K]

**theta_e**
equivalent potential temperature [K]

**n_xxx**
number density of the gas xxx [$cm^{-3}$]

**rho_xxx**
mass density of the gas xxx [$kg/m^3$]

**mmr_xxx**
mass mixing ratio of the gas xxx [kg/kg]

**vmr_xxx**
volume mixing ratio of the gas xxx [$m^3/m^3$]

**rh**
relative humidity over water [percent]

**rh_ice**
relative humidity over ice [percent]

**c_p**
specific heat capacity of the air (humidity and temperature dependent)

**CLWC**
cloud liquid water content [kg/kg]

**CLWD**
cloud liquid water density [$g/m^3$]

**CIWC**
cloud ice water content [kg/kg]

**CIWD**
cloud ice water density [$g/m^3$]

**TCC**
total cloud cover [0-1]

where xxx is one of AIR, O3, O2, H2O, CO2, NO2, BRO, OCLO, or HCHO.

Default output is

```
 output_user lambda lambda, edir, edn, eup, uavgdir, uavgdn,
uavgup
```

for fdisort1, sdisort, and spsdisort, whereas the default for twostr is

```
 output_user lambda, edir, edn, eup, uavg.
```

The lines containing radiances and the output of rte_solver polradtran are not affected.

**ozone_column**

Obsolete, use dens_column instead.

**phi**

Azimuth output angles (in degrees) in increasing order.

```
 phi values
```

The radiance is output at phi and umu.

- Sensor in the North (looking South): 0 deg
- Sensor in the East (looking West): 90 deg
- Sensor in the South (looking North): 180 deg
- Sensor in the West (looking East): 270 deg

For all one-dimensional solvers the absolute azimuth does not matter, but only the relative azimuth phi-phi0.

**phi0**

Azimuth angle of the sun (0 to 360 degrees).

```
phi0 value
```

- Sun in the South: 0 degrees
- Sun in the West: 90 degrees
- Sun in the North: 180 degrees
- Sun in the East: 270 degrees

For all one-dimensional solvers the absolute azimuth does not matter, but only the relative azimuth phi-phi0.

**polradtran_aziorder**

Order of Fourier azimuth series

```
 polradtran_aziorder value
```

The value 0 (default for irradiance) is the azimuthally symmetric case. For radiance computation a higher order is required, thus the default for radiances is 4. This option is only relevant for rte_solver polradtran.

**polradtran_max_delta_tau**
Initial layer thickness for doubling; governs accuracy, 10E-5 should be adequate. Do not go beyond half the real precision, i.e. 10e-8 for REAL*8. Default 1.e-05.

```
polradtran_max_delta_tau value
```

This option is only relevant for rte_solver polradtran.

**polradtran_nstokes**
Number of Stokes parameters

```
polradtran_nstokes value
```

where value is one of

**1** for I (no polarization, default)

**2** for I,Q,U (Since V is very small in the atmosphere, it makes sense to compute only I,Q,U. This saves computation time and memory).

**3** for I,Q,U,V

Default is 1.

**polradtran_quad_type**
Type of quadrature used:

```
polradtran_quad_type type
```

where type is one of

**G** gaussian

**D** double gaussian,

**L** Lobatto

**E (default)** extra-angle(s), this must be used of polradtran is used in combination with umu. Will internally use Gaussian scheme (G). See also radtran documentation (libsrc_f/README.polRadtran).

Default E. This option is only relevant for rte_solver polradtran.

**polradtran_src_code**
Radiation sources included by

```
polradtran_src_code value
```

which may be

**0** none

**1** solar

**2** thermal

**3** both

Default 1. This option is only relevant for `rte_solver polradtran`.

**pressure**

The surface pressure (at the user-defined `altitude`) in hPa.

```
pressure value
```

The pressure profile as well as air, O2 and CO2 density profiles are scaled accordingly.

**pressure_out**

Specify the output levels in pressure coordinates. The syntax is

```
pressure_out p1 p2 ...
```

where '`p1 p2 ...`' are the output levels in hPa. The pressure output levels must be sorted in decreasing order. Output pressure levels must be within the range defined in the `atmosphere_file`. You can also use `toa` for `top of atmosphere` and `sur` for `surface altitude` and `cpt` for `cold point tropopause`.

**prndis**

Specify one or more integers between 1 and 7.

```
prndis value
```

Print various disort input and output in disorts own format. See `libsrc_f/DISORT2.doc` for more information. **Warning:** Produces a lot of output.

**quiet**

If specified, informative messages are turned off. See also `verbose`.

**radiosonde**

This option allows to change the temperature and pressure profile, and optionally to specify one or more density profiles. The entry in the input file looks like this:

```
radiosonde filename [gas_species] [unit] ...
```

Currently the following gas_species are included: ozone (O3), nitrogen dioxide (NO2), water vapor (H2O), bromine oxide (BRO), chlorine dioxide (OCLO), formaldehyde (HCHO), and carbon dioxide (CO2). Each gas species is identified by its abbrevations given in parentheses above. Unit is an optional argument to defines the unit of the density. The profiles can be given in particles per $cm^3$ (CM-3), in

particles per $m^3$ (M-3), as volume mixing ratio (VMR), as mass mixing ratio in kg/kg (MMR), or as relative humidity (RH) (only for water). The default unit is RH for water vapour, MMR for ozone, and CM3 for all other gases. The radiosonde file must have (2 + number of gases) columns:

**1** pressure in hPa

**2** temperature in Kelvin

**3, 4, ...** density of trace gas in the specified unit

A new z-grid will be calculated, starting at `altitude` and assuming a linear temperature variation between levels. The air density will be recalculated according to the ideal gas law, and the density of the well mixed gases O2 and CO2 will be scaled accordingly. The atmospheric data above the radiosonde data is taken from the `atmosphere_file` level by level, starting at the first pressure level above the radiosonde data. The z-grid of the `atmosphere file` in this height region is shifted accordingly. Also if the density in the radiosonde file is specified as -1 at a level, the value from the `atmosphere_file` is used. Possible calls are

```
radiosonde ../examples/radiosonde.dat
```

just in order to change the temperature and pressure profile, or

```
radiosonde ../examples/radiosonde2.dat H2O RH O3 MMR NO2
```

where water vapour density will be given as relative humidity, ozone as mass mixing ratio, and NO2 in $cm^{-3}$ (default).

**radiosonde_levels_only**

The atmosphere considered in the simulation has the same height range as the data in the `radiosonde`-file. No further levels are added above those. This option has only an effect in combination with `radiosonde`.

**rayleigh_crs**

Specify the Rayleigh cross section.

```
rayleigh_crs type
```

Choose between the following Rayleigh scattering cross sections.

**Bodhaine**

Bodhaine et al. (1999) Rayleigh scattering cross section.

**Nicolet**

Nicolet (1984) Rayleigh scattering cross section.

**Penndorf**

Penndorf (1957) Rayleigh scattering cross section.

Bodhaine et al. (1999) is default.

**rayleigh_depol**

Rayleigh depolarization factor.

```
rayleigh_depol value
```

The Rayleigh scattering phase function is $p(\mu) = a + b\mu^2$ where $a = 1.5(1 + \texttt{depol})/(2 + \texttt{depol})$ and $b = 1.5(1 - \texttt{depol})/(2 + \texttt{depol})$. By default the depolarization is calculated using the expressions from Bodhaine et al. (1999).

**rayleigh_tau_file**

Location of Rayleigh scattering optical depth file.

```
rayleigh_tau_file file
```

Usually, the Rayleigh scattering cross section is calculated from the air pressure provided in `atmosphere_file` (scaled with `pressure`). Use this parameter only if you really want to specify the optical depth directly (e.g. for a model intercomparison). The optical thickness profile may be either monochromatic or spectral. The format is exactly the same as for `molecular_tau_file`.

**raman**

The `raman` option includes single order rotational Raman scattering in the calculation. The solution treats Raman as a perturbation similar to the approaches of Vountas et al. (1998) and Spurr et al. (2008).

The `raman` option may only be used for spectral calculation.

A special radiative transfer solver, qdisort, is needed to solve the radiative transfer equation including Raman scattering. This solver is automagically invoked when specifying the `raman` option. It is thus not neccessary to set the `rte_solver`.

Please note that while the `raman` option has been extensively tested and verified, it is nevertheless a new option, hence, use it with care. Also it is not optimized for speed. It is thus computationally very expensive. Major speedups are planned for future releases.

**reflectivity**

Calculate transmission / reflectivity instead of absolute quantities. For irradiances / actinic fluxes the transmission T is defined as

$$T = \frac{E}{E_0 \cos\theta} \tag{6.5}$$

where $E$ is the irradiance / actinic flux, $E_0$ is the extraterrestrial flux, and $\theta$ is the solar zenith angle. The reflectivity R is defined as

$$R = \frac{\pi \cdot L}{E_0 \cos\theta} \tag{6.6}$$

where $L$ is the radiance, $E_0$ is the extraterrestrial flux, and $\theta$ is the solar zenith angle. Obviously, reflectivities do not depend on Sun-Earth distance. Please note the difference to `transmittance`.

**reverse**

Option for the strong and bold. Reverses the atmospheric input to the radiative transfer solvers. That is, the atmosphere is turned on the head. Yes, that is actually useful for some purposes. If you think you need this contact the author. Otherwise, do not use.

**rh_file**

File that defines a profile of relative humidity.

```
rh_file file
```

If specified, the water vapour profile in `atmosphere_file` is over-written. If -1 is specified at a level, the value from `atmosphere_file` is used.

**rpv_file**

4 column file, containing the Rahman, Pinty, and Verstraete (RPV) BDRF parameterization (Rahman et al., 1993a).

```
rpv_file file
```

Bidirectional reflectance distribution functions for a variety of surfaces are given in the paper. This option is only supported with solvers: `cdisort`, `fdisort2` . The columns of the input file are wavelength [nm], rho0, k, and theta. The parameters are interpolated linearly to the internal wavelength grid. To make sure that the results are reasonable, specify the RPV data on a wavelength grid similar or equal to that used internally for the radiative transfer calculation! Optionally, a fifth column with a constant scaling factor may be defined. If it has seven columns, the fifth to seventh are sigma, t1, t2, and if it has eight, the eighth is scale again.

**rpv_k**

Constant RPV k, see `rpv_file`.

```
rpv_k value
```

`rpv_k` overwrites the wavelength-dependent value defined in `rpv_file`.

**rpv_rho0**

Constant RPV rho0, see `rpv_file`.

```
rpv0 value
```

`rpv_rho0` overwrites the wavelength-dependent value defined in `rpv_file`.

**rpv_theta**

Constant RPV theta, see `rpv_file`.

```
rpv_theta value
```

`rpv_theta` overwrites the wavelength-dependent value defined in `rpv_file`.

**rpv_sigma**

Constant RPV sigma, to be used for snow (Degünther and Meerkötter, 2000).

```
rpv_sigma value
```

A wavelength dependent sigma is not yet available.

**rpv_t1**

Constant RPV t1, to be used for snow (Degünther and Meerkötter, 2000).

```
rpv_t1 value
```

A wavelength dependent sigma is not yet available.

**rpv_t2**

Constant RPV t2, to be used for snow (Degünther and Meerkötter, 2000).

```
rpv_t2 value
```

A wavelength dependent sigma is not yet available.

**rpv_scale**

Apply a constant scaling factor for the RPV BRDF.

```
rpv_scale value
```

Required e.g. if the the albedo should be set to a certain value. This factor is only used by `rte_solver cdisort`, `rte_solver fdisort2`.

**rpv_library**

The rpv libraries are collections of spectral BRDFs of different surface types, This option must be used either with `surface_type` or `surface_type_map`, in order to select the specific surface type.

For using a `rpv_library` write

```
rpv_library library_path
```

where `library_path` is the path of the directory, where the BRDF data is stored. The files are expected to have the names `IGBP.01.rpv, IGBP.02.rpv, ...` If `surface_type 1` is specified the BRDF from `IGBP.01.rpv` will be used, and so on. Each file must have the structure like an `rpv_file`. (This option is quite the same as `rpv_file`, except that it offers you an easy way to use the option `surface_type_map` in combinition with your `rpv_files`.)

```
rpv_library IGBP
```

The built-in library contains the first 17 surface types see `albedo_library`. The data is given for the wavelengths 443nm, 565nm, 670nm, and 865nm. Stay

near this wavelength in order to get reasonable results. In future this the rpv-library will be NDVI dependent, but until now the most common NDVI class is selected automatically.

**rte_solver**
Set the radiative transfer equation solver to be used.

```
rte_solver type
```

If not specified the default `rte_solver` is `cdisort`. Choices for `type` are

**cdisort**
C-version of the disort algorithm, translated from Fortran by Tim Dowling. This is the recommended discrete ordinate code in *libRadtran*. For documentation see `src_f/DISORT2.doc` as well as the papers and the DISORT report at ftp://climate1.gsfc.nasa.gov/wiscombe/Multiple_Scatt/. The intensity correction can be performed according to Nakajima and Tanaka (1988) using `disort_icm moments` (like in the original code), or with the improvements described in (Buras, Dowling, Emde, in preparation; default). Can be run in plane-parallel geometry (default) or in pseudo-spherical geometry (using `cdisort_pseudospherical`).

**ctwostr**
C-version of the two–stream radiative transfer solver described by Kylling et al. (1995). Can be run in plane-parallel geometry (default) or in pseudo-spherical geometry (using `ctwostr_pseudospherical` ).

**disort**
Same as `cdisort` which is the recommended discrete ordinate code in *libRadtran*. Before *libRadtran* 1.6 `cdisort` invoked the Fortran code disort 1.3.

**disort2**
Same as `cdisort` which is the recommended *discrete ordinate code* in *libRadtran*. Before *libRadtran* 1.6 `cdisort` invoked the Fortran code disort 2.0.

**fdisort1**
The standard plane–parallel disort algorithm by Stamnes et al. (1988), version 1.3 – provided for compatibility reasons. Use only if you have troubles with the default `disort` or for historical reasons. For documentation see `src_f/DISORT.doc` as well as the papers and the DISORT report at ftp://climate1.gsfc.nasa.gov/wiscombe/Multiple_Scatt/. To optimize for computational time and memory, please adjust the parameters in `src_f/DISORT.MXD` for your application and re-compile. For your application please use `rte_solver fdisort2` which is the advanced version, unless you e.g. want to explore how a specific feature of `fdisort2` (e.g. the Nakajima and Tanaka (1988) intensity correction) improves the `fdisort1` result.

**fdisort2**
Version 2 of the Fortran algorithm disort – provided for compatibility reasons.

Use only if you have troubles with the default `disort` or for historical reasons. For documentation see `src_f/DISORT2.doc` as well as the papers and the DISORT report at ftp://climate1.gsfc.nasa.gov/wiscombe/Multiple_Scatt/ `fdisort2` has several improvements compared to its 'ancestor' `fdisort1` (version 1.3). To optimize for computational time and memory, please adjust the parameters in `src_f/DISORT.MXD` for your application and re-compile.

Note! `fdisort2` is a new version of the original disort code which was implemented in summer 2009. It uses phase functions to calculate the intensity corrections by Nakajima and Tanaka (1988) instead of Legendre moments. Hence it needs cloud properties files which contain the phase functions. It is still possible to use the old version of disort2, you need to specify `disort_icm moments`.

**sdisort**

Pseudospherical disort as described by Dahlback and Stamnes (1991). Double precision version. To optimize for computational time and memory, please adjust the parameters in `src_f/DISORT.MXD` for your application and re-compile.

**spsdisort**

Pseudospherical disort as described by Dahlback and Stamnes (1991), single precision version. **Warning:** it is not recommended to use `spsdisort` for really large solar zenith angles nor for cloudy conditions. For large optical thickness it is numerically unstable and may produce wrong results. To optimize for computational time and memory, please adjust the parameters in `src_f/DISORT.MXD` for your application and re-compile.

**polradtran**

The plane-parallel radiative transfer solver of Evans and Stephens (1991). Includes polarization. The full implementation of the polRadtran solver in `uvspec` is quite new (version 1.4). If you find unusual behaviour, please contact the *libRadtran* authors.

**twostr**

The two–stream radiative transfer solver described by Kylling et al. (1995), in pseudo-spherical geometry.

**twostrpp**

The two–stream radiative transfer solver described by Kylling et al. (1995), in plane-parallel geometry.

**rodents**

Delta-Eddington two–stream code (RObert's Delta-EddingtoN Two-Stream), plane-parallel.

**sslidar**

A simple single scattering lidar simulator by Robert Buras.

**sos**

A scalar pseudospherical succesive orders of scattering code. Works for solar zenith angles smaller than 90 degrees. Can calculate azimuthally averaged radiances. Set `nscat` to specify the order of scattering.

**tzs**
TZS stands for "thermal, zero scattering" and is a very fast analytical solution for the special case of thermal emission in a non-scattering atmosphere. Please note that TZS does only radiance calculations at top of the atmosphere.

**sss**
SSS stands for "solar, single scattering" and is an analytical single scattering approximation which might be reasonable for an optically thin atmosphere. Please note that SSS does only radiance calculations at top of the atmosphere. This is an experimental solver - be careful!

**null**
The NULL solver does not solve the radiative transfer equation. However, it sets up the optical properties, and does the post-processing; useful if you are either interested in the overhead time required by a particular model input or if you are simply interested in the optical properties, as output by `verbose`.

Default: `cdisort`

**satellite_geometry**
With this option the satellite geometry is determinded. The argument for this option

```
satellite_geometry netCDF_file
```

is the location of a `netCDF_file`, which must contain latitude and longitude position as well as zenith and azimuth viewing angle for each pixel.

**satellite_pixel**
This option specifies which pixel of the satellite image that should be simulated.

```
satellite_pixel pixel_x pixel_y
```

The arguments `pixel_nr_x` and `pixel_nr_y` specifies the pixel position in the native system of the satellite, which is determinded by the option `satellite_geometry`.

**slit_function_file**
If specified, the calculated spectrum is convolved with the function found in the `slit_function_file`.

```
slit_function_file file
```

The file must contain two columns. Column 1 is the wavelength, in nm, and relative to the center wavelength. Column 2 is the corresponding slit function value. It must be unity at the maximum. The wavelength steps in the slit function file must be equidistant. Comments start with `#`. Empty lines are ignored. Please note that prior to convolution the spectrum is interpolated to the wavelength steps of the slit function. For this reason, make sure that the resolution of the slit function is high enough even if the slit function is e.g. a simple triangle which could in principle be described with 3 grid points. For an example see `examples/TRI_SLIT.DAT` and the `make_slitfunction` tool.

**solar_file**

Location of file holding the extraterrestrial spectrum.

```
solar_file filename [unit]
```

The file must contain two columns. Column 1 is the wavelength in nm, and column 2 the corresponding extraterrestrial flux. The user may freely use any units he/she wants for the extraterrestrial flux. The wavelength specified grid defines the wavelength resolution at which results are returned. However, the wavelength range is determined by `wavelength`. `solar_file` may be omitted for thermal radiation calculations (`source thermal`) as well as `transmittance` and `reflectivity` calculations. If omitted, the output resolution equals the internal wavelength grid which the model chooses for the radiative transfer calculation. Comments start with #. Empty lines are ignored.

For some purpuses it is useful to tell libRadtran the units of the spectrum. This can be done with the optional second argument. If `unit` is set to `per_nm` libRadtran assumes that the unit of the spectrum is W/(m2 nm), if set to `per_cm-1` it assumes W/(m2 cm-1). Note that `solar_file` is ignored if `correlated_k` is specified.

**source**

Set the radiation source type

```
source type
```

where `type` is either `solar` or `thermal`. Solar radiation is per default output in W/(m2 nm) for spectral and `correlated_k` LOWTRAN calculations. For all other `correlated_k` options the output is integrated over the wavelength band. Thermal radiation is per default output in W/(m2 cm-1), if the bandwidth is equal to 1 cm-1 (default for `correlated_k LOWTRAN` calculations). Otherwise the output is the integrated flux over the wavenumber interval specified by `thermal_bandwith`, `thermal_bands_file`, or by the `correlated_k` option (`kato`, `kato2`, `kato2.96`, `fu`, or `avhrr_kratz`).

**spline**

```
spline lambda_0 lambda_1 lambda_step
```

Spline interpolate the calculated spectrum between wavelengths `lambda_0` and `lambda_1` in steps of `lambda_step`, in nm. Specified as e.g.

```
spline 290.  365.  0.5
```

Here, the calculated spectrum is interpolated to wavelengths 290.0, 290.5, 291.0, ..., 364.5, 365.0. For interpolation to arbitrary wavelengths use `spline_file`. The specified wavelength interval must be within the one specified by `wavelength`.

**spline_file**

Spline interpolate to arbitrary wavelengths, in nm, given as a single column in file

spline_file.

```
spline_file file
```

The specified wavelengths must be within the range specified by wavelength. Comments start with #. Empty lines are ignored.

**sslidar_area**

Set area of single scattering lidar in units of square meters (solver sslidar).

**sslidar_E0**

Set Laser pulse energy for single scattering lidar in units of Joule (solver sslidar). (You can also use a solar_file instead... not yet implemented.)

**sslidar_eff**

Set lidar efficiency for single scattering lidar (solver sslidar).

**sslidar_nranges**

Set number of range bins for single scattering lidar (solver sslidar).

**sslidar_position**

Set lidar position for single scattering lidar in units of km (solver sslidar).

**sslidar_range**

Set lidar range bin width for single scattering lidar in units of km (solver sslidar).

**surface_temperature**

Surface temperature, used for thermal infrared calculations.

```
surface_temperature value
```

If not specified, the temperature of the lowest atmospheric level is used as surface temperature.

**surface_temperature_map**

Specify a surface_temperature map with a *netCDF* file which is used in combinition with the options latitude, longitude, and time.

```
surface_temperature_map file [variable_name]
```

where file is the location of the *netCDF* file. libRadtran reads the value at the nearest pixel to the given latitude and longitude. No spatial interpolation or averaging of the values is done.

**surface_type**

With this option the `surface_type` is selected. This option can be used with `albedo_library` in order to select a spectral albedo or with `rpv_library` in order to select a BRDF function.

```
surface_type surface_type_number
```

where `surface_type_number` is an integer starting from 0, where 0 refers to a black surface and the following numbers to the entries in the specified `library`.

**surface_type_map**

Specify a surface type map, which is used in combinition with `albedo_library`, `latitude`, and `longitude` in order to select the surface type relevant for the simulation. No pixel interpolation is done. The format of the call is:

```
surface_type_map file [variable_name]
```

where `file` is the location of the surface type map file. The map is expected to be in *netCDF* format. The file must contain the variables `double lat(nlat)`, `double lon(nlon)`, and `byte surface_type (nlat, nlon)`. If the name of the surface type variable is different, the optional argument can be used in order to specify the variable name. For format specification see also `data/albedo/IGBP_map/SURFACE_TYPE_IGBP_10min.cdf`.

For using the IGBP map, the call is `surface_type_map IGBP`. This map has a resolution of 10 minutes and contains the surface types 1 to 18 defined in the `albedo_library IGBP`. Fresh snow and sea ice are not included, as their extent is too variable. Attention: That implies e.g. that the Arctic is considered ocean_water and not sea_ice!

Locations on the pixel boundaries are interpreted as the pixel northward and eastward respectively. E.g. location 0 N, 0 E is interpreted like the pixel ranging from 0 to 10min North and from 0 to 10min East.

**sza**

The solar zenith angle (degrees).

```
sza value
```

The default solar zenith angle is 0.

**sza_file**

Location of solar zenith angle file for wavelength-dependent solar zenith angle.

```
sza_file file
```

This option is useful if you want to simulate an instrument which scans so slowly that the solar zenith angle may change significantly during the wavelength scan. The file must have two or three columns. Column 1 is the wavelength, in nm,

and column 2 the corresponding solar zenith angle. Optionally the third column may contain the corresponding solar azimuth angle. The solar azimuth angle is only needed when calculating radiances. The wavelength grid may be freely set. The solar zenith and azimuth angle will be interpolated to the wavelength grid used for the radiation calculation. Comments start with #. Empty lines are ignored.

**thermal_bands_file**

File with the center wavelengths and the wavelength band intervals to be used for calculations in the thermal range.

```
thermal_bands_file file
```

The following three columns are expected: center (or reference) wavelength, lower wavelength limit, upper wavelength limit [nm]. `thermal_bands_file` defines the wavelength grid for the radiative transfer calculation. The RTE solver is called for each of the wavelengths in the first column. The atmospheric (scattering, absorption, etc) properties are also evaluated at these wavelengths. For thermal radiation calculations, the Planck function is integrated over the wavelength bands defined in the second and third columns. The result will therefore be a band-integrated irradiance which does only make sense when the `solar_file` grid equals the `thermal_bands_file` grid.

**thermal_bandwidth**

Specify a constant bandwidth in cm-1 for thermal calculations.

```
thermal_bandwidth value
```

The default is 1 cm-1. This option is ignored if used together with `correlated_k kato/kato2/kato2.96/fu/avhrr_kratz`.

**time**

Specifies the time to simulate.

```
time YYYY MM DD hh mm ss
```

where `YYYY` is the year, `MM` the month, `DD` the day, `hh` the hour, `mm` the minute, `ss` the second in UTC. The time information will be used for a couple of things:

`time` in combination with `latitude`, `longitude`, and any `map`-option is used to select the location where to read the input data.

`time` is used to correct extraterrestrial irradiance for the Sun-Earth distance with the day of year. If not given, the Earth-Sun distance is 1 AU (i.e. equinox distance).

`time` in combination with `latitude` and `longitude` is used to calculate the solar zenith angle if no `sza` is specified.

`time` in combination with `latitude` and `longitude` is used to choose a suitable default atmosphere file, if no `atmosphere_file` is specified.

`time` in combination with an `ECMWF_atmosphere_file` is used to choose a date in the ECMWF input file.

**time_interpolate**

If a map option is used in combination with `time`, the data, which is nearest to the specified `time` is used for the simulation. This means `time_interpolate` is switched off per default.

If this option is switched on, the data fields stored in the *netCDF* files are interpolated to the specified `time`. (Be aware, that this might cause strange effects for data field of moving properties. E.g. an interpolated cloud field might have double horizontal extent, but only half the optical depth.)

**time_interval**

This option can be used in order to calculate an effective solar zenith angle for a time interval, instead of a distinct point in time. The cosine of the solar zenith angle is here replaced by its time average. The azimuth of the sun is replaced by an average of the azimuth position weighted with the cosine of the solar zenith angle.

```
 time_interval dtime_start dtime_end [unit] time_interval -180
180 min
```

the time interval reaches from `time` + `dtime_start` to `time` + `dtime_end`, in the example from 180 minutes before `time` to 180 minutes after `time`. The `unit` argument is optional, and can be one of the following: `s` (seconds), `min` (minutes), or `h` (hour). The default is `s`. This option makes only an effect in combination with `time`, `latitude`, `longitude`, and only has an effect for solar simulations (and of course if no `sza` defined).

**transmittance**

Calculate transmittance / reflectance instead of absolute quantities. That is, set the extraterrestrial irradiance to 1 and do not correct for Sun-Earth distance:

$$T = \frac{E}{E_0} \tag{6.7}$$

where $E$ is the irradiance / actinic flux / radiance and $E_0$ is the extraterrestrial flux. Please note the difference to `reflectivity`.

**transmittance_wl_file**

Location of single column file that sets the wavelength grid used for the internal transmittance calculations.

```
 transmittance_wl__file file
```

The wavelengths must be in nm. Do not use this option unless you know what you are doing. Comments start with `#`. Empty lines are ignored.

**umu**

Cosine of output polar angles in increasing order, starting with negative (downwelling radiance, looking upward) values (if any) and on through positive (upwelling radiance, looking downward) values. Must not be zero.

```
 umu values
```

**verbose**

If specified abundances of informative messages are output to stderr. To make use of this information, you may want to write the standard `uvspec` output to one file and the diagnostic messages to another. To do so, try `(./uvspec < uvspec.inp > uvspec.out) >& verbose.txt` (depending on your shell you might need a slightly different syntax). The irradiances and radiances will be written to `uvspec.out` while all diagnostic messages go into `verbose.txt`. See also `quiet`.

**wavelength**

Set the wavelength range by specifying first and last wavelength in nm.

```
wavelength lambda_0 lambda_1
```

The default output wavelength grid is that defined in `solar_file`, unless `spline` is specified. Note that the radiative transfer calculations are done on an internal grid which can be influenced with `transmittance_wl_file` or `molecular_tau_file`

**wavelength_index**

Set the wavelengths to be selected. To be used together with predefined wavelength grids, such as `transmittance_wl_file`, `molecular_tau_file` and particularly useful in combination with the `correlated_k` option where often only a specified number of wavelength bands is required. E.g., in combination with `correlated_k AVHRR_KRATZ`, `wavelength_index 15 15` will select wavelength index 15 which corresponds to channel 4, or `wavelength_index 10 14` will select those bands required for channel 3. Indices start from 1.

**wc_cloudcover**

Set the fraction of the horizontal sky area which is covered by clouds.

```
wc_cloudcover value
```

When a cloud cover is specified, the result will be calculated by the independent pixel approximation (IPA), that is, as weighted average of cloudless sky and overcast sky, where the cloud properties are taken from `wc_file`, etc. Please note that, if both `wc_cloudcover` and `ic_cloudcover` are set, both must be equal.

This option is ignored, if the option `cloud_fraction_file` is used.

**wc_file**

Location of file defining water cloud properties.

```
wc_file file
```

The file must contain three columns: Column 1 is the altitude in km, column 2 the liquid water content (LWC) in grams per cubic meter, and column 3 the

effective droplet radius in micrometer. Empty lines are ignored. Comments start with #. Note that the definition of cloud altitudes in wc_file refers to sea level, not to altitude above ground. E.g., when altitude is set to 1.63km, and the first cloud level is defined at 3km, the cloud would start at 1.37km above ground. An example of a cloud is given in examples/WC.DAT.

Per default the cloud properties are interpreted as layer properties. Before version 1.4 the default was level properties: The optical depth of a layer was calculated using information from the upper and lower levels defining the layer, see wc_layer and wc_level. To switch to the old behaviour, use wc_level. See section 3.3.4 about water clouds for a realistic example how the contents of the wc_file are converted to optical properties.

**wc_files**

A way to specify cloud extinction coefficient, single scattering albedo, and scattering phase function for each layer.

```
wc_files file
```

The file specified by wc_files has two columns where column 1 is the altitude in km. The second column is the name of a file which defines the optical properties of the layer starting at the given altitude. The files specified in the second column must have the following format:

Column 1: The wavelength in nm. These wavelengths may be different from those in solar_file. Optical properties are interpolated to the requested wavelengths.

Column 2: The extinction coefficient of the layer in units km-1.

Column 3: The single scattering albedo of the layer.

Column 4-(nmom+4): The moments of the scattering phase function.

Note that if using the rte_solver cdisort or rte_solver fdisort2 it makes good sense to make the number of moments larger than nstr. For rte_solver fdisort1 and rte_solver polradtran the number of moments included in the calculations will be nstr+1. Higher order moments will be ignored for these solvers. Please note that the uppermost line of wc_files denotes simply the top altitude of the uppermost layer. The optical properties of this line are consequently ignored. There are two options for this line: either an optical property file with zero optical thickness is specified or "NULL" is used.

**wc_ipa_files**

A two-column file, defining water cloud property files (see wc_file) in the first column and the corresponding weights in the second column.

```
wc_ipa_files file
```

The radiative transfer calculation is performed independently for each cloud column and the result is the weighted average of all independent columns.

If ic_ipa_files and wc_ipa_files are both defined, both must have the same columns in the same order, otherwise uvspec will complain. See examples/UVSPEC_WC_IPA_FILES.INP for an example.

**wc_layer**
Interpret cloud properties as layer properties (this is the default behaviour since version 1.4; see also wc_file). Cloud properties are assumed to be constant over the layer. The layer reaches from the level, where the properties are defined in the wc_file to the level above that one. For example, the following lines

```
#       z      LWC     R_eff
#     (km)   (g/m^3)  (um)
        4.000    0.0    0.0
        3.000    1.0   10.0
```

define a cloud in the layer between 3 and 4 km with sharp boundaries.

**wc_level**
Interpret cloud properties as level properties (this was the default behaviour before version 1.4; see also wc_file). If wc_level is defined, a wc_file would be interpreted as follows:

```
#       z      LWC     R\_eff
#     (km)   (g/m^3)  (um)
        5.000     0       0
        4.000    0.2    12.0
        3.000    0.1    10.0
        2.000    0.1     8.0
```

The value 0.2 g/m$^3$ refers to altitude 4.0km, as e.g. in a radiosonde profile. The properties of each layer are calculated as average over the adjacent levels. E.g. the single scattering properties for the model layer between 3 and 4km are obtained by averaging over the two levels 3km and 4km. To allow easy definition of sharp cloud boundaries, clouds are only formed if both liquid water contents above and below the respective layer are larger than 0. Hence, in the above example, the layers between 2 and 3 as well as between 3 and 4km are cloudy while those between 1 and 2km and between 4 and 5km are not.

**wc_no_scattering**
Switch off scattering by water clouds.

**wc_properties**
Define how liquid water content and effective droplet radius are translated to optical properties.

```
wc_properties type
```

Possible choices for type are

**hu**

Parameterization by Hu and Stamnes (1993); this is the default setting. Note that the parameterization is somewhat different for `correlated_k FU` than for all other cases because in the latter case the parameterization from the newer (March 2000) Fu and Liou code is used while otherwise the data are taken from the original paper by Hu and Stamnes (1993). Note that this parameterization has been developed to calculate irradiances, hence it is less suitable for radiances. This is due to the use of the Henyey-Greenstein phase function as an approximation of the real Mie phase function.

**echam4**

Use the very simple two-band parameterization of the ECHAM4 climate model, described in Roeckner et al. (1996); this is probably only meaningful if you want to compare your results with ECHAM4, the two bands are 0.2 - 0.68 micrometer and 0.68 - 4.0 micrometer; within these bands, the optical properties are assumed constant.

**mie**

Use pre-calculated Mie tables; useful for `correlated_k`; the tables are expected in `data_files_path/correlated_k/.../`.
For spectral or pseudo-spectral (`correlated_k sbdart`) calculations, a set of pre-calculated tables is also available. For spectral or pseudo-spectral calculations `wc_properties_interpolate` has to be defined explicitely to initiate the interpolation of the optical properties to the internal wavelength grid. The Mie tables are not part of the standard distribution (because of their large size) but they are freely available from http://www.libradtran.org. This is the correct option to calculate radiances, to be preferred over the Henyey-Greenstein approach of Hu and Stamnes (1993).

**filename**

Read optical properties from specified filename; file format is as produced by the `mie`-tool of the *libRadtran* package (see `output_user netcdf`).

**wc_properties_interpolate**

Interpolate water cloud optical properties over wavelength; useful for precalculated optical property files defined with `wc_properties`. Please note that this option may be extremely memory-consuming because for each internal wavelength a full set of Legendre moments of the phase function is stored (up to several thousands).

**wc_saturate**

With this option, the relative humidity inside water clouds can easily be adjusted to a user-defined value, e.g. saturated with respect to water. This option has one nessesary and one optional argument:

```
wc_saturate switch [relative_humidity]
```

where `switch` is `on`, `off`, or `ipa`. The second optional argument determines the relative humidity (with respect to water!) inside the cloud. The default value is 100. `ipa` is only relevant for independent column calculations. The

following details apply for independent column simulations: Using `switch on`, the air in *all* columns will be saturated, if there is a cloud in at least one of the columns (this option should be used for stratiform clouds). Using `switch ipa`, only cloudy columns are affected (this option should be used for convective cloud fields.)

**wc_scale_gg**

Scale the water cloud asymmetry factor for all wavelengths and altitudes with a float between 0.0 and 1.0.

```
wc_scale_gg value
```

**wc_scale_ssa**

Scale the water cloud single scattering albedo for all wavelengths and altitudes with a float between 0.0 and 1.0.

```
wc_scale_ssa value
```

**wc_set_gg**

Set the water cloud asymmetry factor for all wavelengths and altitudes to a float between -1.0 and 1.0. Please note that this option is only applied if a Henyey-Greenstein phase function is used but not if an explicit phase function is defined e.g. with a wc_files. It doesn't make sense to modify only the first moment of an explicit phase function.

```
wc_set_gg value
```

This option is useful only for monochromatic calculations or in wavelength regions where the optical properties of water clouds can be considered constant, e.g. the ultraviolet range.

**wc_set_ssa**

Set the water cloud single scattering albedo for all wavelengths and altitudes to a float between 0.0 and 1.0.

```
wc_set_ssa value
```

This option is useful only for monochromatic calculations or in wavelength regions where the optical properties of water clouds can be considered constant, e.g. the ultraviolet range.

**wc_set_tau**

Set the total water cloud optical thickness to a constant value for all wavelengths.

```
wc_set_tau value
```

The optical thickness defined here is the integral from the surface at the user-defined `altitude` to TOA (top of atmosphere). This option is useful only for monochromatic calculations or in wavelength regions where the optical properties of water clouds can be considered constant, e.g. the ultraviolet range.

**wc_set_tau550**

Set the water cloud optical thickness at 550nm.

```
wc_set_tau550 value
```

The optical thickness defined here is the integral from the surface at the user-defined `altitude` to TOA (top of atmosphere). Other wavelengths are scaled accordingly. Note that this option requires for technical reasons that the wavelength interval defined by `wavelength` does contain 550nm.

**wvn**

Deprecated option. Same as `wavelength`.

**zout**

This option is used to specify the output altitudes in km *above surface altitude*. One or more altitudes may be specified in increasing magnitude.

```
zout 0 1 2 3 4 5 ...
```

Output altitudes must be within the range defined in the `atmosphere_file`. Note that `zout` does not restructure the atmosphere model. Hence, if you specify `zout 0.730` and have your atmosphere model in `atmosphere_file` go all the way down to sea level, i.e. 0.0km., output is presented at 0.730km and calculations performed with an atmosphere between 0.0 and 0.730 km (and above of course). If you want calculations done for e.g. an elevated site you have to restructure the atmosphere model and make sure it stops at the appropriate altitude. This you may either due by editing the atmosphere file or by using `altitude`. Note that for `rte_solver polradtran` the atmosphere file must contain the altitudes specified by `zout`. You can also use `toa` for `top of atmosphere` and `sur` for `surface altitude` and `cpt` for `cold point tropopause`.

Instead of specifying the altitudes in km, it is also possible to use keywords as argument for this option. Possible keywords are `atm_levels`, `all_levels`, `model_levels`, `model_layers`, and `model_levels_and_layers`. For `atm_levels`, all levels from the `atmosphere_file` are used as output levels. For `all_levels`, all levels (including levels from `atmosphere_file`, `dens_file`, cloud files, `altitude` options) are used as output levels. For `model_levels`, `model_layers`, `model_levels_and_layers` the levels, layers, or both from the `ECMWF_atmosphere_file` are used as output level. Usage e.g.:

```
zout model_levels [nlev_max]
```

With the optional argument `nlev_max` the user may specify the number of

zout layers from the ground.

**zout_sea**
like zout, but *above sea surface*

**zout_sur**
Same as `zout`.

**z_interpolate**
The profile in the `atmosphere_file` provides the constituents of the atmosphere at the given levels. Where additional levels are introduced and in order to calculate layer properties, an assumption about the variation of the property within the layer is required. These interpolation methods can be changed by the `z_interpolate` option. Two arguments are required, the property, and the interpolation method:

```
z_interpolate property interpolation_method
```

Properties which may be specified are:

**O3, O2, H2O, CO2, NO2, BRO, OCLO, HCHO**

**T**
temperature (here linmix is not suitable)

Possible interpolation methods are:

**linear**
The specified property (number density of the gas or temperature) varies linearly with height.

**log**
The specified property (number density of the gas or temperature) varies logarithmically with height. This is a reasonable option for all well mixed trace gases.

**linmix**
This option is only possible for gas profiles. The mixing ratio of the gas (assuming a logarithmically varying air density) varies linearly with height.

For all gas densities the default interpolation method is `linmix`, for temperature it is `linear`.

**zout_interpolate**
The z-grid of optical properties is determined by the `atmosphere_file`, and, if specified, by other profile files like `dens_file`, `rh_file`, or `refractive_index_file`. Additional levels might be introduced by the `zout` option and the second argument of the `altitude` option. By default (if `zout_interpolate` is not specified) levels introduced by the `zout` option will not affect the optical property profiles, that is, the optical properties

are constant within the layers specified by the `atmosphere_file` and profile files. If `zout_interpolate` is specified, the atmospheric profiles (tracegases, temperature ...) are interpolated to the levels introduced by `zout`, and optical properties are determined from the interpolated atmospheric properties. If `heating_rate`, `rte_solver polradtran`, `rte_solver rodents`, or `rte_solver twostrebe` is specified, `zout_interpolate` will also be automatically activated. `zout_interpolate` generally causes smoother variation of the optical properties.

## 6.2 Tool for Mie calculations - mie

The various input parameters of the mie tool are described in the following.

**aerosol_type**

With this option Mie calculations are performed for the specified aerosol type.

```
aerosol_type type
```

The aerosol properties (refractive index, size distribution, density, humidity) are taken from the OPAC database (Hess et al., 1998) Possible values for type are

**inso**

Water insoluble aerosol consists mostly of soil particles with a certain amount of organic material.

**waso**

Water soluble aerosol originates from gas to particle conversion and consists of various types of sulfates, nitrates, and other, also organic water-soluble substances.

**soot**

Soot is absorbing black carbon, which is not soluble in water. In reality soot particles have a chain-like character, which of course is not accounted for in Mie calculations of optical properties. The optical properties are calculated assuming many very small spherical particles.

**ssam**

Sea salt particles consist of the various kinds of salt contained in seawater. The different modes are given to allow for a different wind-speed-dependant increase of particle number for particles of different size. This aerosol type represents the accumulation mode.

**sscm**

Sea salt particles (coarse mode).

**minm**

Mineral aerosol or desert dust is produced in arid regions. It consists of a mixture of quartz and clay minerals and is modeled with three modes to allow to consider increasing relative amount of large particles for increased turbidity. This aerosol type represents the nucleation mode.

**miam**

Mineral aerosol (accumulation mode).

**micm**

Mineral aerosol (coarse mode).

**mitr**

Mineral transported is used to describe desert dust that is transported over long distances with a reduced amount of large particles.

**suso**
The sulfate component is used to describe the amount of sulfate found in the Antarctic aerosol. This component is not suited to describe antropogenic sulfate aerosols that are included in the water-soluble component.

**basename**
Filename for output of Mie program.

```
basename filebase
```

This option is only used in combination with `output_user netcdf`. The default is wc. for water, ic. for ice, or waso., inso. etc. for OPAC aerosols.

**distribution**
If specified the effective radius is converted into a size distribution of droplets.

```
distribution distribution_type distribution_parameter
```

where distribution type is one of the two following:

**GAMMA**
The Gamma distribuition of cloud droplet sizes is

$$n(r) = ar^{\alpha} \exp(-br), \tag{6.8}$$

where $\alpha$ is the distribution parameter given as second argument. ($a$ and $b$ are determined automatically.) The effective radius of the distribution is $r_{\text{eff}} = (\alpha + 3)/b$. A typical value for water clouds is $\alpha = 7$. For ice clouds a typical value is $\alpha = 1$. A large value of $\alpha$ gives close to a monodisperse distribution.

**LOGNORMAL**
The lognormal distribution of cloud droplet sizes is

$$n(r) = \frac{a}{r} \exp\left(\frac{-\ln(r/r_0)^2}{2\sigma^2}\right), \tag{6.9}$$

where $r_0$ is the logarithmic mode of the distribution (calculated automatically) and $\sigma$ is the standard deviation, which is given by the second argument.

**dx_max**
This option makes sense in combination with `distribution GAMMA` or `distribution LOGNORMAL`. It can be used to specify the maximum widths of the size distribution bins, which are sampled on a size parameter ($\frac{2\pi r}{\lambda}$) grid. The default value is 0.03 which is not very accurate for small wavelengths. In order to get accurate phase matrices this value should be decreased.

**mass_density**
Specifies the mass density of the medium.

```
mass_density value
```

Useful in combinition with `refrac user` and `output_user cloudprop`, as the format of `cloudprop` specifies the extinction coefficient per mass and not per volume as usual in this mie program.

**mie_program**

Specify which Mie program to use:

```
mie_program type
```

where `type` is one of

**BH**

The Mie scattering program by Bohren and Hoffmann, ftp://ftp.astro.princeton.edu/draine/scat/bhmie/bhmie.f

**MIEV0**

The Mie scattering program by W. Wiscombe. For documentation see `libsrc_f/MIEV.doc` and the NCAR Mie report at ftp://climate1.gsfc.nasa.gov/wiscombe/Single_Scatt/Homogen_Sphere/Exact_Mie/

**mimcut**

(positive) value below which imaginary refractive index is regarded as zero (computation proceeds faster for zero imaginary index). Only used by `mie_program MIEV0`.

```
mimcut value
```

**nmom**

Number of moments of the phase function to be calculated (default: 0).

```
nmom value
```

Only possible with `mie_program MIEV0`.

**nmom_netcdf**

Specify the number of Legendre polynomials that are written to the netcdf file.

```
nmom_netcdf value
```

This option only makes sense if `output_user netcdf` is specified. If not specified, all polynomials are written. For the calculation of the phase function all polynomials are of course considered.

**n_r_max**

This option makes sense in combination with `distribution GAMMA` or `distribution LOGNORMAL`. It defines the upper cutoff value for the size distribution in terms of effective radius $r_{\text{eff}}$. The default is 5, which means that the size distribution is cut off at a value of $5 \cdot r_{\text{eff}}$. This value should be increased if only small $r_{\text{eff}}$ are calculated.

**nstokes**

Number of Stokes parameters (default: 1).

```
nstokes value
```

For `nstokes=1` the Legendre polynomials of the phase function will be calculated. To calculate all phase matrix elements required for polarized radiative transfer, set `nstokes=4`.

**nthetamax**

Specify the maximum number of scattering angles to be used to sample the phase matrix.

```
nthetamax value
```

The default value is 1000. If the accuracy of the phase function is less than 1% for `nthetamax` angles a warning is printed to the screen. The option is only meaningful in combination with `output_user netcdf`, otherwise phase functions are not computed.

**output_user**

The mie output is one line of output quantities to standard output (stdout) for each wavelength and each particle radius. With this option the user may specify the columns desired for output:

```
output_user output_1 output_2 ... output_n
```

where output_i is one of following arguments:

**lambda**
Wavelength in nm.

**wavenumber**
Wave number in cm-1.

**r_eff**
particle radius in micro meter.

**refrac_real**
The real part of the refractive index.

**refrac_imag**
The imaginary part of the refractive index.

**qext**
The extinction efficiency factor, if `r_eff` was specified, or the extinction coefficient [km-1] per unit concentration [$cm^3/m^3$], if a `size_distribution_file` was specified. If the medium is liquid water, 1 $cm^3/m^3$ equals a liquid water content of $1 g/m^3$ because the density of water is close to 1 $g/cm^3$. For ice and other substances, the density has to be considered (0.917 $g/cm^3$ for ice at 273K).

**qsca**
The scattering efficiency factor, if r_eff was specified, or the extinction coefficient [km-1] per unit concentration [$cm^3/m^3$], if a size_distribution_file was specified.

**omega**
The single scattering albedo.

**gg**
The asymmetry parameter.

**sforw**
(Complex) forward-scattering amplitude S1 at 0 degrees.

**sback**
(Complex) back-scattering amplitude S1 at 180 degrees.

**spike**
To quote from Wiscombe's MIEV0.doc:

> (REAL) magnitude of the smallest denominator of either Mie coefficient (a-sub-n or b-sub-n), taken over all terms in the Mie series past N = size parameter XX. Values of SPIKE below about 0.3 signify a ripple spike, since these spikes are produced by abnormally small denominators in the Mie coefficients (normal denominators are of order unity or higher). Defaults to 1.0 when not on a spike. Does not identify all resonances (we are still working on that).

Meaningless if a size_distribution_file was specified.

**pmom**
The nmom+1 moments (from 0 to nmom, see option nmom) of the phase function. The phase function $p(\mu)$ is

$$p(\mu) = \sum_{m=0}^{\infty} (2m+1) \cdot k_m \cdot P_m(\mu) \tag{6.10}$$

where $k_m$ is the m'th moment and $P_m(\mu)$ is the m'th Legendre polynomial.

**cloudprp**
This is a special option which, if specified, must be the only option of output_user, as cloudprp specifies a whole format of the output. In particular this option is useful when a correlated-k wavelength grid is specified with wavelength. If specified, the output will be written in a format, which can be directly used by *libRadtran*. See also output netcdf. See uvspec-options ic_properties and wc_properties and there the items mie and filename.

**aerosolprp**
This option is similar to the cloudprp option. The only difference is that the effective radius dimension is replaced by humidity values of the aerosol.

**netcdf**
This option writes the output to a netCDF file which can be used by uvspec using the options ic_properties and wc_properties.

The default output is:

```
lambda refrac_real refrac_imag qext omega gg spike pmom
```

**r_eff**

The radius [micron] of the particle to calculate single scattering properties of. Used together with the wavelength information to calculate the Mie size parameter.

```
r_eff radius
```

The user can optionally specify a 2nd and 3rd argument to make a loop over several radii:

```
r_eff radius_min radius_max radius_step
```

First calculations is done with radius_min, which will be increased by radius_step until radius_max is reached.

**refrac**

Specify which refractive index to use.

```
refrac type
```

The following choices for `type` are valid:

**ice**

The complex refractive index is taken from the REFICE function of W. Wiscombe.

**water**

The complex refractive index is taken from the REFWAT function of W. Wiscombe.

**user**

<re> <im> A user defined refractive index. re and im are the real and imaginary parts (both positive numbers).

**file**

<filename> Read refractive index from a three-column file containing wavelength [nm], and the real and imaginary parts of the refractive index (both positive numbers). The Mie calculation is done for each wavelength defined here.

**size_distribution_file**

Specify a two column file, r [micron], n(r), which describes a size distribution of droplets.

```
size_distribution_file file
```

The Mie calculation is repeated for each value of r found in the size distribution file, and the final result is a weighted average of these values. The user himself therefore has to choose a set of r's suited for his specific purpose.

**temperature**

Ambient temperature, used to calculate the refractive indices of water and ice.

```
temperature value
```

Temperature dependence is only considered above 10 micron (water) and 167 micron (ice), respectively. Default: 300K.

**verbose**

If specified abundances of informative messages are output to stderr. To make use of this information, you may want to write the standard mie output to one file and the diagnostic messages to another. To do so, try `(mie < mie.inp > mie.out) >& verbose.txt` (depending on your shell you might need a slightly different syntax).

**wavelength**

Sets the wavelength range, in nm.

```
wavelength lambda_min lambda_max
```

The wavelength step is specified by `wavelength_step`. For unregular wavelength grid it is also possible to specify a file, where the wavelength grid is stored.

```
wavelength wvl_filename
```

where `wvl_filename` is the path and name of the file, which contains the wavelength grid. It is expected that the wavelength values in nm are stored in the second column. For the `correlated_k` schemes implemented in `uvspec` you can use following abbrevations instead of a filename (in this case `data_files_path` must be specified also): `kato`, `kato2`, `kato2.96`, `fu`, and `avhrr_kratz`. This option is ignored if `refrac file` is specified.

**wavelength_step**

The wavelength step, in nm. Ignored if `refrac file` is specified.

```
wavelength_step value
```

**wavelength_index**

Set the wavelengths to be selected. This might be the normal wavelength grid defined by `wavelength` and `wavelength_step` or a `correlated_k` wavelength grid. E.g., in combination with `wavelength AVHRR_KRATZ`, `wavelength_index 15 15` will select wavelength index 15 which corresponds to channel 4, or `wavelength_index 10 14` will select those bands required for channel 3. Indices start from 1.

# Bibliography


Anderson, G., Clough, S., Kneizys, F., Chetwynd, J., and Shettle, E.: AFGL atmospheric constituent profiles (0-120 km), *Tech. Rep. AFGL-TR-86-0110*, Air Force Geophys. Lab., Hanscom Air Force Base, Bedford, Mass., 1986.

Baldridge, A. M., Hook, S. J., Grove, C. I., and Rivera, G.: The ASTER spectral library version 2.0, Remote Sens. Environ., 113, 711–715, doi:10.1016/j.rse.2008.11.007, 2009.

Bass, A. M. and Paur, R. J.: The ultraviolet cross–section of ozone, I, The measurements, in: Atmospheric Ozone: Proceedings of the Quadrennial Ozone Symposium, edited by Zerefos, C. S. and Ghazi, A., pp. 601–606, D. Reidel, Norwell, Mass., 1985.

Baum, B., Heymsfield, A., Yang, P., and Bedka, S.: Bulk scattering models for the remote sensing of ice clouds. Part 1: Microphysical data and models, J. of Applied Meteorology, 44, 1885–1895, 2005a.

Baum, B., Yang, P., Heymsfield, A., Platnick, S., King, M., Hu, Y.-X., and Bedka, S.: Bulk scattering models for the remote sensing of ice clouds. Part 2: Narrowband models, J. of Applied Meteorology, 44, 1896–1911, 2005b.

Baum, B., Yang, P., Nasiri, S., Heidinger, A., Heymsfield, A., and Li, J.: Bulk scattering properties for the remote sensing of ice clouds. Part 3: High resolution spectral models from 100 to 3250 cm-1, J. of Applied Meteorology, 46, 423–434, 2007.

Belward, A. and Loveland, T.: The DIS 1-km land cover data set, GLOBAL CHANGE, The IGBP Newsletter, 27, 1996.

Blanco-Muriel, M., Alarcón-Padilla, D. C., López-Moratella, T., and Lara-Coira, M.: Computing the solar vector, Solar Energy, 70, 431–441, 2001.

Bodhaine, B. A., Wood, N. B., Dutton, E. G., and Slusser, J. R.: On Rayleigh optical depth calculations, J. Atm. Ocean Technol., 16, 1854–1861, 1999.

Bohren, C. F. and Huffman, D. R.: Absorption and scattering of light by small particles, Wiley, second edition edn., 1998.

Chandrasekhar, S.: Radiative Transfer, Dover, New York, 1960.

cheng Ou, S. and Liou, K.-N.: Ice microphysics and climatic temperature feedback, Atmospheric Research, 35, 127–138, 1995.

Cox, C. and Munk, W.: Measurement of the roughness of the sea surface from photographs of the sun's glitter, Journal of the Optical Society of America, 44, 838–850, 1954a.

Cox, C. and Munk, W.: Statistics of the sea surface derived from sun glitter, Journal of Marine Research, 13, 198–227, 1954b.

Dahlback, A.: Measurements of biologically effective UV doses, total ozone abundances, and cloud effects with multichannel, moderate bandwidth filter instruments, Appl. Opt., 35, 6514–6521, 1996.

Dahlback, A. and Stamnes, K.: A new spherical model for computing the radiation field available for photolysis and heating at twilight, Planet. Space Sci., 39, 671–683, 1991.

Degünther, M. and Meerkötter, R.: Effect of remote clouds on surface UV irradiance, Annales Geophysicae, 18, 679–686, 2000.

Edwards, D. P.: GENLN2: A general line-by-line atmospheric transmittance and radiance model: Version 3.0 description and users guide, National Center for Atmospheric Research (NCAR), NCAR/TN-367+STR, Boulder, Colorado, 1992.

Evans, K. F. and Stephens, G. L.: A new polarized atmospheric radiative transfer model, J. Quant. Spectrosc. Radiat. Transfer, 46, 413–423, 1991.

Fu, Q.: An accurate parameterization of the solar radiative properties of cirrus clouds for climate models, J. of Climate, 9, 2058–2082, 1996.

Fu, Q. and Liou, K.: On the correlated k-distribution method for radiative transfer in non-homogeneous atmospheres, J. Atmos. Sci., 49, 2139–2156, 1992.

Fu, Q. and Liou, K.: Parameterization of the radiative properties of cirrus clouds, J. Atmos. Sci., 50, 2008–2025, 1993.

Fu, Q., Yang, P., and Sun, W. B.: An accurate parameterization of the infrared radiative properties of cirrus clouds for climate models, J. of Climate, 11, 2223–2237, 1998.

Hess, M., Koepke, P., and Schult, I.: Optical Properties of Aerosols and Clouds: The Software Package OPAC, Bulletin of the American Meteorological Society, 79, 831–844, 1998.

Hu, Y. X. and Stamnes, K.: An accurate parameterization of the radiative properties of water clouds suitable for use in climate models, J. of Climate, 6, 728–742, 1993.

Hu, Y. X. and Stamnes, K.: Climate sensitivity to cloud optical properties, Tellus, 52B, 81–93, 2000.

Kato, S., Ackerman, T. P., Mather, J. H., and Clothiaux, E.: The $k$–distribution method and correlated–$k$ approximation for a shortwave radiative transfer model, J. Quant. Spectrosc. Radiat. Transfer, 62, 109–121, 1999.

Key, J. R., Yang, P., Baum, B. A., and Nasiri, S. L.: Parameterization of shortwave ice cloud optical properties for various particle habits, J. Geophys. Res., 107, doi:10.1029/2001JD000 742, 2002.

Kratz, D. P. and Varanasi, P.: The correlated k-distribution technique as applied to the AVHRR channels, J. Quant. Spectrosc. Radiat. Transfer, 53, 501–517, 1995.

Kuo, K.-S., Weger, R. C., Welch, R. M., and Cox, S. K.: The Picard iterative approximation to the solution of the integral equation of radiative transfer-part II: Three-dimensional geometry, J. Quant. Spectrosc. Radiat. Transfer, 55, 195–213, 1996.

Kurucz, R.: Synthetic infrared spectra, in: Proceedings of the 154th Symposium of the International Astronomical Union (IAU); Tucson, Arizona, March 2-6, 1992, Kluwer, Acad., Norwell, MA, 1992.

Kylling, A.: Radiation transport in cloudy and aerosol loaded atmospheres, Ph.D. thesis, Alaska Univ., Fairbanks., 1992.

Kylling, A. and Stamnes, K.: Efficient yet accurate solution of the linear transport equation in the presence of internal sources: the exponential–linear–in–depth approximation, J. Com. Phys., 102, 265–276, 1992.

Kylling, A., Stamnes, K., and Tsay, S.-C.: A reliable and efficient two–stream algorithm for spherical radiative transfer: documentation of accuracy in realistic layered media , J. of Atmospheric Chemistry, 21, 115–150, 1995.

Kylling, A., Webb, A. R., Gobbi, R. K. G. P., Ammannato, L., Barnaba, F., Bais, A., Wendisch, S. K. M., Jäkel, E., Schmidt, S., Kniffka, A., Thiel, S., Junkermann, W., Blumthaler, M., Silbernagl, R., Schallhart, B., Scmitt, R., Kjeldstad, B., Thorseth, T. M., Scheirer, R., and Mayer, B.: Spectral actinic flux in the lower troposphere: measurement and 1-D simulations for cloudless, broken cloud and overcast situations, Atmos. Chem. Phys., 5, 1975–1997, 2005.

M., D., Brion, J., Charbonnier, J., and Malicet, J.: Ozone UV spectroscopy I: Absorption cross-sections at room temperature, J. of Atmospheric Chemistry, 15, 145–155, 1992.

Madronich, S.: Photodissociation in the atmosphere 1. Actinic flux and the effects of ground reflections and clouds, J. Geophys. Res., 92, 9740–9752, 1987.

Malicet, J., Daumont, D., Charbonnier, J., Parisse, C., Chakir, A., and Brion, J.: Ozone UV spectroscopy. II. Absorption cross–sections and temperature dependence, J. of Atmospheric Chemistry, 21, 263–273, 1995.

Mayer, B. and Kylling, A.: Technical note: The libRadtran software package for radiative transfer calclations – description and examples of use, Atmos. Chem. Phys., 5, 1855–1877, 2005.

Mayer, B., Seckmeyer, G., and Kylling, A.: Systematic long–term comparison of spectral UV measurements and UVSPEC modeling results, J. Geophys. Res., 102, 8755–8767, 1997.

Mayer, B., Kylling, A., Madronich, S., and Seckmeyer, G.: Enhanced absorption of UV radiation due to multiple scattering in clouds: Experimental evidence and theoretical explanation, J. Geophys. Res., 103, 31,241–31,254, 1998.

Mihalas, D.: Stellar atmospheres, The University of Chicago Press, iSBN 0-7167-0359-9, 1978.

Mishchenko, M. I.: Vector radiative transfer equation for arbitrarily shaped and arbitrarily oriented particles: a microphysical derivation from statistical electromagnetics, Appl. Opt., 41, 7114–7134, 2002.

Molina, L. T. and Molina, M. J.: Absolute absorption cross sections of ozone in the 185– to 350–nm wavelength range, J. Geophys. Res., 91, 14,501–14,508, 1986.

Nakajima, T. and Tanaka, M.: Effect of wind-generated waves on the transfer of solar radiation in the atmosphere-ocean system, J. Quant. Spectrosc. Radiat. Transfer, 29, 521–537, 1983.

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

Nicolet, M.: On the molecular scattering in the terrestrial atmosphere: An empirical formula for its calculation in the homosphere, Planet. Space Sci., 32, 1467–1468, 1984.

Penndorf, R.: Tables of the refractive index for standard air and the Rayleigh scattering coefficient for the spectral region between 0.2 and 20.0 $\mu$ and their application to atmospheric optics, J. Opt. Soc. Am., 47, 176–182, 1957.

Rahman, H., Pinty, B., and Verstraete, M. M.: Coupled surface–atmosphere reflectance (CSAR) model 2. semiempirical surface model usable with NOAA advanced very high resolution radiometer data, J. Geophys. Res., 98, 20,791–20,801, 1993a.

Rahman, H., Verstraete, M., and Pinty, B.: Coupled surface-atmosphere reflectance (CSAR) model. 1. Model description and inversion on synthetic data, Journal of Geophysical Research, 98, 20 779–20 789, 1993b.

Rees, M. H.: Physics and chemistry of the upper atmosphere. !. Upper atmosphere, Cambridge University Press, iSBN 0-521-32305-3, 1989.

Reif, F.: Statistical and thermal physics, McGraw-Hill Inc., iSBN 0-07-Y85615-X, 1965.

Ricchiazzi, P., Yang, S., Gautier, C., and Sowle, D.: SBDART: A research and Teaching software tool for plane-parallel radiative transfer in the Earth's atmosphere, Bulletin of the American Meteorological Society, 79, 2101–2114, 1998.

Roeckner, E., Arpe, K., Bengtsson, L., Christoph, M., Claussen, M., andE. Esch, L. D., Giorgetta, M., Schlese, U., and Schulzweida, U.: The atmospheric general circulation model ECHAM-4: model description and simulation of present-day climate, Tech. rep., Max Planck-Institut für Meteorologie, Report No. 218, 1996.

Rottmann, K.: Mathematische Formelsammlung, B.I.-Hochschultaschenbuch, iSBN 3-411-70134-X, 1991.

Shettle, E.: Models of aerosols, clouds and precipitation for atmospheric propagation studies, in: Atmospheric propagation in the uv, visible, ir and mm-region and related system aspects, no. 454 in AGARD Conference Proceedings, 1989.

Spencer, J. W.: Fourier series representation of the position of the sun, Search, 2, No. 5, 1971.

Spurr, R., de Haan, J. D., van Oss, R., and Vasilkov, A.: Discrete ordinate radiative transfer in a stratified medium with first order rotational Raman scattering, J. Quant. Spectrosc. Radiat. Transfer, 109, 404–425, 2008.

Stamnes, K.: The Theory of Multiple Scattering of Radiation in Plane Parallel Atmospheres, Reviews of Geophysics, 24, 299–310, 1986.

Stamnes, K., Tsay, S.-C., Wiscombe, W., and Jayaweera, K.: Numerically stable algorithm for discrete–ordinate–method radiative transfer in multiple scattering and emitting layered media, Appl. Opt., 27, 2502–2509, 1988.

Stamnes, K., Slusser, J., and Bowen, M.: Derivation of total ozone abundance and cloud effects from spectral irradiance measurements, Appl. Opt., 30, 4418–4426, 1991.

Stamnes, K., Tsay, S.-C., Wiscombe, W., and Laszlo, I.: DISORT, a General-Purpose Fortran Program for Discrete-Ordinate-Method Radiative Transfer in Scattering and Emitting Layered Media: Documentation of Methodology, Tech. rep., Dept. of Physics and Engineering Physics, Stevens Institute of Technology, Hoboken, NJ 07030, 2000.

Thomas, G. E. and Stamnes, K.: Radiative Transfer in the Atmosphere and the Ocean, Cambridge University Press, iSBN 0521890616, 1999.

Vountas, M., Rozanov, V. V., and Burrows, J. P.: Ring effect: impact of rotational Raman scattering on radiative transfer in Earht's atmosphere, J. Quant. Spectrosc. Radiat. Transfer, 60, 943–961, 1998.

Wanner, W., Strahler, A., Hu, B., Lewis, P., Muller, J.-P., Li, X., Barker Schaaf, C., and Barnsley, M.: Global retrieval of bidirectional reflectance and albedo over land from EOS MODIS and MISR data: Theory and algorithm, J. Geophys. Res., 102, 17 143–17 161, 1997.

Warren, S. G. and Wiscombe, W. J.: A model for the spectral albedo of snow, II, Snow containing atmospheric aerosols, J. Atmos. Sci., 37, 2734–2745, 1980.

Weitkamp, C.: LIDAR: Range-Resolved Optical Remote Sensing of the Atmosphere, Springer Series in Optical Sciences, Springer, 2005.

Wiscombe, W.: Improved Mie scattering algorithms, Applied Optics, 19, 1505–1509, 1980.

Yang, P., Liou, K. N., Wyser, K., and Mitchell, D.: Parameterization of the scattering and absorption properties of individual ice crystals, J. Geophys. Res., 105, 4699–4718, 2000.

Zdunkowski, W., Trautmann, T., and Bott, A., eds.: Radiation in the Atmosphere, Cambridge U. Press, Cambridge, UK, 2007.

# Index