ARTS User Guide

edited by

Patrick Eriksson and Stefan Buehler

October 8, 2020
ARTS Version 2.0.49

The content and usage of ARTS are not only described by this document. An overview of ARTS documentation and help features are given in Section 1.2. For continuous reports on changes of the source code and this user guide, subscribe to the ARTS developers mailing list at `http://www.sat.ltu.se/arts/support/`.

We welcome gladly comments and reports on errors in the software or the document. Send then an e-mail to: `patrick.eriksson(at)chalmers.se` **or** `sbuehler(at)ltu.se`**.**

If you use data generated by ARTS in a scientific publication, then please mention this and cite the most appropriate of the ARTS publications. The relevant publications are summarised at `http://www.sat.ltu.se/arts/docs/`.

Contributing authors

Author/email	Main contribution(s)
Stefan Buehler[a] sbuehler (at) ltu.se	Editor, Sections 1, 4 and 8.
Claudia Emde[c] claudia.emde (at) dlr.de	Sections 9 and 10.
Patrick Eriksson[b] patrick.eriksson (at) chalmers.se	Editor, Sections 2, 3, 5 , 6, 7 and 12.
Oliver Lemke[a] olemke (at) core-dump.info	Latex fixes.

The present address is given for active contributors, while for others the address to the institute where the work was performed is given:
[a] Department of Computer Science, Electrical and Space Engineering, Division of Space Technology, Luleå University of Technology, Box 812, SE-98128 Kiruna, Sweden.
[b] Department of Earth and Space Science, Chalmers University of Technology, SE-41296 Gothenburg, Sweden.
[c] Institute of Environmental Physics, University of Bremen, P.O. Box 33044, D-28334 Bremen, Germany.

Contents

Chapter 1

Basics

This section introduces and describes the basic ideas underlying the ARTS program. It also presents some terminology. You should read it if you want to understand how the program works and how it can be used efficiently.

1.1 What is ARTS?

The Atmospheric Radiative Transfer Simulator, ARTS, is a software for performing simulations of atmospheric radiative transfer. ARTS is a relatively general and flexible program, where new calculation features can be easily added. Originally, the development of ARTS was initiated to deal with passive mm and sub-mm measurements. The radiation source for such measurements is emission in the atmosphere or by the Earth's surface. Thermal IR radiation is governed by the same basic physical principles and therefore this wavelength region is also well handled in ARTS now. But ARTS contains so far no dedicated methods for scattering of solar radiation and there is therefore a restriction to simulations of longwave radiation (microwave to thermal IR).

One main application of ARTS should be to perform retrievals for remote sensing data. A special feature of ARTS in this context is its high flexibility when defining observation geometry (including scanning features) and sensor characteristics. Jacobians (weighting functions) are also provided. ARTS can also be used for basic studies of atmospheric radiation, as for example in *Buehler et al.* [2006] or *John et al.* [2006].

There exist two versions of ARTS. This user guide deals with the later of the two versions [*Eriksson et al.*, 2011], here denoted as just ARTS. ARTS-1, the first version of ARTS [*Buehler et al.*, 2005], can only handle 1D atmospheres with unpolarised radiation and situations where scattering can be neglected. These restrictions have been removed in the current version. A short summary of ARTS's main features is:

The atmosphere can be 1D, 2D or 3D. That is, atmospheric variables (temperature, gas

History	
110505	Complete revision by Stefan Buehler. Also integrated text from ARTS2 article first submission.
2002-10	Written, mainly by Stefan Buehler, some parts by Patrick Eriksson.

concentrations etc.) can be assumed to only vary in the vertical dimension (1D), to have no longitude variation (2D) or vary in all three spatial dimensions (3D).

The surface and the geoid can have arbitrary shapes. That is, there are no limitations to a flat or spherical surface.

Polarisation is fully described by using the Stokes formalism.

Scattering can be considered by two modules, DOIT (Section 10) and MC (*ARTS Theory*, Section 6). The scattering calculations are for efficiency reasons confined to a region of the atmosphere denoted as the cloud box.

Observation geometry is free. That is, the forward model can be used to simulate ground-based, down-looking, limb sounding and balloon/aircraft measurements.

Sensor characteristics can be incorporated in a flexible and efficient manner.

Jacobians, the partial derivatives of simulated measurement with respect to forward model variables, can be provided for a number of variables, where analytical expressions are used as far as possible.

Details are found in later parts of the user guide. Use the table of contents and the index for navigating through the user guide.

1.2 Documentation

We know that the ARTS documentation is far from perfect. It is quite complete in some areas, but patchy in others. It also contains bugs and more serious errors. We are struggling to make it as good as possible, but it is ongoing work, and we do not have any direct funding for it. All help from users to extend or correct the documentation is highly appreciated! Having said that, the documentation that already is available for ARTS is described in the following subsections.

1.2.1 Guide documents

ARTS User Guide: This document.

ARTS Developer Guide: Guide for ARTS developers.

ARTS Theory: Describes the theoretical basis for some parts of ARTS.

1.2.2 Articles

Buehler et al. **[2005]:** General description of the old ARTS version without scattering. Many basic features are still the same, so this article is relevant also for the current ARTS version.

Eriksson et al. **[2011]:** Introduction and overview to ARTS-2.

Emde et al. [2004]**:** Describes the Discrete Ordinate Iterative method (DOIT) for handling scattering.

Davis et al. [2005]**:** Describes the Monte Carlo scattering method.

Eriksson et al. [2006]**:** Describes the calculation approach for the incorporation of sensor characteristics.

Buehler et al. [2010]**:** Describes a method to efficiently handle broadband infrared channels, that is implemented in ARTS.

Buehler et al. [2011]**:** Describes the absorption look-up table approach used inside ARTS.

1.2.3 Built-in documentation

ARTS contains built-in documentation for all functions and variables that are directly visible to the user (in ARTS terminology called workspace functions and workspace variables, and explained further below). The easiest way to access this documentation is on the web page `http://www.sat.ltu.se/arts/docserver-stable`. Alternatively, start ARTS with

```
arts -s
```

or

```
arts --docserver
```

and then point your browser to `http://localhost:9000/`.

This user guide also contains links to the built-in documentation. If you are reading the pdf file on a computer, then names of ARTS objects, such as f_grid, will be links to the corresponding entries in the built-in documentation.

1.2.4 Test and include controlfiles

ARTS calculations are governed by controlfiles (see below). The ARTS distribution already comes with a large number of controlfiles, which fall into two categories: includes and tests. They are described in more detail below, but already mentioned here as an important source of information for new users. In particular, ARTS already comes with controlfiles to simulate some well-known instruments, such as for example MHS or HIRS.

1.2.5 Source code documentation

All internal ARTS functions are documented with DOXYGEN at the source code level. This documentation is intended mostly for those who want to do ARTS development work. It can be accessed at `http://www.sat.ltu.se/arts/misc/arts-doc/doxygen/html/`.

1.2.6 Build instructions

Instructions on how to configure and compile the ARTS source code can be found in the file `README` in the top directory of the ARTS distribution.

```
Arts2 {
StringCreate(s)
StringSet(s,
   "Hello World")
Print(s)
}
```

```
arts-1.14.122
Executing Arts
{
- StringCreate
- StringSet
- Print
Hello World
}
This run took 0.03s (0.03s CPU time)
Everything seems fine. Goodbye.
```

Figure 1.1: Left: A minimal ARTS controlfile example. Right: ARTS output when running this controlfile.

1.2.7 Command line parameters

ARTS offers a number of useful command line parameters. In general, there is a short form and a long form for each parameter. The short form consists of a minus sign and a single letter, whereas the long form consists of two minus signs and a descriptive name. To get a full list of available command line parameters, type

```
arts -h
```

or

```
arts --help
```

1.3 ARTS as a scripting language

One of the main goals in the ARTS development was to make the program as flexible as possible, so that it can be used for a wide range of applications and new features can be added in a relatively simple manner. As a result, ARTS behaves like a scripting language. An ARTS controlfile contains a sequence of instructions. When ARTS is executed, the controlfile is parsed, and then the instructions are executed sequentially. Controlfile `somefile.arts` is executed by running

```
arts somefile.arts
```

A minimal ARTS controlfile example (the well-known ‘Hello World’ program) is given in Figure 1.1. In this example, the variable `s` is called a *workspace variable*. We use this name to distinguish it from the variables that appear internally in the ARTS source code.

In a similar spirit, the functions StringCreate, StringSet, and Print in the example are called *workspace methods*. We use this name to distinguish them from the functions that appear internally in the ARTS source code. For brevity, we may sometimes drop the workspace qualifier and refer to them just as methods.

ARTS consists roughly of three parts. Firstly, the ARTS core contains the controlfile parser, and the engine that executes the controlfile. This part is quite compact and constitutes only a small fraction of the total source code. Secondly, there is a large collection of workspace methods that can be used to carry out various sub-tasks (at the time of writing approximately 300). Thirdly, there is a large number of predefined workspace variables (at

```
*--------------------------------------------------------*
Workspace variable = f_grid
----------------------------------------------------------
The frequency grid for monochromatic pencil beam
calculations.

Usage: Set by the user.

Unit:  Hz
----------------------------------------------------------
Group = Vector
*--------------------------------------------------------*
```

Figure 1.2: Built-in documentation for variable f_grid, obtained by command '`arts -d f_grid`', or on page `http://www.sat.ltu.se/arts/docserver-stable/variables/f_grid`.

the time of writing more than 200). These predefined variables make it easier to set up controlfiles, since they provide hints on how the different workspace methods fit together.

ARTS has built-in documentation for all workspace methods and variables, which can be accessed as described in Section 1.2.3. In this user guide, just clicking on the name of a variable or method will take you directly to the built-in documentation for that object. Below, we will discuss workspace variables and methods in some more detail and give more examples.

1.3.1 Workspace variables

Workspace variables (such as the variable `s` in Figure 1.1) are the variables that are manipulated by the workspace methods during the execution of an ARTS controlfile. Workspace variables belong to different *groups* (Index, String, Vector, Matrix, etc.). The built-in documentation lists all groups, at the time of writing there are approximately 60 of them.

As the example in Figure 1.1 shows, workspace variables can be freely created by the user with methods like StringCreate, VectorCreate, and so on. Each group has its own create method.

However, in most cases it is not necessary to create new variables in this way, since a lot of variables are predefined in ARTS. The built-in documentation describes all predefined variables. As an example, Figure 1.2 shows the description for the variable f_grid, which stores the frequency grid and is used as input by many workspace methods, for example those that calculate absorption coefficients. The built-in documentation also lists all methods that take f_grid as input and all methods that produce f_grid as output.

1.3.2 Workspace methods

As shown in Figure 1.1, names of workspace methods in an ARTS controlfile are followed by their output and input arguments (workspace variables) in parentheses. ('`Print(s)`' prints the content of variable `s`.)

```
*--------------------------------------------------------------*
Workspace method = WriteXML
----------------------------------------------------------------
Writes a workspace variable to an XML file.

This method can write variables of any group.

If the filename is omitted, the variable is written
to <basename>.<variable_name>.xml.

Synopsis:

WriteXML( output_file_format, v, filename )

Authors: Oliver Lemke

Variables:

IN    output_file_format (String): Output file format.
GIN   v (Any): Variable to be saved.
GIN   filename (String, Default: ""): Name of the XML file.
*--------------------------------------------------------------*
```

Figure 1.3: Built-in documentation for method WriteXML, obtained by command '`arts -d WriteXML`', or on page `http://www.sat.ltu.se/arts/docserver-stable/methods/WriteXML`.

From the methods point of view, arguments can be *output*, *input*, or both, and additionally they can be either *specific* (= referring to a predefined variable) or *generic* (= not referring to a predefined variable). To illustrate this, Figure 1.3 shows the built-in documentation for method WriteXML, the most common method to write ARTS variables to a file. The list at the bottom of the documentation shows that output_file_format is a specific input argument, and that `v` and `filename` are generic input arguments.

What this means is that output_file_format already automatically exists as a variable, whereas `v` and `filename` do not. The built-in documentation provides descriptions also of these generic arguments and lists the allowed values.

The predefined variables, combined with specific method arguments, are meant to help in combining methods into meaningful calculations. Predefined variables are typically relevant for more than one method. For example, variable output_file_format can be used to change the format of all produced files at the same time. However, the use of a specific variable in the controlfile is not mandatory, so '`WriteXML( output_file_format, v, "test.xml")`', '`WriteXML( "ascii", v, "test.xml" )`', and '`WriteXML( my_format, v, "test.xml" )`' are all allowed. (But in the last example the variable `my_format` must have been defined before.)

Besides the variable names, the built-in documentation also lists the allowed variable groups (or types). In the example, the groups for workspace variable `v` are 'Any', which means that `v` can belong to any of the known groups. The group for `filename` is 'String', which means that a string is expected here. Method arguments can be a literal, as in ‘`WriteXML( "ascii", v, "test.xml" )`’, or a variable, as in ‘`WriteXML( "ascii", v, s )`’, where in the latter case variable `s` must be already defined.

The built-in documentation further states that argument `filename` has a default value (in this case the empty string). Because of this, the argument can actually be omitted, so ‘`WriteXML( "ascii", v )`’ will also work.

One additional rule has to be mentioned here. If all arguments to a method are specific, and the user wants to use all the predefined variables, then the entire argument list (including parentheses) may be omitted.

1.3.3 Agendas

Agendas are a special group of workspace variables, which allow to modify how a calculation is performed. A variable of group agenda holds a list of workspace methods. It can be executed, which means that the methods it contains are executed one after another.

Figure 1.4 gives an example, for the agenda abs_scalar_gas_agenda. Several radiative transfer methods use this agenda as input variable. When they need local absorption coefficients for a point in the atmosphere, they execute the agenda with the local pressure, temperature, and trace gas volume mixing ratio values as inputs. The agenda then provides absorption coefficients as output.

The bottom of Figure 1.4 shows two different ways how this agenda could be defined in the controlfile. In the first case a line-by-line absorption calculation is performed when the agenda is executed (every time absorption coefficients are needed). In the second case the absorption coefficients are extracted from a pre-calculated lookup table.

On invocation, an agenda executes its methods one after the other. The inputs and outputs defined for the agenda must be satisfied by the invoked workspace methods. E.g., if an agenda has abs_scalar_gas in its list of output workspace variables, at least one workspace method which generates abs_scalar_gas must be added to the agenda in the controlfile.

1.4 Include controlfiles

ARTS controlfiles can *include* other ARTS controlfiles, which is achieved by statements such as `INCLUDE "general.arts"`. This mechanism is used to predefine general default settings, settings for typical applications, and/or settings for the simulation of well-known instruments. A variety of include controlfiles are collected in directory `includes` of the ARTS distribution.

You should normally at least include the file `general.arts`, which contains general default settings. Because giving the full path for every include file is inconvenient, ARTS will look for include files in a special directory. This can be set by the command line option `-I <includepath>`, or by the environment variable `ARTS_INCLUDE_PATH`. If none of these are set, ARTS will assume the include path to point to the includes directory in the ARTS distribution.

```
*-------------------------------------------------------------------*
Workspace variable = abs_scalar_gas_agenda
---------------------------------------------------------------------

Calculation of scalar gas absorption.

This agenda calculates absorption coefficients for all gas species
as a function of the given atmospheric state for one point in the
atmosphere. The result is returned in *abs_scalar_gas*, the
atmospheric state has to be specified by *rte_pressure*,
*rte_temperature*, and *rte_vmr_list*.

A mandatory input parameter is f_index, which is used as follows:

1. f_index < 0 : Return absorption for all frequencies (in f_grid).

2. f_index >= 0 : Return absorption for the frequency indicated by
   f_index.

The methods inside this agenda may require a lot of additional
input variables, such as *f_grid*, *species*, etc.
---------------------------------------------------------------------
Group  = Agenda
Output = abs_scalar_gas
Input  = f_index, rte_pressure, rte_temperature, rte_vmr_list
*-------------------------------------------------------------------*
```

```
AgendaSet(abs_scalar_gas_agenda)
{
  abs_scalar_gasCalcLBL
}
```

```
AgendaSet(abs_scalar_gas_agenda)
{
  abs_scalar_gasExtractFromLookup
}
```

Figure 1.4: Top: Built-in documentation for variable abs_scalar_gas_agenda, obtained by command ‘`arts -d abs_scalar_gas_agenda`’, or on page `http://www.sat.ltu.se/arts/docserver-stable/agendas/abs_scalar_gas_agenda`. Bottom left: Controlfile agenda definition for line-by-line absorption calculation. Bottom right: Controlfile agenda definition to extract absorption from a pre-calculated lookup table.

1.5 Test controlfiles

The directory `tests` in the ARTS distribution contains some test and example controlfiles. You should study them to learn more about how the program works. You can run these controlfiles like this:

```
arts TestAbs.arts
```

This assumes that you are in the directory where the control file resides, and that the `arts` executable is in your path.

Alternatively, you can run a standard set of the test controlfiles by going to the `tests` directory under your build directory (*not* the one in the ARTS distribution), and say

```
make check
```

This standard set of test is run by us on every automatic build, that means every time a new ARTS version is submitted to the subversion repository. If your ARTS installation is healthy, `make check` should run through without any errors. (See file `README` in the ARTS distribution for detailed instructions on how to build ARTS.)

1.6 Verbosity levels

The command line parameter

```
arts -r
```

or

```
arts --reporting
```

can be used to set how much output ARTS produces. You can supply a three-digit integer here. Each digit can have a value between 0 and 3.

The last digit determines, how verbose ARTS is in its report file. If it is 0, the report file will be empty, if it is 3 it will be longest.

The middle digit determines, how verbose ARTS is on the screen (stdout). The meaning of the values is exactly as for the report file.

The first digit is special. It determines how much you will see of the output of agendas (other than the main program agenda). Normally, you do not want to see this output, since many agendas are called over and over again in a normal program run.

The agenda verbosity applies in addition to the screen or file verbosity. For example, if you set the reporting level to '123', you will get:

- From the main agenda: Level 1-2 outputs to the screen, and level 1-3 outputs to the report file.
- From all other agendas: Only level 1 outputs to both screen and report file.

As another example, if you set the reporting level to '120' the report file will be empty.

The default setting for ARTS (if you do not use the command line flag) is '010', i.e., only the important messages to the screen, nothing to the report file, and no sub-agenda output.

Chapter 2

Description of the atmosphere

This section discusses the model atmosphere: how it is defined, its boundaries and the variables describing the basic properties. One aspect that can cause confusion is that several vertical coordinates must be used (Section 2.1). The main vertical coordinate is pressure and atmospheric quantities are defined as a function of pressure (Section 2.3), but the effective vertical coordinate from a geometrical perspective (such as the determination of propagation paths) is the radius (Section 2.2). Pressures and radii are linked by specifying the geometrical altitudes (z_field).

2.1 Altitude coordinates

Pressure The main altitude coordinate is pressure. This is most clearly manifested by the fact that the vertical atmospheric grid consists of levels with equal pressure. The vertical grid is consistently denoted as the pressure grid and the corresponding workspace variable is p_grid. The choice of having pressure as main altitude coordinate results in that atmospheric quantities are retrieved as a function of pressure, not as a function of geometrical altitude.

Pressure altitude A basic assumption in ARTS is that atmospheric quantities (temperature, geometric altitude, species VMR etc.) vary linearly with the logarithm of the pressure. This corresponds roughly to assuming a linear variation with altitude.

Radius Geometrical altitudes are needed to determine the propagation path through the atmosphere etc. The main geometrical altitude coordinate is the distance to the centre of the coordinate system used, the radius. This is a natural consequence of using a spherical or polar coordinate system. The radius is used inside ARTS for all geometrical calculations and to store the position of the sensor (Section 3.7).

Geometrical altitude The term geometrical altitude signifies here the difference in radius between a point and the geoid (Section 5.1) along the vector to the centre of the coor-

History	
110610	Revised by Patrick Eriksson.
020315	First version by Patrick Eriksson.

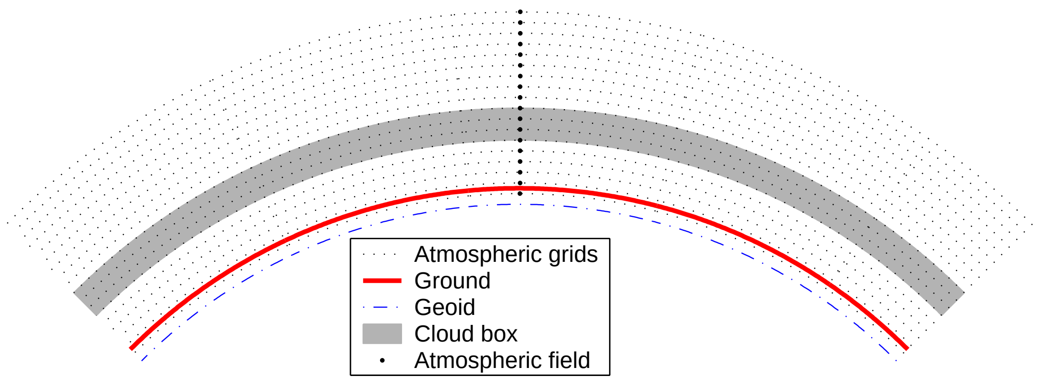

Figure 2.1: Schematic of a 1D atmosphere. The atmosphere is here spherically symmetric. This means that the radius of the geoid, the surface and all the pressure levels are constant around the globe. The fields are specified by a value for each pressure level. The extension of the cloud box is either from the surface up to a pressure level, or between two pressure levels (which is the case shown in the figure). The figure shows further that the surface must be above the lowermost pressure level.

dinate system (Equation 5.7). This is consistent with the usage of geocentric latitudes (see below). Hence, the altitude is not measured along the local zenith direction.

2.2 Atmospheric dimensionality

The structure of the modelled atmosphere can be selected to have different degree of complexity, the atmospheric dimensionality. There exist three levels for the complexity of the atmosphere, 1D, 2D and 3D, where 1D and 2D can be seen as special cases of 3D. The significance of these different atmospheric dimensionalities and the geometrical coordinate systems used are described below in this section. The atmospheric dimensionality is selected by setting the workspace variable atmosphere_dim to a value between 1 and 3. The atmospheric dimensionality is most easily set by the functions AtmosphereSet1D, AtmosphereSet2D and AtmosphereSet3D.

3D In this, the most general, case, the atmospheric fields vary in all three spatial coordinates, as in a true atmosphere (Figures 2.3 and 2.4). A spherical coordinate system is used where the dimensions are radius (r), latitude (α) and longitude (β), and a position is given as (r, α, β). With other words, the standard way to specify a geographical position is followed. However, the way to specify the radial position differs depending on the context, which is described in Section 2.1. The valid range for latitudes is $[-90^\circ, +90^\circ]$, where +90°corresponds to the North pole etc. Longitudes are counted from the Greenwich meridian with positive values towards the east. Longitudes can have values from -360°to +360°. When the difference between the last and first value of the longitude grid is $\geq 360^\circ$ then the whole globe is considered to be covered. The user must ensure that the atmospheric fields for β and $\beta + 360^\circ$ are equal. If a point of propagation path is found to be outside the range of the longitude

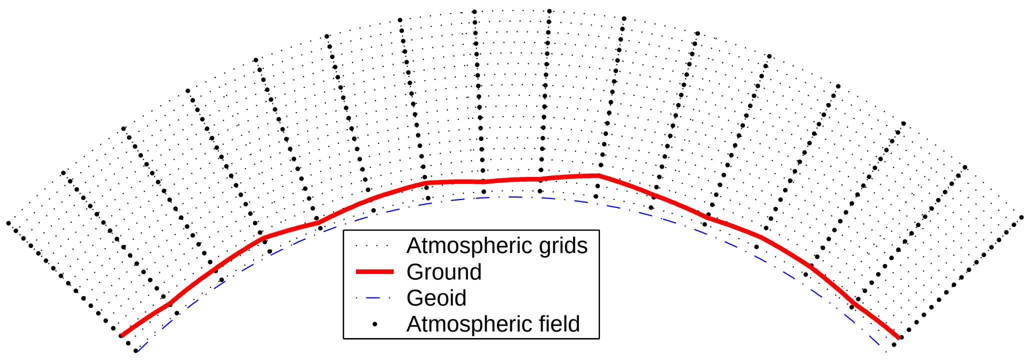

Figure 2.2: Schematic of a 2D atmosphere. The radii (for the geoid, the surface and the pressure levels) vary here linear between the latitude grid points. The atmospheric fields vary linearly along the pressure levels and the latitude grid points (that is, along the dotted lines). Inside the grid cells, the fields have a bi-linear variation. No cloud box is included in this figure.

grid, this will results in an error if not the whole globe is covered. When possible, the longitude is shifted with 360° in the relevant direction.

1D A 1D atmosphere can be described as being spherically symmetric (Figure 2.1). The term 1D is used here for simplicity and historical reasons, not because it is a true 1D case (a strictly 1D atmosphere would just extend along a line). A spherical symmetry means that atmospheric fields and the surface extend in all three dimensions, but they have no latitude and longitude variation. This means that, for example, atmospheric fields vary only as a function of altitude and the surface constitutes the surface of a sphere. The radial coordinate is accordingly sufficient when dealing with atmospheric quantities, but the angular distance between the sensor and a point along the propagation path can be of interest, for example when determining the cross-link between two satellites. This distance is tracked, basically using the 2D view (see below) with the sensor position set to be the zero point for the latitudes. A 1D atmosphere is shown in Figure 2.1.

2D In contrast to the 1D and 3D cases, a 2D atmosphere is strictly defined only inside a plane (Figure 2.2). A spherical coordinate system is accordingly not needed and a polar system, consisting of a radial and an angular coordinate, is used. The 2D case is most likely used for satellite measurements where the atmosphere is observed inside the orbit plane. The angular coordinate corresponds then to the angular distance along the satellite track, but the coordinate is for simplicity denoted as the latitude. The zero point for the 2D latitude is arbitrary. No lower and upper limit exists for the 2D latitude, and this allows that measurements from several subsequent orbits can be simulated as one unit. The atmosphere is normally treated to be undefined outside the considered plane, but some scattering calculations treat the surrounding atmosphere in an simplified manner [FIXME: Add reference to FOS, when chapyer exists.]. A 2D atmosphere is shown in Figure 2.2.

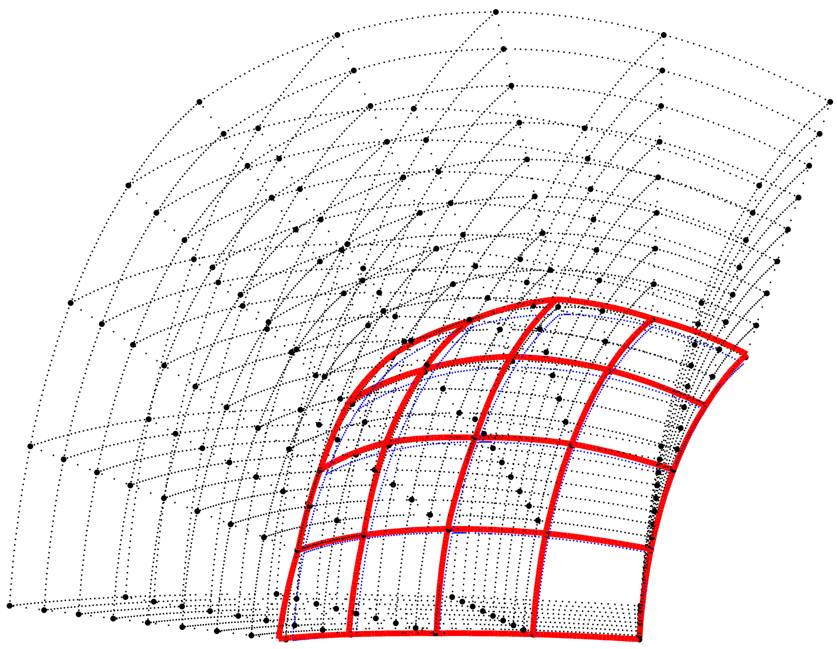

Figure 2.3: Schematic of a 3D atmosphere. Plotting symbols as in Figure 2.2. Radii and fields are here defined to vary linearly along the latitude and longitude grid points. This means that the radius of a pressure level has a bi-linear variation inside the area limited by two latitude and longitude grid values, while the atmospheric fields have a tri-linear variation inside the grid cells.

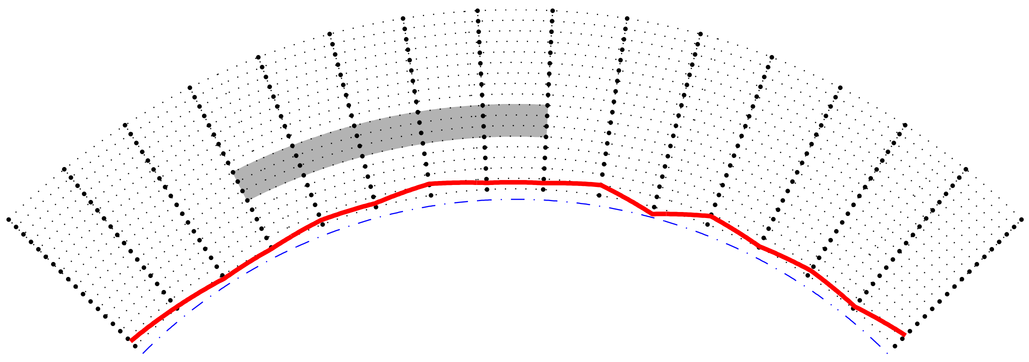

Figure 2.4: A latitudinal, or longitudinal, cross section of a 3D atmosphere. Plotting symbols as in Figure 2.1. Radii and fields inside the cross section match the definitions for 2D. The vertical extension of the cloud box is defined identical for 1D and 3D. The horizontal extension of the cloud box is between two latitude and longitude grid positions, where only one of the dimensions is visible in this figure.

2.3 Atmospheric grids and fields

As mentioned above, the vertical grid of the atmosphere consists of a set of layers with equal pressure, the pressure grid (p_grid). This grid must of course always be specified. The upper end of the pressure grid gives the practical upper limit of the atmosphere as vacuum is assumed above. With other words, no absorption and refraction take place above the uppermost pressure level.

A latitude grid (lat_grid) must be specified for 2D and 3D. For 2D, the latitudes shall be treated as the angular distance along the orbit track, as described above in Section 2.2. The latitude angle is throughout calculated for the vector going from the centre of the coordinate system to the point of concern. Hence, the latitudes here correspond to the definition of the geocentric latitude, and not geodetic latitudes (see Section 5.1.1). This is in accordance to the definition of geometric altitudes above. For 3D, a longitude grid (lon_grid) must also be specified. Valid ranges for latitude and longitude values are given in Section 2.2.

The atmosphere is treated to be undefined outside the latitude and longitude ranges covered by the grids, if not the whole globe is covered. This results in that a propagation path is not allowed to cross a latitude or longitude end face of the atmosphere, if such exists, it can only enter or leave the atmosphere through the top of the atmosphere (the uppermost pressure level). See further Section 3.3. The volume (or area for 2D) covered by the grids is denoted as the model atmosphere.

If the longitude and latitude grids are not used for the selected atmospheric dimensionality, then the longitude grid (for 1D and 2D) and the latitude grid (for 1D) can normally be set to be empty, but some methods require a scalar latitude and longitude.

The basic atmospheric quantities are represented by their values at each crossing of the involved grids (indicated by thick dots in Figures 2.1–2.4), or for 1D at each pressure level (thick dots in Figure 2.1). This representation is denoted as the field of the quantity. The field must, at least, be specified for the geometric altitude of the pressure levels (z_field), the temperature (t_field) and considered atmospheric species (vmr_field). The fields are assumed to be piece-wise linear functions vertically (with pressure altitude as the vertical coordinate, Section 2.1), and along the latitude and longitude edges of 2D and 3D grid boxes. For points inside 2D and 3D grid boxes, multidimensional linear interpolation is applied (that is, bi-linear interpolation for 2D etc.). Note especially that this is also valid for the field of geometrical altitudes (z_field). Fields are rank-3 tensors. For example, the temperature field is $T = T(P, \alpha, \beta)$. That means each field is like a book, with one page for each pressure grid point, one row for each latitude grid point, and one column for each longitude grid point. In the 1D case there is just one row and one column on each page. The representation of atmospheric fields, and other quantities, is discussed further in Sec. 11.2, where the concept of basis functions is introduced. In short, the basis functions give the mapping from the set of discrete values to the continous representation of the quantity.

2.4 Hydrostatic equilibrium

There is no general demand that the model atmosphere fulfils hydrostatic equilibrium. That is, t_field and z_field can be specified independently of each other. On the other hand, ARTS provides means for ensuring that a model atmosphere matches hydrostatic equilibrium by the method z_fieldFromHSE. This method is presently valid only for the planet Earth, but considers that gravitation varies with latitude and altitude.

Hydrostatic equilibrium gives only constrain for the distance between the pressure levels, not for the absolute geometrical altitudes. For this reason, a "reference point" must be introduced. This is done by setting the pressure of this point by p_hse (common for all latitude and longitudes). The geometrical altitudes matching p_hse are taken from the original values in z_field.

2.5 The geoid and surface

Geometrical altitudes are specified as the vertical distance to a reference geoid. The geoid is defined by giving the radius for each latitude and longitude grid point. Accordingly, this is a matrix, named as r_geoid. For 1D atmosphere, this matrix has the size $[1, 1]$. See further Section 5.1.

The lower boundary of the atmosphere is denoted as the surface. The surface is specified by its geometrical altitude on the latitude and longitude grids. The workspace variable holding these data is called z_surface, and is a matrix of same size as r_geoid.

It is not allowed that there is an altitude gap between the surface and the lowermost pressure level. That is, the surface pressure must be smaller than the pressure of the lowermost vertical grid level. On the other hand, it is not necessary to match the surface and the first pressure level, the pressure grid can extend below the surface level.

2.6 The cloud box

In order to save computational time, calculations involving scattering and polarised gas absorption are limited to a special atmospheric domain. This atmospheric region is denoted as the cloud box, reflecting that it was introduced primarily to simplify scattering calculations. The cloud box is discussed here as it acts as an additional atmospheric limit when calculating propagation paths (see Section 3.5).

The cloud box is defined to be rectangular in the used coordinate system, with limits exactly at points of the involved grids. This means, for example, that the vertical limits of the cloud box are two pressure levels. For 3D, the horizontal extension of the cloud box is between two points of the latitude grid and likewise in the longitude direction (Figure 2.4). The latitude and longitude limits for the cloud box cannot be placed at the end points of the corresponding grid as it must be possible to calculate the incoming intensity field. The cloud box is activated by setting the variable cloudbox_on to 1. The limits of the cloud box are stored in cloudbox_limits. It is recommended to use the method CloudboxOff when no scattering calculations shall be performed. This method assigns dummy values to all workspace variables not needed when scattering is neglected.

When the radiation entering the cloud box is calculated this is done with the cloud box turned off. This to avoid to end up in the situation that the radiation entering the cloud box depends on the radiation coming out from the cloud box. **It is the task of the user to define the cloud box in such way that the link between the outgoing and ingoing radiation fields of the cloud box can be neglected**. The main point to consider here is radiation reflected by the surface. To be formally correct there should never be a gap between the surface and the cloud box. This is the case as radiation leaving the cloud box can then be reflected back into the cloud box by the surface. If it is considered that the surface is a scattering object it is clear that the surface should in general be part of the cloud box. However, for many cases it can be accepted to have a gap between the surface and the cloud box, with the gain that the cloud box can be made smaller. Such a case is when the surface is treated to act as blackbody, the surface is then not reflecting any radiation. Reflections from the surface can also be neglected if the zenith optical thickness of the atmosphere between the surface and cloud box is sufficiently high.

Chapter 3

Overview of clear-sky radiative transfer calculations

This section gives an overview of the variables and the approach used to handle the actual radiative transfer calculations. This includes an overview of how effects caused by the sensor and surface are incorporated. The default assumption is that there is no particle scattering and the (atmospheric) absorption/emission is unpolarised. This is for simplicity denoted as clear-sky calculations. Cases involving scattering or polarised absorption are handled by the "cloud box" (Sec. 2.6). In short, the more demanding calculations are restricted to a smaller domain of the model atmosphere, and the radiative transfer in this domain is treated by dedicated workspace methods.

The section deals with general aspects of radiative transfer and the algorithms applied outside the cloud box. Even though the atmospheric absorption and emission outside the cloud box are unpolarised, the expressions to apply must allow that polarisation signals from the surface and the cloud box are correctly propagated to the sensor.

3.1 Stokes dimensionality

To full polarisation state of radiation can be described by the Stokes vector. The vector can be defined in different ways, but it has always four elements. The Stokes vector, $\mathbf{I}$, is here written as

$$\mathbf{I} = \begin{bmatrix} I \\ Q \\ U \\ V \end{bmatrix}, \tag{3.1}$$

where the first component (I) is the full intensity of the radiation, the second component (Q) is the difference between vertical and horizontal polarisation, the third component (U)

History

110611	Extended and general revision (Patrick Eriksson).
050613	First complete version by Patrick Eriksson.

is the difference for $\pm 45°$ polarisation and the last component (V) is the difference between left and right circular polarisation. That is:

$$\begin{aligned} I &= I_v + I_h = I_{+45°} + I_{-45°} = I_{lhc} + I_{rhc}, \quad (3.2)\\ Q &= I_v - I_h, \quad (3.3)\\ U &= I_{+45°} - I_{-45°}, \quad (3.4)\\ V &= I_{lhc} - I_{rhc}, \quad (3.5)\end{aligned}$$

where I_v, I_h, $I_{+45°}$, and $I_{-45°}$ are the intensity of the component linearly polarised at the vertical, horizontal, +45° and -45° direction, respectively, and I_{rhc}, and I_{lhc} are the intensity for the right- and left-hand circular components. Further details on polarisation and the definition of the Stokes vector are found in *ARTS Theory*, Section 4.

ARTS is a fully polarised forward model, but can be run with a smaller number of Stokes components. The selection is made with the workspace variable stokes_dim. For example, gaseous absorption and emission are in general unpolarised, and if not scattering has to be considered it is sufficient to only include the first Stokes components in the simulations. To include higher order Stokes components results only, in this case, in slower calculations. The general case is here denoted as vector radiative transfer, while scalar radiative transfer refers to the case when only the first Stokes component is considered.

3.2 Overall calculation procedure

The structure of the part handling complete radiative transfer calculations is fixed, where the main workspace method is denoted as yCalc. That is, most ARTS control files include a call of yCalc and this section outlines this method and the associated main variables.

The calculation approach fits with the formalism presented in Sections 1.1-1.2 of *ARTS Theory*, where the separation between atmospheric radiative transfer and inclusion of sensor effects shall be noted especially, and a similar nomenclature is used here:

$\mathbf{y}$: Complete measurement vector. In addition to atmospheric radiative transfer, the vector can include effects by sensor characteristics and data reduction operations. The corresponding workspace variable is y.

$\mathbf{i}_b$: Monochromatic pencil beam data for a measurement block. The definition of a measurement block is found in Section 3.7.4. This vector is only affected by atmospheric radiative transfer. Only used internally and there is no corresponding workspace variable.

$\mathbf{i}_y$: Monochromatic data for a single (pencil beam) line-of-sight. $\mathbf{i}_b$ consists of one or several $\mathbf{i}_y$ appended. The corresponding workspace variable is iy.

$\mathbf{H}_b$: The complete sensor response matrix, for a measurement block. Can include data reduction. The corresponding workspace variable is sensor_response.

The yCalc method is outlined in Algorithm 1. For further details of each calculation step, see the indicated equation or section. In summary, yCalc appends data from different pencil beam calculations and applies the sensor response matrix ($\mathbf{H}_b$). The actual radiative transfer calculations are not part of yCalc.

Algorithm 1 Outline of the overall clear sky radiative transfer calculations (yCalc).

```
allocate memory for the matrix $\mathbf{y}$ (Equation 3.16)
allocate memory for the matrix $\mathbf{i}_b$ (Equation 3.15)
for all sensor positions do
  for all pencil beam directions of the block do (Section 3.7.4)
    call iy_clearsky_agenda, giving $\mathbf{i}_y$ (Algorithm 2)
    unit conversion of $\mathbf{i}_y$ following y_unit (Section 3.6)
    copy $\mathbf{i}_y$ to correct part of $\mathbf{i}_b$
  end for
  put the product $\mathbf{H}_b\mathbf{i}_b$ in correct part of $\mathbf{y}$
end for
```

Algorithm 2 The main operations for methods to be part of iy_clearsky_agenda. The same applies to methods for iy_clearsky_basic_agenda.

```
determine the propagation path by ppath_calc (Section 3.3)
determine the radiation at the start of the propagation path (Section 3.4)
perform radiative transfer along the propagation path (Section 3.5)
```

Atmospheric radiative transfer is solved for each pencil beam direction (line-of-sight) separately. It is the task of iy_clearsky_agenda (Algorithm 2) to perform a single such clear sky radiative transfer calculation. This agenda, in its turn, makes us of other agendas, such as ppath_step_agenda. All methods developed for iy_clearsky_agenda adapt automatically to the value of stokes_dim.

That is, yCalc is a common method, independent of the details of the radiative transfer. For example, yCalc is used both if emission measurements or pure transmission data are simulated, that choice is made inside iy_clearsky_agenda (see further Section 3.5).

The three following sections describes the main calculation steps of iy_clearsky_agenda, in the order they are executed.

3.3 Propagation paths

A pencil beam path through the atmosphere to reach a position along a specific line-of-sight is denoted as the propagation path. Propagation paths are described by a set of points on the path, and the distance along the path between the points. These quantities, and a number of auxiliary variables, are stored together in a structure described in Section 6.3. The path points are primarily placed at the crossings of the path with the atmospheric grids (p_grid, lat_grid and lon_grid). A path point is also placed at the sensor if it is placed inside the atmosphere. Points of surface reflections and tangent points are also included if such exist. More points can also be added to the propagation path, for example, by setting an upper limit for the distance along the path between the points. This is achieved by the variable ppath_lmax, see further Sections 3.8 and 6.1.

The propagation paths are determined basically by starting at the sensor and following the path backwards by some ray tracing technique. If the sensor is placed above the model atmosphere, geometrical calculations are used (as there is no refraction in space) to find the crossing between the path and the top of the atmosphere where the ray tracing then

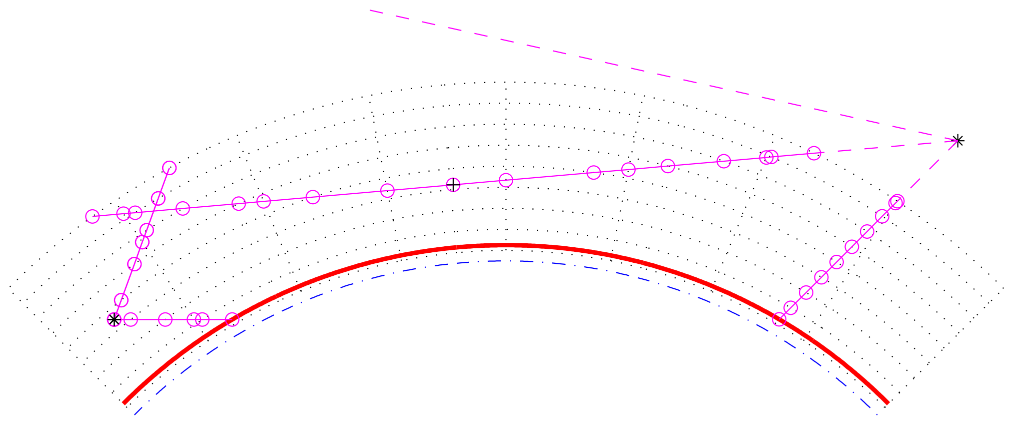

Figure 3.1: Examples on allowed propagation paths for a 2D atmosphere. The atmosphere is plotted as in Figure 2.2 beside that the points for the atmospheric fields are not emphasised. The position of the sensor is indicated by an asterisk ($*$), the points defining the paths are plotted as circles ($\circ$), joined by a solid line. The part of the path outside the atmosphere, not included in the path structure, is shown by a dashed line. Path points corresponding to a tangent point are marked by an extra plus sign ($\oplus$). The shown paths include the minimum set of definition points. There exists also the possibility to add points inside the grid cells, for example, to ensure that the distance between the path points does not exceed a specified limit.

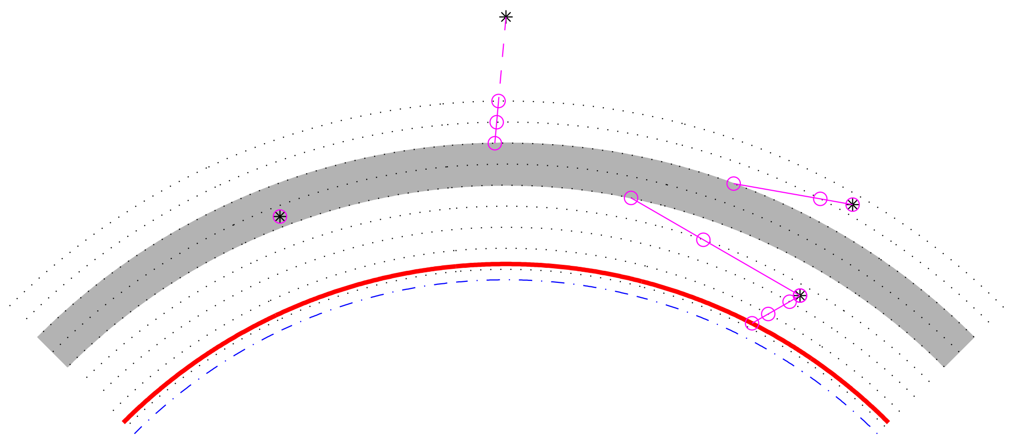

Figure 3.2: Examples on allowed propagation paths for a 1D atmosphere with an activated cloud box. Plotting symbols as in Figure 3.1. When the sensor is placed inside the cloud box, the path is defined with a single point, to know for which position and line-of-sight the intensity field of the cloud box shall be interpolated.

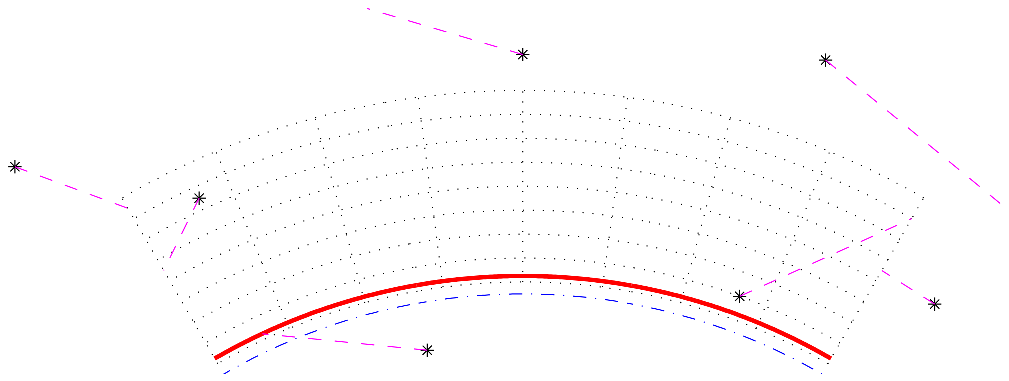

Figure 3.3: Examples on *not* allowed propagation paths for a 2D atmosphere. The constraints for allowed paths are discussed in the text.

starts. Paths are tracked backwards until the top of the atmosphere is reached, or there is an intersection with the cloud box or the surface. The propagation path (or paths) before a surface reflection is calculated when determining the up-welling radiation from the surface (Section 3.4.2). Example on propagation paths are shown in Figures 3.1 and 3.2.

Not all propagation paths are allowed for 2D and 3D. The paths can only enter and leave the model atmosphere at the top of the atmosphere, as the atmospheric fields are treated to be undefined outside the covered latitude and longitude ranges (Figure 3.3). In addition, if the sensor is placed outside the model atmosphere, the line-of-sight zenith angle must be $\geq 90°$, and the tangent point position of the propagation paths must be inside the end points of the latitude and longitude grids, but can be above the top of the atmosphere. Hence, it is allowed that the propagation path is totally outside the atmosphere, as long as the viewing direction is downward and the lowest point of the path, the tangent point, is inside the latitude and longitude limits of the model atmosphere.

Propagation paths can be calculated separately by the method ppathCalc. However, for standard calculations the propagation paths are calculated internally by yCalc. The calculation of the path from one crossing of the grids to next crossing is defined by ppath_step_agenda. Depending on which function that is selected for ppath_step_agenda, refraction will be considered etc. Available workspace methods are presented in Section 6.1.

3.4 The radiative background

3.4.1 Overview

The radiative intensity at the starting point of the path, and in the direction of the line-of-sight at that point, is denoted as the radiative background. Four possible radiative backgrounds exist:

Space When the propagation path starts at the top of the atmosphere, space is the radiative background. The normal case should be to set the radiation at the top of the atmosphere to be cosmic background radiation. An exception is when the sensor is directed towards the sun. The radiative background at the top of the atmosphere is determined by iy_space_agenda. If a propagation path is totally outside the model atmosphere, the observed monochromatic pencil beam intensity ($\mathbf{i}_y$ in Algorithm 1) equals the output of iy_space_agenda.

The surface The sum of surface emission and radiation reflected by the surface is the radiative background when the propagation path intersects with the surface. The calculation of the up-welling radiation from the surface is treated by a dedicated section below (Section 3.4.2.)

Surface of cloud box For cases when the propagation path enters the cloud box the radiative background is the intensities leaving the cloud box. This radiation is obtained by iy_cloudbox_agenda.

Interior of cloud box If the sensor is situated inside the cloud box, there is basically no propagation path. The radiative background, and also the final spectrum, equals the internal intensity field of the cloud box at the position of the sensor, in the direction of the sensor line-of-sight. This case is also handled by iy_cloudbox_agenda.

It should be noted that except for the first case above, the determination of the radiative background involves further radiative transfer calculations. For example, in the case of surface reflection, the down-welling radiation is determined by a new call of iy_clearsky_agenda and the radiative background for that calculation is then space or the cloud box. The intensity field entering the cloud box is calculated by calls of iy_clearsky_basic_agenda (with cloud box deactivated) and the radiative background is then space or the surface. This results in that space is normally the ultimate radiative background for the calculations. The exception is for propagation paths that intersects with the surface, and the surface is treated to act as a blackbody. For such cases, the propagation path effectively starts at the surface.

3.4.2 Surface scattering and emission

If there is an interception of the propagation path by the surface, emission and scattering by the surface must be considered. The overall treatment of these effects is fixed. The upwelling radiation from the surface can be written as (Figure 3.4)

$$\mathbf{i}_s^u = \mathbf{i}_e + \sum_l \mathbf{R}_l \mathbf{i}_l^d \tag{3.6}$$

where $\mathbf{i}$ is the Stokes vector for one frequency, $\mathbf{i}_s^u$ is the total upward travelling intensity from the surface along the propagation path, $\mathbf{i}_e$ is the emission from the surface, $\mathbf{i}_l^d$ is the downward travelling intensity reaching the surface along direction l, and $\mathbf{R}_l$ is the reflection coefficient matrix from direction l to the present propagation path. The emission from the surface $(\mathbf{i}_e)$ is stored in surface_emission, the directions l for which downward travelling intensities are given by surface_los, and the reflection coefficients $(\mathbf{R})$ are stored in surface_rmatrix. These workspace variables are handled by surface_prop_agenda. Surface reflections and emission are discussed further in Section 5.3.

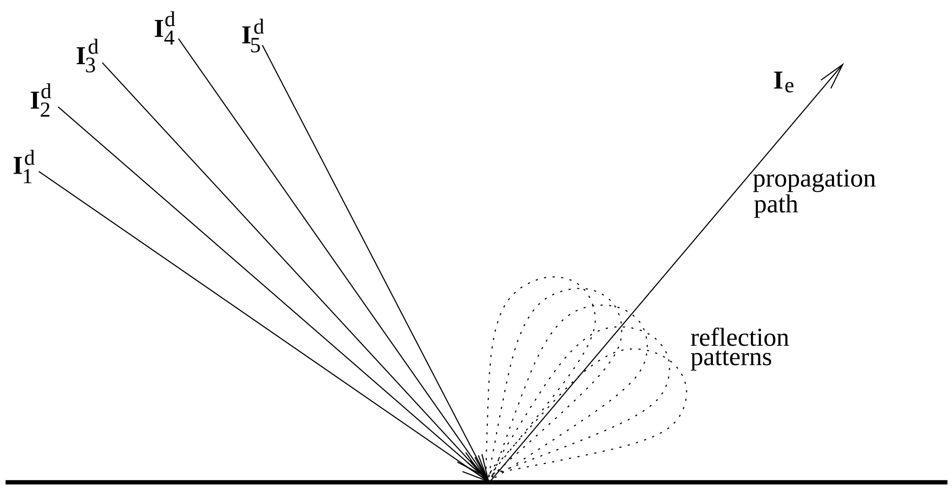

Figure 3.4: Schematic of Equation 3.6.

3.5 Basic radiative transfer variables and expressions

This section describes how the core radiative transfer equation is solved practically in ARTS. Focus is put on emission measurements as ARTS is intended primarily for such observations. However, simulations of transmission measurements are also possible.

Beside the actual radiances, iy_clearsky_agenda can provide weighting functions and auxiliary data. These later variables are not supported by all parts of ARTS, and for efficiency reasons there exists a simpler version of the agenda. This version is denoted as iy_clearsky_basic_agenda and returns only radiances.

3.5.1 Cases with emission

The complete vector radiative transfer equation, including scattering, is given by Equation 5.35 in *ARTS Theory*. If scattering can be neglected, the equation can be written as

$$\frac{\mathrm{d}\mathbf{I}}{\mathrm{d}s} = -\mathbf{K}\mathbf{I} + \mathbf{a}B, \tag{3.7}$$

where $\mathbf{I}$ is the intensity vector (the Stokes vector), s is the distance along the propagation path, $\mathbf{K}$ is the extinction matrix, $\mathbf{a}$ is the absorption vector and B is the source function (a scalar). If local thermodynamic equilibrium applies, B equals the Planck function describing blackbody radiation. See further *ARTS Theory*, Section 5. For “clear-sky conditions” the matrix $\mathbf{K}$ is diagonal, with all diagonal elements equal, and only the first of the elements of $\mathbf{a}$ is non-zero.

The radiative transfer equation above can be solved in many ways, and with different level of refinement. The standard approach in ARTS is to solve the radiative transfer from one point of the propagation path to next. For the first Stokes element the following expression is applied (compare *ARTS Theory*, Equation 5.50)

$$I_{i+1} = I_i e^{-\tau_i} + \bar{B}_i(1 - e^{-\tau_i}), \tag{3.8}$$

with

$$\begin{aligned}\bar{B}_i &= (B(T_i) + B(T_{i+1}))/2, & (3.9)\\ \tau_i &= \Delta s_i (k_i + k_{i+1})/2, & (3.10)\end{aligned}$$

where I_i, T_i and k_i are the radiance, temperature and absorption coefficient, respectively, at point i of the propagation path, and Δs_i is the distance along the path between point i and $i+1$. That is, $\bar{B}_i$ is a Planck function average of the path step, and the absorption is assumed to vary linearly between the two points. The start value of I is governed by the radiative background (Section 3.4).

As mentioned above, the emission is unpolarised for the conditions assumed here, and the emission term vanishes for higher Stokes elements. Accordingly, the expression for the second Stokes component is

$$Q_{i+1}(\nu) = Q_i(\nu) e^{-\tau_i}. \qquad (3.11)$$

The third and forth Stokes component are handled likewise.

An alternative way to perform the calculations for the first Stokes element would be

$$I = \sum_i \mathcal{T}_{i+1} \bar{B}_i (1 - e^{-\tau_i}), \qquad (3.12)$$

where I is the final intensity and $\mathcal{T}_i$ is the transmission between the sensor and point i. This calculation approach is not used as it fits poorer with the calculation of weighting functions (Section 12). However, the calculation of weighting functions is simplified if $\mathcal{T}_i$ is at hand, and this quantity is also tracked by iyEmissionStandardClearsky. The value of $\mathcal{T}$ at the radiative background can also be outputted as auxiliary data (iy_aux).

The expressions above are implemented in the workspace method iyEmissionStandardClearsky, intended to be part of iy_clearsky_agenda. For inclusion in iy_clearsky_basic_agenda, there is a version denoted as iyEmissionStandardClearskyBasic. This version is not calculating $\mathcal{T}_i$, or provides any auxiliary data.

The term B is set by emission_agenda. The setting of this agenda basically determines the unit of the final outcome of Eq. 3.8, see further Sec. 3.6. For radiance calculations, the standard workspace method to use inside emission_agenda is emissionPlanck. The Planck function is in this method, and in ARTS generally, defined as

$$B(T) = \frac{2h\nu^3}{c^2(exp(h\nu/k_bT) - 1)} \qquad (3.13)$$

where h is the Planck constant, c the speed of light and k_b the Boltzmann constant. This expression gives the total power, per unit frequency per unit area per solid angle. (The Planck function can also be defined as a function of wavelength.)

As long as cosmic background radiation is the only type of non-telluric radiation that has to be considered, the standard method for inclusion in iy_space_agenda is MatrixCBR (together with some calls of Ignore). Please, note that emission_agenda and iy_space_agenda must be defined in a consistent manner, that they use the same unit for B.

3.5.2 Pure transmission calculations

If only the attenuation of a signal is of concern (i.e. atmospheric and surface emissions are neglected), Equation 3.11 can be applied for all Stokes elements. The initiation of the Stokes vector by the radiative background must also be adopted. The standard choice for iy_space_agenda should here be MatrixUnitIntensity (instead of MatrixCBR). Otherwise the calculations are performed exactly as for cases with emission (emission_agenda can be left undefined).

Calculations of pure transmission type are performed by the method denoted as iyBeerLambertStandardClearsky. This method is complemented by iyBeerLambertStandardCloudbox, to be applied inside the cloud box.

3.6 Output unit

There is no fixed unit for calculated spectra (y), it depends on the calculation set-up. For example, if emission is considered, or if just transmissions are calculated.

The primary unit for emission data (radiances) is [W/(Hz·m^2·sr)], The emission intensity corresponds directly with the definition of the Planck function (Eq. 3.13). This radiance can be converted to other emission units by the workspace variable y_unit. For comments on the practical aspects, see the built-in documentation of y_unit. Otherwise, considerations and expressions for the unit conversion are mainly discussed in ARTS-2 journal paper [*Eriksson et al.*, 2011, Sec. 5.7].

3.7 Compulsory sensor and data reduction variables

The instrument that detects the simulated radiation is denoted as the sensor. The forward model is constructed in such way that a sensor must exist. For cases when only monochromatic pencil beam radiation is of interest, the positions and directions for which the radiation shall be calculated are given by specifying an imaginary sensor with infinite frequency and angular resolution. The workspace variables for the sensor that always must be specified are sensor_response, sensor_pos, sensor_los, antenna_dim, mblock_za_grid and mblock_aa_grid. These variables are presented separately below.

3.7.1 Sensor position

The observation positions of the sensor are stored in sensor_pos. This is a matrix where each row corresponds to a sensor position. The number of columns in the matrix equals the atmospheric dimensionality (1 column for 1D etc.). The columns of the matrix (from first to last) are radius, latitude and longitude. Accordingly, row i of sensor_pos for a 3D case is (r_i, α_i, β_i). The sensor position can be set to any value, but the resulting propagation paths (also dependent on sensor_los) must be valid with respect to the model atmosphere (see Section 3.3). An obviously incorrect choice is to place the senor below the surface altitude. If the sensor is placed inside the model atmosphere, any sensor line-of-sight is allowed, this including the cases that the sensor is placed on the surface looking down, and that the sensor is placed inside the cloud box.

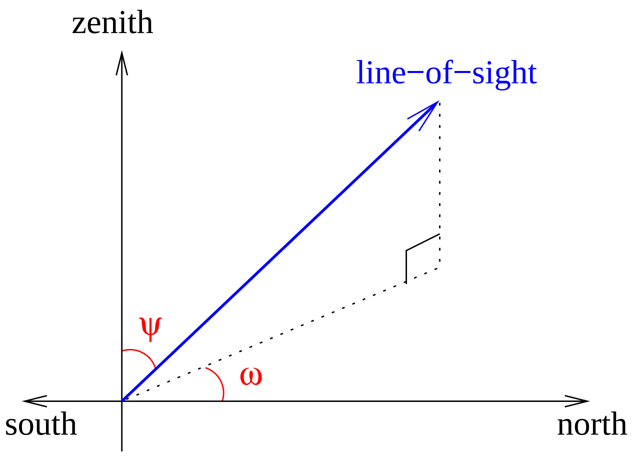

Figure 3.5: Definition of zenith angle, ψ, and azimuth angle, ω, for a line-of-sight. The figure shows a line-of-sight with a negative azimuth angle.

The fact that the sensor position can be given any value implies that the radius must be used in sensor_pos, in contrast to z_surface and z_field where the altitude above the geoid is applied. This is the case as, for 2D and 3D, the sensor can be placed outside the covered latitude and longitude ranges, thus outside the defined geoid, and the geometrical altitude is undefined.

The sensor is treated to be motionless when calculating the spectrum, or spectra, for each given observation position. One or several spectra can be calculated for each position as described in Section 3.7.4.

3.7.2 Line-of-sight

The viewing direction of the sensor, the line-of-sight, is described by two angles, the zenith angle (ψ) and the azimuth angle (ω). The zenith angle exists for all atmospheric dimensionalities, while the azimuth angle is defined only for 3D. The term line-of-sight is not only used in connection with the sensor, it is also used to describe the local propagation direction along the path taken by the observed radiation (Section 3.3). The zenith and azimuthal angles are defined in an identical way in both of these contexts (sensor pointing direction; local propagation direction). This is expected as the position of the sensor is the end point of the propagation path. The sensor line-of-sight is the direction the antenna is pointed to receive the radiation. The line-of-sight for propagation paths is defined likewise, it is the direction in which a hypothetical sensor must be placed to receive the radiation along the propagation path at the point of interest. This means that the line-of-sight and the photons are going in opposite directions. As a true sensor has a finite spatial resolution (described by the antenna pattern), theoretically there is an infinite number of line-of-sights associated with the sensor, but in the forward model, spectra are only calculated for a discrete set of directions. If a sensor line-of-sight is mentioned without any comments, it refers to the direction in which the centre of the antenna pattern is directed.

The zenith angle, ψ, is simply the angle between the line-of-sight and the zenith direction (Figure 3.5). The valid range for 1D and 3D cases is $[0, 180°]$. In the case of 2D, zenith angles down to -180° are also allowed, where the distinction is that positive angles mean a viewing direction towards higher latitudes, and negative angles mean a viewing direction towards lower latitudes. It should be mentioned that the zenith and nadir directions are here defined to be along the line passing the centre of the coordinate system and the point of concern (Section 5.1.1). A nadir observation, $\psi = 180°$, is thus a measurement towards the centre of the coordinate system.

The azimuth angle, ω, is given with respect to the meridian plane. That is, the plane going through the north and south poles of the coordinate system ($\alpha = \pm 90°$) and the sensor. The valid range is $[-180°, 180°]$ where angles are counted clockwise; 0°means that the viewing or propagation direction is north-wise and +90° means that the direction of concern goes eastward. This definition does not work for position on the poles. To cover these special cases, the definition is extended to say that for positions on the poles the azimuth angle equals the longitude along the viewing direction. For example, if standing on any of the poles and the viewing direction is towards Greenwich, the azimuth angle is 0°.

The sensor line-of-sights are stored in sensor_los. This workspace variable is a matrix, where the first column holds zenith angles and the second column is azimuth angles. For 1D and 2D there is only one column in the matrix, while for 3D a row i of the matrix is (ψ_i, ω_i). The number of rows for sensor_los must be the same as for sensor_pos.

3.7.3 Sensor characteristics and data reduction

The term "sensor characteristics" is used here as a comprehensive term for the response of all sensor parts that affect how the field of monochromatic pencil beam intensities are translated to the recorded spectrum. For example, the antenna pattern, the side-band filtering and response of the spectrometer channels are normally the most important characteristics for a microwave heterodyne radiometer. Any processing of the spectral data that takes place before the retrieval is denoted as data reduction. The most common processing is to represent the original spectra with a smaller set of values, that is, a reduction of the data size. The most common data reduction techniques is binning and Hotelling transformation by an eigenvector expansion.

In ARTS, the influence of sensor characteristics and data reduction is incorporated by transfer matrices. The application of these transfer matrices assumes that each step is a linear operation, which should be the case for the response of the parts of a well designed instrument. Non-linear data reduction could be handled by special workspace methods.

The sensor and data reduction are described as a series of units, each having its own transfer matrix. There is only one compulsory transfer matrix and it is sensor_response. There are several workspace variables associated with this transfer matrix where antenna_dim, mblock_za_grid and mblock_aa_grid are the compulsory ones.

The variable antenna_dim gives the dimensionality of the antenna pattern, where the options are 1 and 2, standing for 1D and 2D, respectively. A 1D antenna dimensionality means that the azimuth extension of the antenna pattern is neglected, there is only a zenith angle variation of the response. A 2D antenna pattern is converted to a 1D pattern by integrating the azimuth response for each zenith angle. For cases with 1D antenna patterns, mblock_aa_grid must be set to be an empty vector.

For each sensor position, a number of monochromatic pencil beam spectra are calculated. The monochromatic frequencies are given by f_grid. The pencil beam directions are obtained by summing the sensor line-of-sight angles (sensor_los) for the position and the values of mblock_za_grid and mblock_aa_grid. For example, pencil beam zenith angle i is calculated as

$$\psi_i = \psi_0 + \Delta\psi_i \tag{3.14}$$

where ψ_0 is the sensor line-of-sight for the position of concern and $\Delta\psi_i$ is value i of mblock_za_grid. With other words, mblock_za_grid and mblock_aa_grid give the grid (relative to the sensor line-of-sight) for the calculation of the intensity field that will be weighted with the antenna response.

3.7.4 Measurement sequences and blocks

The series of observations modelled by the simulations is denoted as the measurement sequence. That is, a measurement sequence covers all spectra recorded at all considered sensor positions. A measurement sequence consists of one or several measurement blocks. The observations inside the various blocks differ only with an off-set of the line-of-sight, all other factors should be common for all blocks. A block can be treated as a measurement cycle that is repeated, an integer number of times, to form the measurement sequence. The measurement blocks correspond normally to each unique sensor position of the sequence.

A measurement block covers one or several recorded spectra, depending on the measurement conditions and the atmospheric dimensionality. A block can consist of several spectra when there is no effective motion of the sensor with respect to the atmospheric fields. It should be noted that for 1D cases, a motion along a constant radius has no influence on the simulated spectra as the same atmospheric fields are seen for a given viewing direction. It is favourable, if possible, to handle all spectra as a single block, instead of using a block for each sensor position. This is the case as the antenna patterns for the different line-of-sights are normally overlapping and a pencil beam spectrum can be used in connection with several measurement spectra to estimate the intensity field. If a measurement sequence is divided into several blocks even if a single block would be sufficient, pencil beam spectra for basically identical propagation paths can be calculated several times, which of course will increase the computational time. To summarise, for cases when the sensor is not in motion, or with a 1D atmosphere and a sensor not moving vertically, the aim should be to use a single block for the measurement sequence.

If not a single block is used, the standard option should be that the blocks cover one spectrum each. There could exist reasons to select an intermediate solution, to let the extent of the blocks be several spectra (but not the full measurement sequence). This could be the case when the atmospheric dimensionality is 2D or 3D, and the sensor is moving but the movement during some subsequent spectra can be neglected. If this can be done must be judged by comparing the movement of the sensor during the extent of the considered block size and the spatial resolution, in the direction of the movement, that is hoped to be achieved. If this intermediate solution shall be an option, the difference in zenith and azimuth angles between the spectra must be the same for all blocks, otherwise sensor_response cannot be applied for all blocks as done below in Equation 3.16.

For each block, pencil beam spectra are calculated for the line-of-sights obtained when summing sensor_los and mblock_za_grid (and possibly mblock_aa_grid), as described in Section 3.7.3. The pencil beam spectra for each line-of-sight are appended vertically to form a common vector, $\mathbf{i}_b$. Values are put in following the order in f_grid. Hence, the frequencies for this vector are

$$\mathbf{i}_b = \begin{bmatrix} \begin{bmatrix} \nu_1 \\ \vdots \\ \nu_n \end{bmatrix} \\ \vdots \\ \begin{bmatrix} \nu_1 \\ \vdots \\ \nu_n \end{bmatrix} \end{bmatrix} \tag{3.15}$$

where ν_i is element i of f_grid and n the length of the same vector. The order of the angles inside mblock_za_grid and mblock_aa_grid is followed when looping the pencil beam directions, where the azimuth angle direction is the innermost loop. That is, for 2D antenna patterns all azimuth angles are looped for the first zenith angle etc.

The workspace variable sensor_response is here denoted as $\mathbf{H}_b$. It is applied on each $\mathbf{i}_b$ and the results are appended vertically, following the order of the positions in sensor_pos

$$\mathbf{y} = \begin{bmatrix} \mathbf{H}_b\mathbf{i}_{b,1} \\ \mathbf{H}_b\mathbf{i}_{b,2} \\ \vdots \\ \mathbf{H}_b\mathbf{i}_{b,n} \end{bmatrix} \tag{3.16}$$

where 1 indicates the first sensor position etc. This equation shows that sensor_response shall contain at least a description of the antenna response. The matrix `Hb` can also cover other sensor characteristics and data reduction if the features of concern are common for all measurement blocks.

As the sensor line-of-sight and block grid values are just added, there is an ambiguity of the line-of-sight. It is possible to apply a constant off-set to the line-of-sights, if the block grids are corrected accordingly. For example, if the simulations deal with limb sounding and a 1D atmosphere, where normally a single block should be used despite a number of spectra are recorded, it could be practical to set the line-of-sight to the viewing direction of the uppermost or lowermost spectrum, and the zenith angles in mblock_za_grid will not be centred around zero which is the case when the "true" line-of-sight is used.

It should be noted that the compulsory sensor variables give no information about the content of the obtained $\mathbf{y}$, as it is not clear which parts and features the block transfer matrix covers. If `Hb` only incorporates the antenna pattern, the result is a set of hypothetical spectra corresponding to a point inside the sensor. On the other hand, if `Hb` includes the whole of the sensor and an eigenvector data reduction, the result is not even a spectrum in traditional way, it is just a column of coefficients with a vague physical meaning.

3.8 Calculation accuracy

The accuracy of the calculations depends on many factors. For many factors, such as spectroscopic parameters, there is nothing else to do than using best available data. On the other hand, for other factors there is a trade-off between accuracy and speed. More accurate calculations requires normally also more computer memory. All different grids and the propagation path step length fall into this category of accuracy factors. It could be worth discussing the selection of atmospheric grids and the path step length as there can be some confusion about how that affects the accuracy.

The main purpose of the atmospheric grids (p_grid, lat_grid and lon_grid) is to build up the mesh on which the atmospheric fields are defined. This means that the spacing of these grids shall be selected having the representation of the atmospheric fields in mind. That is, the spacing shall be fine enough that the atmospheric field is sufficiently well approximated by the piecewise (multi-)linear representation between the grid crossings. The result is that a finer spacing must be used to represent correctly atmospheric fields with a lot of structure, while the grids can have fewer points when the atmospheric fields are smooth.

The accuracy when performing the actual radiative transfer calculations depends on the refinement of the expressions used and the discretisation of the propagation path. If Equation 3.8 is used, the underlaying assumption is that the Planck function and the absorption vary linearly along the propagation path step. These assumptions are of course less violated if the path step length is made small. An upper limit of the path step length is set by ppath_lmax. In many cases it should suffice to just include path points at the crossings of the atmospheric grids (ppath_lmax≤ 0). An exception can be limb sounding where the path step length can be very long around the tangent point, but a limit of about 25 km should suffice normally.

As points are always included in the propagation paths at the crossings of the atmospheric grids, finer grids will give shorter path steps. However, it is neither good practise or efficient to use the atmospheric grids to control the accuracy of the radiative transfer calculations. An upper limit on the path step length shall be applied for this purpose.[1]

[1]Further discussion can be found in message 399 and 410 of the ARTS developers mailing list.

Chapter 4

Gas absorption

4.1 Introduction

The absorption coefficient describes how strongly the atmosphere is absorbing (and emitting). Its unit (in ARTS) is m^{-1}. For a definition, see for example Equation 4.1. Both gases and particles in the atmosphere absorb, and the total absorption is the sum of these two contributions. But this chapter is only about the gas absorption.

When calculating radiative transfer, the local absorption coefficient at each point in the atmosphere has to be known. Furthermore, if one also wants to calculate Jacobians, then the partial absorption coefficients for different atmospheric components (different absorption species) also have to be known.

This chapter discusses different practical aspects of absorption in ARTS. Section 4.2 explains how absorption is handled inside radiative transfer calculations. Section 4.3 discusses how absorption is actually calculated, and how the calculation is set up. Finally, Section 4.4 describes how absorption is stored in a lookup table, and how it is extracted again.

Here in the User Guide we focus on practical aspects of absorption in ARTS. But absorption calculations also have a deep theoretical background, particularly the line-by-line calculations and the continuum models. Some of this background is discussed in *ARTS Theory*, Chapter 2.

History

2011-07-05	Added intro and sections on abs in RT and abs calculation. Also revised lookup table section. First attempt of a complete absorption chapter for ARTS2.
2003-03-28	Documentation for WSM abs_fieldCalc extended by Stefan Buehler after comment from Sreerekha T. R..
2003-03-10	Lookup tables added by Stefan Buehler.
2002-06-04	Restarted for ARTS-1-1 by Stefan Buehler.

Table 4.1: Examples of symbols used in this chapter, the corresponding notation in the ARTS source code and a short description of the quantity.

Here	Unit	In ARTS	Description
I	$\frac{\text{W}}{\text{m}^2\ \text{Hz sr}}$	i_rte, i_field, ...	Intensity
l	m		Path length element
κ_i	m^2	xsec	Absorption cross section of absorbing species i
n_i	m^{-3}		Number density of species i
α_i	m^{-1}	abs_scalar_gas	Absorption coefficient of absorbing species i
α_{total}	m^{-1}		Total gas absorption coefficient

4.2 Gas absorption in radiative transfer simulations

The interface between the radiative transfer (RT) part of ARTS and the absorption part of ARTS is very simple, so this will be a short section.

The interface consists in the agenda abs_scalar_gas_agenda. RT functions execute this agenda whenever they need local absorption coefficients. See the built-in documentation for the exact input and output arguments of the agenda. The idea is that input arguments are the local atmospheric conditions (temperature, pressure, trace gas volume mixing ratios, etc.). These are all scalars.

The output of the agenda is a single variable, abs_scalar_gas, a matrix with dimensions of frequency times absorption species. In a typical ARTS run, this agenda will be executed many times over, for different points in the atmosphere.

Typical contents of this agenda are as follows:

4.2.1 On-the-fly absorption

```
AgendaSet( abs_scalar_gas_agenda )
{
 abs_scalar_gasCalcLBL
}
```

This will calculate absorption coefficients for the exact local conditions that are given. The setup for the calculation (which absorption species and many other details of the calculation) has to be done before, outside the agenda. More information on how to set up the absorption calculation can be found in Section 4.3.

4.2.2 Absorption from lookup table

```
AgendaSet( abs_scalar_gas_agenda )
{
 abs_scalar_gasExtractFromLookup
}
```

This will extract absorption coefficients from a pre-calculated lookup table. More information on the lookup table and how to generate it can be found in Section 4.4.

4.2.3 Doppler shift

Both abs_scalar_gasCalcLBL and abs_scalar_gasExtractFromLookup take into account a possible Doppler shift. However, there is an important difference in the implementation: For abs_scalar_gasCalcLBL the shift is implemented by simply making the line-by-line calculation on a shifted frequency grid, so the treatment is exact. For abs_scalar_gasExtractFromLookup, on the other hand, the shift is implemented numerically, by shifting the frequency grid and then linearly interpolating the pre-calculated absorption data.

4.3 Calculating gas absorption

This section deals with calculating gas absorption coefficients in ARTS. This typically occurs in two different contexts, either on-the-fly in the radiative transfer calculation (see Section 4.2), or when preparing a gas absorption lookup table (see Section 4.4). A third and more unusual case is that the user is only interested in the absorption itself, for a particular atmospheric scenario, and not in the radiative transfer simulation.

In all these cases, the same core absorption routines will be used, although the interfaces differ.

4.3.1 Absorption species

Absorption is additive, so the total absorption is simply the sum of all partial absorptions. And the partial absorption for gases that have spectral lines can be calculated as a sum over the absorption of each spectral line, plus some more or less empirical continuum terms.

An absorption species in ARTS is an abstract entity that has a partial absorption coefficient associated with it, and that usually can be associated with a volume mixing ratio of a corresponding gas (the VMRs are stored in variable vmr_field). Total absorption is the sum of the partial absorptions of all absorption species. Absorption species are defined in the ARTS controlfile by special 'tags', which are stored in the variable abs_species, and set by the method SpeciesSet.

The absorption species tags describe not only the different absorbers, but also the model that should be used to calculate the absorption. There are two types of tags, those for explicit line-by-line calculations, and those for continua and complete absorption models. An example of the first kind is 'H2O-18', which identifies a particular isotopologue of water vapor. An example of the second kind is 'H2O-ForeignContCKDMT100', which identifies a particular continuum model. Tags can be combined, if they refer to the same molecule (different isotopologues are allowed). Even continuum tags can be combined with explicit line-by-line tags, if they refer to the same molecule.

It should be noted that isotopic ratios are taken into account implicitly when line strengths are calculated, so even if you make calculations for individual isotopologues, the VMR numbers in the variable vmr_field should not be adjusted for the isotopic ratio. As an example, to make a line-by-line calculation for all ozone isotopologues, you could represent them in different ways by SpeciesSet.

```
a) SpeciesSet(abs_species,
              ["O3"])
```

```
b) SpeciesSet(abs_species,
              ["O3-666, O3-668, O3-686, O3-667, O3-676"])
c) SpeciesSet(abs_species,
              ["O3-666", "O3-668", "O3-686", "O3-667", "O3-676"])
```

Options (a) and (b) are equivalent, you will have one ozone species that represents are isotopologues, and that will be associated with a single VMR field in vmr_field. With option (c) you have five different ozone species, so you have to supply five different VMR fields. If those five fields are identical (exactly same numerical values), you will get the same total absorption as with options (a) and (b).

Overall, the tag mechanism allows quite complex absorption setups. The built-in documentation for SpeciesSet gives a detailed explanation of the tag syntax and some examples.

It is important to note that there is no 'intelligence' in ARTS that checks that the chosen tag combination makes sense, so the user should know what she or he is doing, or follow one of the many examples in the ARTS `includes` directory.

4.3.2 Explicit line-by-line calculations

For absorption species with explicit line-by-line calculation the calculation involves the steps summarized in Table 4.2. The list of variables and methods in the table is not complete, the idea is to give an overview over the important ones and show how they work together. Missing are particularly the input variables that describe the atmospheric conditions, and continuum description variables, which normally do not have to be set by the user anyway.

Chapter 2 of *ARTS Theory* contains more information on the internal format of the spectral line data. It also contains theoretical background for the calculation itself.

See the built-in documentation of the various variables and methods for more information. It is on purpose not repeated here, for better maintainability. If you are viewing this pdf file on a computer, just click on a variable or method name to get to the corresponding built-in documentation.

4.3.3 Continua and complete absorption models

ARTS includes many absorption continua and complete absorption models, which are described in *ARTS Theory*, Chapter 2. The common property of all of these is that they do not use the standard ARTS line-by-line calculation mechanism. They may include spectral lines, but then these lines are hardwired into the absorption model itself. Consequently, the first four steps in Table 4.2 are not needed for these models.

The pure continua are intended to be used together with an explicit ARTS line-by-line calculation, the complete models are intended to be used alone. To select a continuum or complete absorption model, simply use the corresponding tag with SpeciesSet. Currently available models are listed in Table 4.3.

The names should be fairly self-explanatory and can be used to find background information on the various models in *ARTS Theory*. The condensate absorption models are a bit special and perhaps need some extra explanation. They are absorption parameterizations by Liebe, and allow the inclusion of condensate in the (rare) cases where scattering is not important. Their general applicability is therefore fairly limited.

#	Step	Variables and Methods
1	Define line shape function(s) to use.	Variable: abs_lineshape. Methods: abs_lineshapeDefine (same shape for all species), abs_lineshape_per_tgDefine (different shapes for different species).
2	Read spectral line data (the order of the first two steps does not matter).	Variable: abs_lines. Methods: abs_linesReadFromArts, abs_linesReadFromHitran, abs_linesReadFromHitran2004, abs_linesReadFromJpl, abs_linesReadFromMytran2 (different methods are for different catalogue formats). For the ARTS internal format, the standard method ReadXML works also, but does not allow to select a frequency range, as the others do.
3	Split line data for different absorption species.	Variable: abs_lines_per_species. Methods: abs_lines_per_speciesCreateFromLines. Alternatively, read lines from different catalogues for different species directly with abs_lines_per_speciesReadFromCatalogues.
4	Optimize line data.	Variable: abs_lines_per_species. Methods: Add mirror lines for VVW line shape with abs_lines_per_speciesAddMirrorLines (see *ARTS Theory*, Chapter 2). Remove lines that are outside the line shape cutoff with abs_lines_per_speciesCompact.

The first four steps are preparation, and typically have to be done only once per ARTS run. The fifth step is the actual absorption calculation, which can occur in different contexts.

5a	Calculate absorption on-the-fly.	Agenda: abs_scalar_gas_agenda. Variable: abs_scalar_gas. Methods: abs_scalar_gasCalcLBL. Alternative: abs_scalar_gasExtractFromLookup (extract absorption from pre-calculated lookup table).
5b	Calculate absorption lookup table.	Variable: abs_lookup. Methods: abs_lookupCreate. Alternative: Load lookup table from file with ReadXML, it then has to be adapted to the current calculation (and checked) with abs_lookupAdapt.
5c	Just calculate absorption (not for RT purposes).	Variable: abs_coef. Methods, high level: abs_coefCalc, abs_coefCalcSaveMemory. Methods, low level: abs_xsec_per_speciesInit, abs_xsec_per_speciesAddLines (the core method for the actual line-by-line calculation, used internally by all higher level methods), abs_xsec_per_speciesAddConts (add continua or complete absorption models, see Section 4.3.3), abs_coefCalcFromXsec (calculate absorption coefficients from absorption cross-sections).

Table 4.2: Steps for line-by-line absorption calculation, and associated ARTS workspace variables and methods.

Class	Tag name
Water vapor continua	H2O-SelfContStandardType
	H2O-ForeignContStandardType
	H2O-ForeignContMaTippingType
	H2O-ContMPM93
	H2O-SelfContCKDMT100
	H2O-ForeignContCKDMT100
	H2O-SelfContCKD222
	H2O-ForeignContCKD222
	H2O-SelfContCKD242
	H2O-ForeignContCKD242
	H2O-SelfContCKD24
	H2O-ForeignContCKD24
	H2O-ForeignContATM01
Complete water vapor models	H2O-CP98
	H2O-MPM87
	H2O-MPM89
	H2O-MPM93
	H2O-PWR98
Carbon dioxide continua	CO2-CKD241
	CO2-CKDMT100
	CO2-SelfContPWR93
	CO2-ForeignContPWR93
Oxygen continua	O2-CIAfunCKDMT100
	O2-v0v0CKDMT100
	O2-v1v0CKDMT100
	O2-SelfContStandardType
	O2-SelfContMPM93
	O2-SelfContPWR93
Complete oxygen models	O2-PWR98
	O2-PWR93
	O2-PWR88
	O2-MPM93
	O2-MPM92
	O2-MPM89
	O2-MPM87
	O2-MPM85
Nitrogen continua	N2-SelfContMPM93
	N2-SelfContPWR93
	N2-SelfContStandardType
	N2-SelfContBorysow
	N2-CIArotCKDMT100
	N2-CIAfunCKDMT100
	N2-DryContATM01
Condensate absorption models	liquidcloud-MPM93
	icecloud-MPM93
	rain-MPM93

Table 4.3: ARTS continua and complete absorption models. The molecular species can be inferred from the start of the tag name. See *ARTS Theory*, Chapter 2 for more information on the various models.

The behavior of the continua and complete absorption models can be modified by passing them some additional parameters, stored in the variables abs_cont_names, abs_cont_models, and abs_cont_parameters. Basically, abs_cont_names identifies the model, abs_cont_models contains switches that select different behavior (for example taking only the lines, or only the continuum part of a complete model), and abs_cont_parameters can contain numerical parameters.

Yes, the nomenclature for these additional continuum parameters, particularly abs_cont_models, is confusing. However, most users will never have to deal with these variables explicitly. They are set to default values in the include file `continua.arts`. Users should therefore always include this file at the start of their controlfiles with

```
INCLUDE "continua.arts"
```

Unless you work on continuum model development of verification, you should never have to modify these default settings.

The core method to calculate continua and complete absorption models is abs_xsec_per_speciesAddConts. Users normally do not have to call this method explicitly, since it is used implicitly by higher level methods, such as abs_coefCalc.

4.4 The gas absorption lookup table

4.4.1 Introduction

Calculating gas absorption coefficient spectra in a line by line way is quite an expensive thing to do. Sometimes contributions from thousands or ten thousands of lines have to be summed up. To make matters worse, this has to be done over and over again for each point in the atmosphere.

Actually, the absorption coefficient depends not directly on position, but on the atmospheric state variables:

- Pressure
- Temperature
- Trace gas concentrations

The basic idea of the lookup table is to pre-calculate absorption for discrete combinations of these variables, and then use interpolation to extract absorption for the actual atmospheric state.

The lookup table concept and implementation is described only very briefly here in the user guide. Much more details and validation results can be found in *Buehler et al.* [2011].

4.4.2 Lookup table concept

The fundamental law of Beer[1] states that extinction is proportional to the intensity of radiation, and to the amount of absorbing substance:

$$\frac{dI}{dl} = -I\sum_i \kappa_i n_i = -I\sum_i \alpha_i = -I\alpha_{\text{total}} \tag{4.1}$$

[1]According to C. Melsheimer, Beer's law is: 'The taller the glass, the darker the brew, the less the amount of light that comes through'. He might have been quoting someone else, there, but I do not know whom.

where the meaning of the symbols is defined in Table 4.1.

As one can see from the above equation, a large part of the pressure dependence of α_i comes from n_i. (If one assumes constant volume mixing ratio of species i, then n_i is proportional to the total pressure according to the ideal gas law.) Therefore, the lookup table should store κ, rather than α. We then have to worry only about the dependence of κ on the atmospheric state variables.

Pressure dependence

The pressure dependence is the most important dependence of κ. It comes from the fact that the width of the line shape functions is governed by pressure broadening. We have to store the κ_i on some pressure grid and interpolate if we need them for intermediate values.

Temperature dependence

This is the next effect to take into account. Both the line widths and the line intensities depend on temperature. Of course, only certain combinations of pressure and temperature occur in the Earth's atmosphere. Hence, storing the κ_i in a two dimensional table as a function of pressure and temperature would waste a lot of space. Instead, they are stored for a reference temperature and set of temperature perturbations for each pressure level. E.g., if the set of perturbations is $[-10,\ 0,\ +10]$, then the κ_i would be stored for three different temperatures for each pressure level: $[T_\mathrm{R}(p) - 10\,\mathrm{K},\ T_\mathrm{R}(p),\ T_\mathrm{R}(p) + 10\,\mathrm{K}]$, where $T_\mathrm{R}(p)$ is the reference temperature for each pressure level.

Trace gas concentration dependence

This is a second order effect. The width of the line depends not only on total pressure, but also on the partial pressure of one or more trace gases. In theory this is always the case, because the broadening is different for each combination of collision partners. However, in practice trace gas concentrations in the Earth's atmosphere are normally so low that this can be safely neglected. An important exception is water vapor in the lower troposphere, which can reach quite high volume mixing ratios. Therefore, the effect of water vapor mixing ratio on water vapor absorption (self broadening), as well as on oxygen absorption (for example according to the parameterization by *Rosenkranz* [1993]) may not be negligible.

This is handled by storing water vapor perturbations. In contrast to the temperature case, the water vapor perturbations are multiplicative, not additive. Hence, if the set of perturbations is $[0,\ 1,\ 10]$, then the κ_i would be stored for three different H_2O VMRs for each pressure/temperature grid point: $[0,\ \mathrm{VMR}_\mathrm{R}(p,T),\ 10{*}\mathrm{VMR}_\mathrm{R}(p,T)]$, where $\mathrm{VMR}_\mathrm{R}(p,T)$ is the reference water vapor VMR for each pressure/temperature grid point.

Interpolation

The interpolation scheme is quite important for the accuracy of the lookup table. In particular, higher order interpolation gives considerably better accuracy for the same table grid spacing. The interpolation orders in the ARTS implementation of the lookup table can be chosen by the user. The settings that are recommended, and set as defaults in file general.arts, are quite high interpolation orders of 5, 7, and 5 for pressure, temperature, and water vapor, respectively. Such high orders are only appropriate because the function to be interpolated (the κ_i) is very smooth.

4.4.3 Workspace variables and methods

The gas absorption lookup table is implemented by the class GasAbsLookup, which resides in the files gas_abs_lookup.cc and gas_abs_lookup.h.

The lookup table itself is stored in the workspace variable abs_lookup. It can be generated with the method abs_lookupCreate. ARTS also includes some methods that automatically set input parameters for abs_lookupCreate, such as grid ranges and reference profiles of pressure, temperature, and trace gas concentrations. These methods are abs_lookupSetup, abs_lookupSetupBatch, and abs_lookupSetupWide. The first two will take into account the actual atmospheric state, or set of atmospheric states, for the calculation. The third alternative simply sets up a table that should cover most reasonable atmospheric conditions. *Buehler et al.* [2011] contains more information on these setup methods.

Alternatively, the table can be loaded from a file with ReadXML. After loading, the method abs_lookupAdapt has to be called. It will make sure that the lookup table agrees exactly with your calculation. For example, it has to check that the frequencies that you want to use are included in the set of frequencies for which the table has been calculated. There is no interpolation in frequency. This is on purpose, because the gas absorption spectrum is the quantity that changes most rapidly as a function of frequency. Frequency interpolation here could be quite dangerous. The abs_lookupAdapt method also sorts the species in exactly the same way that they occur in your calculation. It sets the variable abs_lookup_is_adapted to flag that the table is now ok.

When the table has been successfully adapted, one can extract absorption coefficients with the method abs_scalar_gasExtractFromLookup. This will extract *absorption coefficients*, i.e., the cross sections stored in the table are not only interpolated to the desired atmospheric conditions, but are also multiplied with the partial number density of the present absorbers.

The abs_scalar_gasExtractFromLookup method is meant to be used inside the agenda abs_scalar_gas_agenda, which is used in several places where absorption coefficients are needed, both inside the scattering box and outside.

It is also possible to calculate absorption for the entire atmospheric field. This is done by the method abs_fieldCalc, and is useful for testing and plotting gas absorption. (For RT calculations, gas absorption is calculated or extracted locally, therefore there is no need to calculate a global field. But the abs_fieldCalc method is handy for easy plotting of absorption vs. pressure, for example.)

Because of the different usage contexts, the method abs_scalar_gasExtractFromLookup can calculate absorption either for all frequencies in the frequency grid (input variable f_index<0), or just for the frequency indicated by the input variable f_index (f_index>=0).

Chapter 5

Geoid and surface properties

5.1 The geoid

The geoid is an imaginary surface used as a reference when specifying the surface altitude and the altitude of pressure levels. Any shape of the geoid is allowed but a smoothly varying geoid is the natural choice, with the centers of the geoid and the coordinate system coinciding. The geoid should normally be set to the reference ellipsoid for some global geodetic datum, such as WGS-84.

Inside ARTS, the geoid is represented as a matrix (r_geoid), holding the geoid radius, $r_\odot$, for each crossing of the latitude and longitude grids, $r_\odot = r_\odot(\alpha, \beta)$. The geoid is not defined outside the ranges covered by the latitude and longitude grids, with the exception for 1D where the geoid by definition is a full sphere.

5.1.1 Geoid ellipsoids

A geodetic datums is based on a reference ellipsoid. The ellipsoid is rotationally symmetric around the north-south axis. That is, the ellipsoid radius has no longitude variation, it is only a function of latitude. The ellipsoid is described by an equatorial radius, r_e, and a polar radius, r_p. These radii are indicated in Figure 5.1. The radius of the ellipsoid for a given latitude, α, is

$$r_\odot(\alpha) = \sqrt{\frac{r_e^2 r_p^2}{r_e^2 \sin^2\alpha + r_p^2 \cos^2\alpha}} \tag{5.1}$$

The radius given by Equation 5.1 can be directly applied for 2D and 3D cases. On the other hand, for 1D cases the reference geoid is by definition a sphere and the radius of this sphere shall be selected in such way that it represents the local shape of a reference ellipsoid. This is achieved by setting $r_\odot$ to the radius of curvature of the ellipsoid. The curvature radius differs from the local radius except at the equator and an east-west direction. For example, at

History

110614 Extended and revised (Patrick Eriksson).

050613 First version finished by Patrick Eriksson.

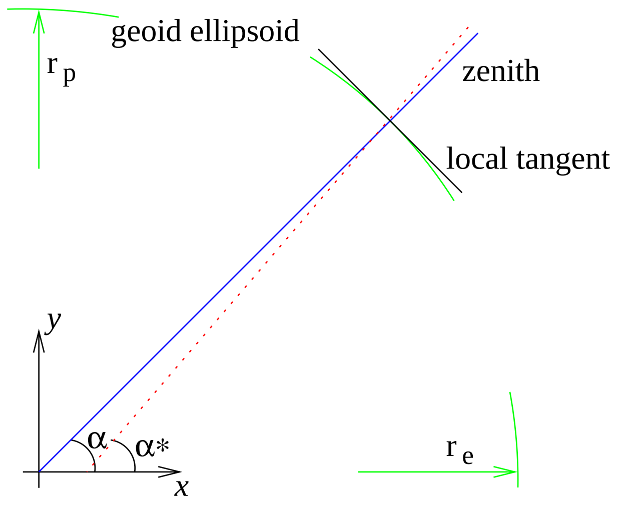

Figure 5.1: Definition of the ellipsoid radii, r_e and r_p, geocentric latitude, α, and geodetic latitude, $\alpha*$. The dotted line is the normal to the local tangent of the geoid ellipsoid. The zenith and nadir directions, and geometrical altitudes, are here defined to follow the solid line.

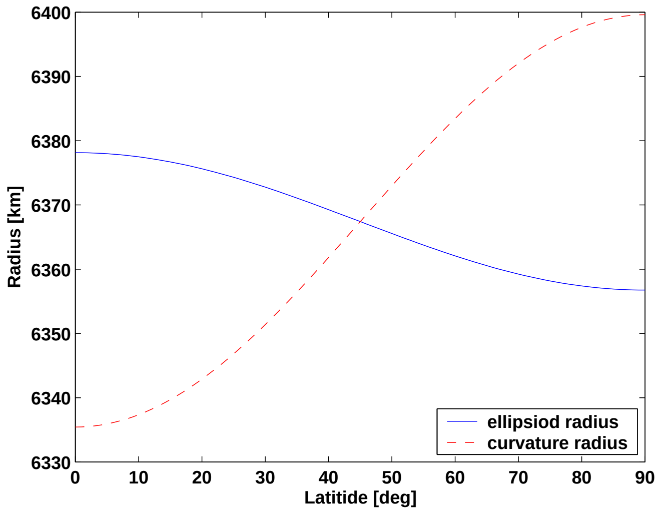

Figure 5.2: The ellipsoid radius ($r_\odot$) and curvature radius (r_c) for the WGS-84 reference ellipsoid. The curvature radii are valid for the north-south direction.

the equator and a north-south direction, the curvature radius is smaller then the local radius, while at the poles (for all directions) it is greater (see further Figure 5.2). The curvature radius, r_c, of an ellipsoid is [*Rodgers*, 2000]

$$r_c = \frac{1}{r_{ns}^{-1}\cos^2\alpha + r_{ew}^{-1}\sin^2\alpha} \tag{5.2}$$

where r_{ns} and r_{ew} are the north-south and east-west curvature radius, respectively,

$$r_{ns} = r_e^2 r_p^2 (r_e^2 \cos^2\omega + r_p^2 \sin^2\omega)^{-\frac{3}{2}} \tag{5.3}$$

$$r_{ew} = r_e^2 (r_e^2 \cos^2\omega + r_p^2 \sin^2\omega)^{-\frac{1}{2}} \tag{5.4}$$

The azimuth angle, ω, is defined in Section 3.7.2. The latitude and azimuth angle to apply in Equations 5.2–5.4 shall rather be valid for a middle point of the propagation paths (such as some tangent point), instead of the sensor position.

Table 5.1 gives the equatorial and polar radii of the reference ellipsoid for the geodetic datums handled by ARTS.

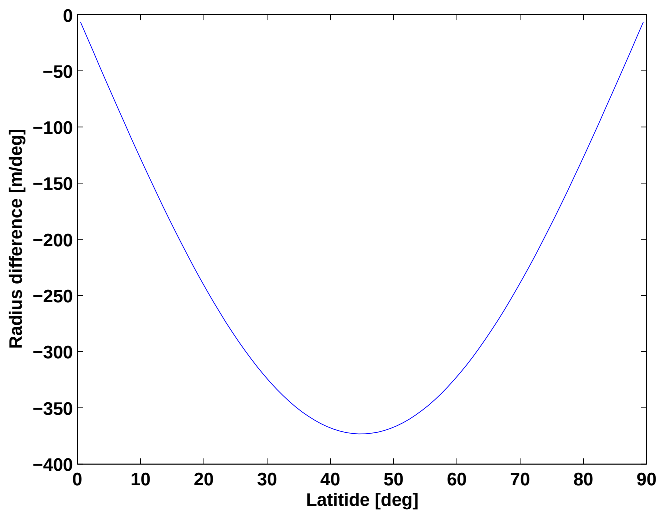

Figure 5.3: The change of the WGS-84 ellipsoid radius for 1° latitude differences.

Datum	r_e	r_p	$1/f$	Reference
WGS-84	6378.137 km	*6356.752 km*	298.2572235	*Montenbruck and Gill* [2000]

Table 5.1: Equatorial and polar radius of reference ellipsoids. Values given as *italic* are derived by the other two values and Equation 5.6.

5.1.2 Geocentric and geodetic latitudes

The fact that the geoid is an ellipsoid, instead of a sphere, opens up for the two different definitions of the latitude. The geocentric latitude, which is the the one used here, is the angle between the equatorial plane and the vector from the coordinate system centre to the position of concern. The geodetic latitude is also defined with respect to the equatorial plane, but the angle to the normal to the reference ellipsoid is considered here, as shown in Figure 5.1. It could be mentioned that a geocentric latitude does not depend on the geoid ellipsoid used, while the geodetic latitudes change if another reference ellipsoid is selected. An approximative relationship between the geodetic (α^*) and geocentric (α) latitudes is [*Montenbruck and Gill*, 2000]

$$\alpha^* = \alpha + f\ \sin(2\alpha) \tag{5.5}$$

where f is the flattening of the ellipse:

$$f = \frac{r_e - r_p}{r_e} \tag{5.6}$$

The value of f for the Earth is about 1/298.26. This means that the largest differences between α and α^* are found at mid-latitudes and the maximum value is about 12 arc-minutes.

5.2 Surface altitude

The surface altitude, z_g, is given as the geometrical altitude above the geoid. The radius for the surface is accordingly

$$r_s = r_\odot + z_s \tag{5.7}$$

As also mentioned in Section 2.5, a gap between the surface and the lowermost pressure level is not allowed.

The ARTS variable for the surface altitude (z_surface) is a matrix of the same size as the geoid matrix. For 1D, the surface is a sphere by definition (as the geoid), while for 2D and 3D any shape is allowed and a rough model of the surface topography can be made.

5.3 Surface emission and reflection

An introduction to the treatment of surface emission and reflections is given in Section 3.4.2. As described in that section, the surface properties are described by three workspace variables and the methods developed to handle different surface properties set these variables: surface_emission, surface_los and surface_rmatrix. The sections below outlines the methods available, how these set the output variables. Section 5.7 of *ARTS Theory* provides the theoretical background.

5.3.1 Blackbody surface

If the surface can be assumed to act as a blackbody, the workspace method surfaceBlackbody can be used. This method sets surface_emission to $[B, 0, 0, 0]^T$, and surface_los and surface_rmatrix to be empty.

5.3.2 Specular reflections

Several methods to incorporate a flat surface exist, including surfaceFlatRefractiveIndex and surfaceFlatSingleEmissivity. The methods differ in how the dielectric properties of the surface are given, and if these are constant or not with frequency.

In the case of specular reflections, surface_los has the length 1. The specular direction is calculated by the internal function `surface_specular_los`[1]. Equations 5.67-5.69 in *ARTS Theory*give the values of surface_rmatrix and surface_emission.

5.3.3 Lambertian surface

A basic treatment of Lambertian surfaces is provided by the method surfaceLambertianSimple. This method assumes that the down-welling radiation has no azimuthal dependency, which fits the assumptions for 1D atmospheres. The number of angles to apply in surface_los is selected by the user.

[1]Any tilt of the surface is neglected when determining the specular direction. If there would be any need to consider surface tilt, almost complete code for this task existed in surface_specular_los but was removed in version 1-1-876. The code can be obtained by e.g. checking out version 1-1-875.

For a Lambertian surface the reflected radiation is unpolarised (thus independent of the polarisation of the down-welling radiation). That is, each surface_rmatrix has the structure:

$$\mathbf{R} = \begin{bmatrix} w & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}. \tag{5.8}$$

When determining the "weight" w above, the method assumes that the down-welling radiance (I) is constant inside each zenith angle range: $[\theta_a, \theta_b]$. Hence, w equals (cf. Equation 5.70 of *ARTS Theory*)

$$w = \int_{\theta_a}^{\theta_b} \int_{\phi_a}^{\phi_b} \cos(\theta) f(\theta, \phi, \theta_1, \phi_1) \sin(\theta) \,\mathrm{d}\phi \,\mathrm{d}\theta. \tag{5.9}$$

that gives

$$w = \frac{r_d}{2} \left[\cos(2\theta_a) - \cos(2\theta_b) \right]. \tag{5.10}$$

Thus, this value is a combination of the surface reflectivity and an solid angle weight.

The emission (surface_emission) becomes:

$$\mathbf{b} = \begin{bmatrix} r_d B \\ 0 \\ 0 \\ 0 \end{bmatrix}. \tag{5.11}$$

Chapter 6

Propagation paths

A propagation path is the name given in ARTS to the way the radiation travels to reach the sensor for a specified line-of-sight. Propagation paths are introduced in Section 3.3 and this section provides further details. For a general usage of ARTS, it should suffice to read Section 6.1. The remaining sub-sections deal with more low-level aspects of the calculations, and are of interest only if you want to understand the finer details of ARTS. It turned out that the main obstacle to handle 2D and 3D cases was to find stable algorithms for the propagation path calculations, and the sub-sections starting with Section 6.2 aim also at documenting the experience of that development work.

6.1 Practical usage

The ray tracing algorithm to be applied for the calculation of propagation path is effectively selected by specifying ppath_step_agenda (see further Section 3.3). The fastest calculations are obtained if refraction is neglected, denoted as geometrical calcutions. The workspace method to apply if this assumption can be made is ppath_stepGeometric.

The main consideration for using ppath_stepGeometric is to select a value for ppath_lmax. This variable controls to some extent the calculation accuarcy, as described in Section 3.8. This variable sets the maximum distance between points of the (final) propagation path. Set this variable to e.g. -1 if you don't want to apply such a length criterion.

A straightforward, but inefficient, treatment of refraction is provided by ppath_stepRefractionEuler. This method divides the propagation path into a series of geomtrical ray tracing steps. The size of the ray tracing steps is selected by ppath_lraytrace. This variable affects only the ray tracing part, the distance between points of the propagation path actually returned is controled by ppath_lmax as above.

History

110613	Slightly extended and revised (except equations) by Patrick Eriksson.
030310	First complete version written by Patrick Eriksson.

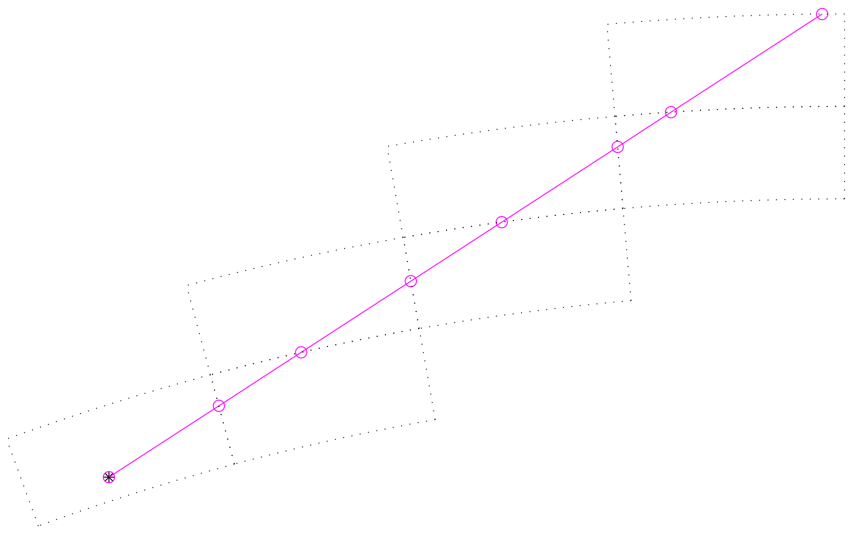

Figure 6.1: Tracking of propagation paths. For legend, see Figure 6.2. The figure tries to visualize how the calculations of propagation paths are performed from one grid cell to next. In this example, the calculations start directly at the sensor position ($*$) as it placed inside the model atmosphere. The circles give the points defining the propagation path. Path points are always included at the crossings of the grid cell boundaries. Such a point is then used as the starting point for the calculations inside the next grid cell.

6.2 Calculation approach

The propagation paths are calculated in steps, as outlined in Section 3.3. The path steps are normally from one crossing of the atmospheric grids to next. To introduce propagation paths steps was necessary to handle the iterative solution for scattering inside the cloud box, as made clear from Figure 10.2.

A full propagation path is stored in the workspace variable ppath, that is of the type Ppath (see Section 6.3). The paths are determined by calculating a number of path steps. A path step is the path from a point to the next crossing of either the pressure, latitude or longitude grid (Figure 6.1). There is one exception to this definition of a path step, and that is when there is an intersection with the surface, which ends the propagation path at that point. The starting point for the calculation of a path step is normally a grid crossing point, but can also be an arbitrary point inside the atmosphere, such as the sensor position. Only points inside the model atmosphere are handled. The path steps are stored in the workspace variable ppath_step, that is of the same type as ppath.

Propagation paths are calculated with the internal function `ppath_calc`. The communication between this method and ppath_step_agenda is handled by ppath_step. That variable is used both as input and output to ppath_step_agenda. The agenda gets back ppath_step as returned to ppath_calc and the last path point hold by the structure is accordingly the starting point for the new calculations. If a total propagation path shall be determined, the agenda is called repeatedly until the starting point of the propagation path is found and ppath_step will hold all path steps that together make up ppath. The starting point is included in the

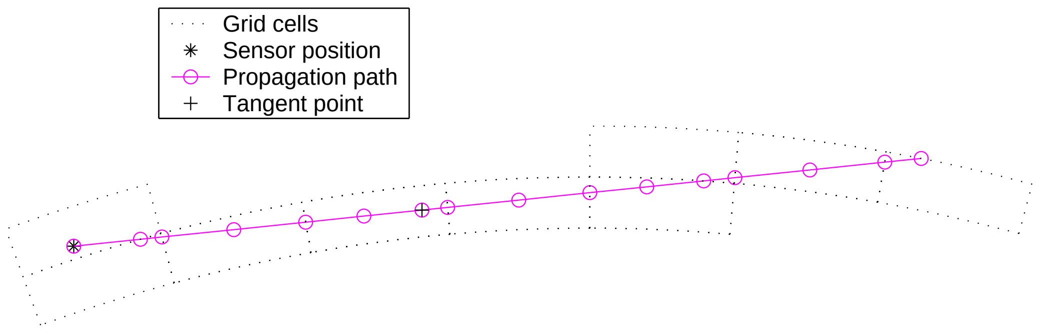

Figure 6.2: As Figure 6.1, but with a length criterion for the distance between the points defining the path. The inclusion of the tangent point is not a result of this length criterion, it is always included among the path points.

returned structure.

The path is determined by starting at the end point and moving backwards to the starting point. The calculations are initiated by filling ppath_step with the practical end point of the path. This is either the position of the sensor (true or hypothetical), or some point at the top of the atmosphere (determined by geometrical calculations starting at the sensor). This initialization is not handled by ppath_step_agenda. The field `constant` is set by ppath_calc to the correct value if the sensor is above the model atmosphere. Otherwise, the field is set to be negative and is corrected by ppath_step_agenda at the first call. This procedure is needed as the propagation path constant changes if refraction is considered, or not, when the sensor is placed inside the atmosphere.

The agenda performs only calculations to next crossing of a grid, all other tasks are performed by ppath_calc, with one exception. If there is an intersection with the surface, the calculations stop at this point. This is flagged by setting the background field of ppath_step. Beside this, ppath_calc checks if the starting point of the calculations is inside the scattering box or below the surface level, and check if the last point of the path has been reached.

In many cases the propagation path can/must be considered to consist of several parts. One exemple is surface reflection (see Figure 3.4). The variable ppath describes then only a single part of the propagation path.

6.3 The propagation path data structure

A propagation path is represented by a structure of type `Ppath`. This structure holds also auxiliary variables to facilitate the radiative transfer calculations and to speed up the interpolation. The fields of Ppath are described below, where the data type is given inside square brackets.

dim [Index] The atmospheric dimensionality. This field shall always be equal to the workspace variable atmosphere_dim.

np [Index] Number of positions to define the propagation path. Allowed values are ≥ 0. The number of rows of `pos` and `los`, and the length of `z`, `gp_p`, `gp_lat` and `gp_lon`, shall be equal to `np`. The length of `l_step` is `np` - 1. If `np` $\leq$ 1, the observed spectrum is identical to the radiative background. For cases where the sensor is placed inside the model atmosphere and `np` = 1, the stored position is identical to the sensor position and that position can be used to determinate the radiative background (see below).

constant [Numeric] The propagation path constant. Such a constant can be assigned to all geometrical paths and for 1D cases (with or without refraction). This field can be initiated to a negative value to indicate that the constant is undefined or not yet set. For cases where the constant applies, ppath_step_agenda sets this constant at the first call of the agenda if the given value is negative.

pos [Matrix] The position of the propagation path points. This matrix has `np` rows and up to 3 columns. Each row holds a position where column 1 is the radius, column 2 the latitude and column 3 the longitude (cf. Section 3.7.1). The number of columns for 1D and 2D is 2, while for 3D it is 3. The latitudes are stored for 1D cases as these can be of interest for some applications and are useful if the propagation path shall be plotted. The latitudes for 1D give the angular distance to the sensor (see further Section 2.2). The propagation path is stored in reversed order, that is, the position with index 0 is the path point closest to the sensor (and equals the sensor position if it is inside the atmosphere). The full path is stored also for 1D cases with symmetry around a tangent point (in contrast to ARTS-1).

z [Vector] The geometrical altitude for each path position. The length of this vector is accordingly `np`. This is a help variable for plotting and similar purposes. It shall not be used to interpolate the atmospheric fields, as pressure is the main altitude coordinate.

l_step [Vector] The length along the propagation path between the positions in `pos`. The first value is the length between the first and second point etc. For `np` $\geq$ 2, the length of the vector is `np` - 1. Otherwise it is 0.

gp_p [ArrayOfGridPos] Index position with respect to the pressure grid. The structure for grid positions is described in *ARTS Developer Guide*, Section 5.4.

gp_lat [ArrayOfGridPos] As `gp_p` but with respect to the latitude grid.

gp_lon [ArrayOfGridPos] As `gp_p` but with respect to the longitude grid.

los [Matrix] The line-of-sight of the propagation path at each point. The number of rows of the matrix is `np`. For 1D and 2D, the matrix has a single column holding the zenith angle. For 3D there is an additional column giving the azimuth angle. The zenith and azimuth angles are defined in Section 3.7.2. If the radiative background is the cloud box, the last position (in `pos`) and line-of-sight give the relevant information needed when extracting the radiative background from the cloud box intensity field.

background [String] The radiative background for the propagation path. The possible options for this field are 'space', 'blackbody surface', 'cloud box interior' and 'cloud box surface', where the source of radiation should be clear the content of the strings.

tan_pos [Vector] The position of the tangent point. This vector is only set if there exists a tangent point (above the surface level), the length of the vector is otherwise 0. The tangent point is defined as the point with the lowest radius along the path. This means that (the absolute value of) the zenith angle at the tangent point is always 90°. For 2D and 3D this point can deviate from the point with lowest geometrical altitude.

geom_tan_pos [Vector] The position of the geometrical tangent point. This vector is set for all downward observations. Refraction and surface reflections are neglected when calculating this tangent point position. This field is not handled by ppath_step_agenda. Definition of the tangent point as for `tan_pos`.

nreal [Vector] The real part of the refractive index at each path position. Length is accordingly `np`.

6.4 Structure of implementation

The workspace method for calculating propagation paths is ppathCalc, but this is just a getaway function for ppath_calc. The main use of ppathCalc is to debug and test the path calculations, and that WSM should normally not be part of the control file. Propagation paths, or steps, are generated from inside other functions.

6.4.1 Main functions for clear sky paths

The master function to calculate full clear sky propagation paths is `ppath_calc`. This function is outlined in Algorithm 3. The function can be divided into three main parts, initialization (handled by ppath_start_stepping), a repeated call of ppath_step_agenda and putting data into the return structure (ppath).

The main task of the function `ppath_start_stepping` is to set up ppath_step for the first call of ppath_step_agenda, which means that the practical starting point for the path calculations must be determined. If the sensor is placed inside the model atmosphere, the sensor position gives directly the starting point. For cases when the sensor is found outside the atmosphere, the point where the path exits the atmosphere must be determined. The exit point can be determined by pure geometrical calculations (see Sections 6.6 and 6.7) as the refractive index is assumed to have the constant value of 1 outside the atmosphere. The problem is accordingly to find the geometrical crossing between the limit of the atmosphere and the sensor line-of-sight (LOS). The function performs further some other tasks, which include:

- For all LOS with a zenith angle $\geq 90°$ the position of the geometrical tangent point is calculated.
- If the sensor is placed inside the model atmosphere
  - Checks that the sensor is placed above the surface level. If not, an error is issued.
  - Checks for 2D and 3D and when the sensor position as at an end point of the latitude or longitude grid, that the LOS is inwards with respect to the atmospheric limit.

Algorithm 3 Outline of the function ppath_calc.

```
check consistency of function input
call ppath_start_stepping to set ppath_step
create an array of Ppath structures, ppath_array
add ppath_step to ppath_array
while radiative background not reached do
  call ppath_step_agenda
  if path is at the highest pressure surface then
    radiative background is space
  else if path is at either end point of latitude or longitude grid then
    this is not allowed, issue an runtime error
  end if
  if cloud box is active then
    if path is at the surface of the cloud box then
      radiative background is the cloud box surface
    end if
  end if
  add ppath_step to ppath_array
end while
initialize the WSV ppath to hold found number of path points
copy data from ppath_array to ppath
```

  - If the sensor and surface altitudes are equal, and the sensor LOS is downward, the radiative background is set to be the surface. For 2D and 3D, the tilt of the surface radius is considered when determining if the LOS is downward.
  - If the cloud box is active and the sensor position is inside the cloud box, the radiative backsurface is set to be “cloud box interior”. All sensor positions on the cloud box surface are for 2D and 3D treated as points inside the box (for simplicity reasons), while for 1D the behavior is as expected.

- If the sensor is placed outside the model atmosphere
  - Checks that the zenith angle is $\geq 90°$. Upward observations are here not allowed.
  - If it is found for 2D and 3D that the exit point of the path not is at the top of the atmosphere, but is either at a latitude or longitude end face of the atmosphere, an error is issued. This problem can not appear for 1D.

For further details, see the code.

6.4.2 Main functions for propagation path steps

Example on workspace methods to calculate propagation path steps are ppath_stepGeometric and ppath_stepRefractionEuler. All such methods adapt automatically to the atmospheric dimensionality, but the different dimensionalities are handled by separate internal functions. For example, the sub-functions to ppath_stepGeometric are `ppath_step_geom_1d`, `ppath_step_geom_2d` and `ppath_step_geom_3d`. See `m_ppath.cc` to get the names of the sub-functions for other propagation path step workspace methods.

Algorithm 4 Outline of the function `ppath_step_geom_2d`.

```
call ppath_start_2d
if ppath_step.ppc < 1 then
  calculate the path constant (this is then first path step)
end if
call do_gridcell_2d
call ppath_end_2d
if calculated step ends with tangent point then
  call ppath_step_geom_2d with temporary Ppath structure
  append temporary Ppath structure to ppath_step
end if
```

Many tasks are independent of the algorithm for refraction that is used, or if refraction is considered at all. These tasks are solved by two functions for each atmospheric dimensionality. For 1D the functions are `ppath_start_1d` and `ppath_end_1d`, and the corresponding functions for 2D and 3D are named in the same way. The functions to calculate geometrical path steps are denoted as `do_gridrange_1d`, `do_gridcell_2d` and `do_gridcell_3d`. Paths steps passing a tangent point are handled by a recursive call of the step function. Algorithm 4 summerizes this for geometrical 2D steps.

6.5 General comments

The calculation of propagation paths involves a number of mathematical expressions and they are presented in Sections 6.6–6.8. In addition, the path calculations present a number of practical problems. These practical problems are discussed briefly in this section. For further details, see the code.

6.5.1 Numerical precision

The aim here is not to make a complete discussion around the limited numerical accuracy, but just to point out some of the problems caused. We can start by noticing that the precision with which atmospheric positions can be given is about 0.5 m when the numeric type is float and $2{\cdot}10^{-8}$ m for double (assuming that the mantissa has 24 and 48 bits, respectively). The numbers given correspond to the change of the position for a change of 1 bit, in either radius, latitude and longitude. Already these numbers cause problems for the approach taken to calculate propagation paths. For any path along the border of a grid cell, any rounding error in the wrong direction will move the position outside the grid cell, which would lead to a crash of the code without countermeasures.

The values above give the representation precision. The precision will be even poorer if a position is obtained by calculations as numerical problems tend to accumulate. The calculation precision depends on what mathematical expressions that are involved. For example, a radius or length obtained by the Pythagorean relation will have a relatively high uncertainty as the calculations involve taking the square of a radius in the order of 6400 km. It was found that for calculations performed using only float as numeric type, could lead to displacements from the true position up to 10 m. It was first tried to hard-code double as the numerical type for the most critical passages of the calculations, but a total success was not achieved and some code had to be duplicated (to be used with either the float or double option by if-statements for the pre-compiler) to avoid compiler warnings. A step further was then taken, and double is now hard-coded for all internal variables of `ppath.cc`. This deviation from the rule to have an uniform numeric type inside ARTS was introduced to avoid more complicated coding and it has a very small impact on the overall calculation speed. However, this measure will not lead to that the precision of the path calculations will be the same for float and double, as the results will be converted to float between each propagation path step when copied to ppath_step.

As pointed out above, the most critical cases are when the path goes along the boundary of a grid cell. This situation is not common for arbitrary observation positions, but it is a standard case for 3D scattering calculations as the starting point for the calculations there is always a crossing point of the atmospheric grids. The solution to this problem is to introduce special treatment for such geometrical paths. For strictly vertical 2D and 3D paths, the latitude, and also longitude for 3D, of the start and end points shall be identical. Paths in 3D with an azimuth angle of 0°or 180° have a constant longitude; the paths are in the north-south plane, and this should also then be valid for the longitude value of the start and end positions of the path step.

The variables connected to different problems associated with the numerical inaccuracy and singularity of mathematical expressions are defined at the top of the file `ppath.cc`. The variables include the accepted tolerance when making asserts in internal functions that the given point is inside the specified grid cell. Another example is the latitude limit to use the special mathematical expressions needed for positions on the poles.

6.5.2 Propagation paths and grid positions

The grid positions are calculated on the same time as the path is determined. The main reason to this is that the grid positions make it possible to quickly determine inside which grid box the path step is found. Without the grid positions, each call of the functions would need a costly search to locate the starting position with respect to the grids. If you are not familiar with grid positions, it is recommended to read *ARTS Developer Guide*, Section 5 before you continue here.

The limited numerical accuracy requires some care when setting the grid positions. First of all, rounding errors can give a fractional distance < 0 or > 1 and this must be avoided. The function `gridpos_check_fd` was created for this purpose, and should be called for each grid position. This function just sets all values below 0 to 0 and all value above 1 to 1. In addition, the grid position for the end point of a path step (beside when there is an intersection with the ground) must have one fractional distance of exactly 0 or 1, but this is not ensured by `gridpos_check_fd` and for end points the function `gridpos_force_end_fd` shall also be called.

Some care is needed to determine in which grid range a path step is found. First of all, there exists an ambiguity for the fractional distance at the grid points. It can either be 0 or 1. In addition, if a position is exactly on top of a grid point, the observation direction determines the interesting grid range. As an help to resolve these question there is the function `gridpos2gridrange`. This function takes an argument describing the direction of the line-of-sight with respect to the grids. This argument shall be set to 1 if the viewing direction is towards higher indexes. The direction argument can be set with the following logical expressions, for the different combinations of atmospheric dimensionality and grid of interest:

1D-3D, pressure: $|\psi| \leq 90°$
2D, latitude: $\psi \geq 0°$
3D, latitude: $\omega \leq 90°$
3D, longitude: $\omega \geq 0°$

6.6 Some basic geometrical relationships for 1D and 2D

This section gives some expressions to determine positions along a propagation path when refraction is neglected. The expressions deal only with propagation path inside a plane, where the latitude angle is the angular distance from an arbitrary point. This means that the expressions given here can be directly applied for 1D and 2D. Some of the expression are also of interest for 3D. The ARTS method for making the calculation of concern is given inside parenthesis above each equation, if not stated explicitly. A part of a geometrical propagation path is shown in Figure 6.3.

The law of sines gives that the product must $r\sin(\psi)$ be constant along the propagation path:

$$p_c = r\sin(\psi) \tag{6.1}$$

where the absolute value is taken for 2D zenith angles as they can for such cases be negative. The propagation path constant, p_c, is determined by the position and line-of-sight of the sensor, a calculation done by the function `geometrical_ppc`. The constant equals also the radius of the tangent point of the path (that is found along an imaginary prolongation of the path behind the sensor if the viewing direction is upwards). The expressions below are based on p_c as the usage of a global constant for the path should decrease the sensitivity to numerical inaccuracies. If the calculations are based solely on the values for the neighboring point, a numerical inaccuracy can accumulate when going from one point to next. The propagation path constant is stored in the field `constant` of ppath and ppath_step.

The relationship between the distance along the path for an infinitesimal change in radius is here denoted as the geometrical factor, g. If refraction is neglected, valid expressions for the geometrical factor are

$$g = \frac{\mathrm{d}l}{\mathrm{d}r} = \frac{1}{\cos(\psi)} = \frac{1}{\sqrt{1-\sin^2(\psi)}} = \frac{r}{\sqrt{r^2-p_c^2}} \tag{6.2}$$

For the radiative transfer calculations, only the distance between the points, Δl, is of interest, but for the internal propagation path calculations the length from the tangent point (real or imaginary), l, is used. By integrating Equation 6.2, we get that (`geomppath_l_at_r`)

$$l(r) = \sqrt{r^2-p_c^2} \tag{6.3}$$

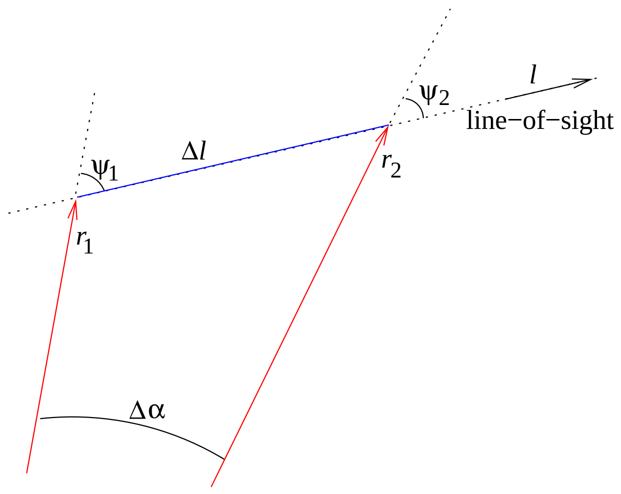

Figure 6.3: The radius (r) and zenith angle (ψ) for two points along the propagation path, and the distance along the path (Δl) and the latitude difference ($\Delta\alpha$) between these points.

As refraction is here neglected, the tangent point, the point of concern and the center of the coordinate system make up a right triangle and Equation 6.3 corresponds to the Pythagorean relation where p_c is the radius of the tangent point. The distance between two points (Δl) is obtained by taking the difference of Equation 6.3 for the two radii.

The radius for a given l is simply (`geomppath_r_at_l`)

$$r(l) = \sqrt{l^2 + p_c^2} \tag{6.4}$$

The radius for a given zenith angle is simply obtained by rearranging Equation 6.1 (`geomppath_r_at_za`)

$$r(\psi) = \frac{p_c}{sin(\psi)} \tag{6.5}$$

The zenith angle for a given radius is (`geomppath_za_at_r`)

$$\psi(r) = \begin{cases} 180 - \sin^{-1}(p_c/r) & \text{for} \quad 90^\circ < \psi_a \leq 180^\circ \\ \sin^{-1}(p_c/r) & \text{for} \quad 0^\circ \leq \psi_a \leq 90^\circ \\ -\sin^{-1}(p_c/r) & \text{for} \quad -90^\circ \leq \psi_a < 0^\circ \\ \sin^{-1}(p_c/r) - 180 & \text{for} \quad -180^\circ \leq \psi_a < -90^\circ \end{cases} \tag{6.6}$$

where ψ_a is any zenith angle valid for the path on the same side of the tangent point. For example, for a 1D case, the part of the path between the tangent point and the sensor has zenith angles $90^\circ < \psi_a \leq 180^\circ$.

The latitude for a point (`geomppath_lat_at_za`) is most easily determined by its zenith angle

$$\alpha(\psi) = \alpha_0 + \psi_0 - \psi \tag{6.7}$$

where ψ_0 and α_0 are the zenith angle and latitude of some other point of the path. Equation 6.7 is based on the fact that the quantities ψ_1, ψ_2 and $\Delta\alpha$ fulfill the relationship

$$\Delta\alpha = \psi_1 - \psi_2, \tag{6.8}$$

this independently of the sign of the zenith angles. The definitions used here result in that the absolute value of the zenith angle always decreases towards zero when following the path in the line-of-sight direction, that is, when going away from the sensor. It should then be remembered that the latitudes for 1D measures the angular distance to the sensor, and for 2D a positive zenith angle means observation towards higher latitudes.

The radius for a given latitude (`geomppath_r_at_lat`) is obtained by combining Equations 6.7 and 6.5.

6.7 Calculation of geometrical propagations paths

This section describes the calculation of geometrical propagation paths for different atmospheric dimensionalities. That is, the effect of refraction is neglected. These calculations are performed by the workspace method ppath_stepGeometric. This method, as all methods for propagation path steps, adjust automatically to the atmospheric dimensionality, but the actual calculations are performed a sub-function for each dimensionality.

6.7.1 1D

The core function for this case is `do_gridrange_1d`. The lowest and highest radius value along the path step is first determined. If the line-of-sight is upwards ($\| \psi | \leq 90°$), then the start point of the step gives the lowest radius, and the radius of the pressure surface above gives the highest value. In the case of a downwards line-of-sight, the lowest radius is either the tangent point, the pressure surface below or the surface. The needed quantities to describe the propagation path between the two found radii are calculated by the function `geompath_from_r1_to_r2`, that has the option to introduce more points to fulfill a length criterion between the path points. The mathematics of `geompath_from_r1_to_r2` are given by Equations 6.1–6.7.

6.7.2 2D

The definitions given in Sections 2.2 results in that for a 2D case the radius of a pressure surface varies linearly from one point of the latitude grid to next. This is the main additional problem to solve, compared to the 1D case. Figure 6.4 gives a schematic description of the problem at hand, which is handled by the internal function `psurface_crossing_2d`.

The law of sine gives the following relationship for the crossing point:

$$\frac{\sin \Theta_p}{r_0 + c\alpha} = \frac{\sin(\pi - \alpha - \Theta_p)}{r_p} \tag{6.9}$$

which can be re-written to

$$r_p \sin(\Theta_p) = (r_0 + c\alpha)(\sin \Theta_p \cos \alpha + \cos \Theta_p \sin \alpha) \tag{6.10}$$

This equation has no analytical solution. A first step to find an approximative solution is to note that α will be limited to relatively small values. For example, if it shall be possible for the angular distance α to reach the value of 3°, the vertical spacing between the pressure surfaces must be about 8 km, while it normally is below 2 km. For angles $\alpha \leq 3°$, the sine

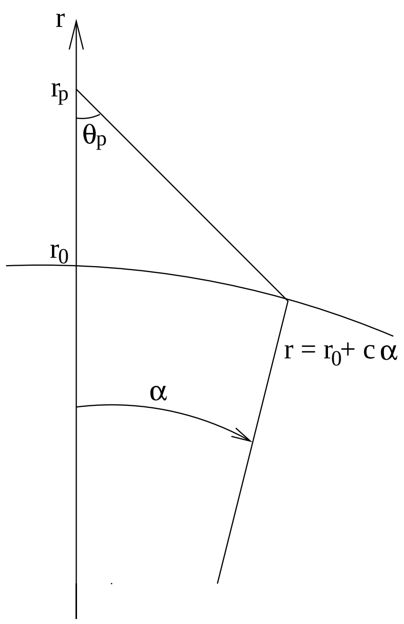

Figure 6.4: Quantities used to describe how to find the crossing between a geometrical propagation path and a tilted pressure surface. The angle α is the angular distance from a reference point on the path. The problem at hand is to find α for the crossing point. The radius of the pressure surface at $\alpha = 0$ is denoted as r_0. The tilt of the pressure surface is c.

and cosine terms can be replaced with the two first terms of their Taylor expansions with a relative accuracy of $< 4 \cdot 10^{-7}$. That is,

$$\begin{aligned}\cos\alpha &\approx 1 - \alpha^2/2\\ \sin\alpha &\approx \alpha - \alpha^3/6\end{aligned}$$

Equation 6.10 becomes with these replacements a polynomial equation of order 4:

$$\begin{aligned}0 &= p_0 + p_1\alpha + p_2\alpha^2 + p_3\alpha^3 + p_4\alpha^4 \qquad (6.11)\\ p_0 &= (r_0 - r_p)\sin\Theta_p\\ p_1 &= r_0\cos\Theta_p + c\sin\Theta_p\\ p_2 &= -(r_0\sin\Theta_p)/2 + c\cos\Theta_p\\ p_3 &= -(r_0\cos\Theta_p)/6 - (c\sin\Theta_p)/2\\ p_4 &= -(c\cos\Theta_p)/6\end{aligned}$$

This equation is solved numerically with the root finding algorithm implemented in the function `poly_root_solve`. Solutions of interest shall not be imaginary.

Geometrical 2D propagation path steps are determined by `do_gridcell_2d`. This function uses `psurface_crossing_2d` to calculate the latitude distance to a crossing of the pressure surface below and above the present path point. If the closest crossing point with the pressure surfaces is outside the latitude range of the grid cell, it is the crossing of the path with the end latitude (in the viewing direction) that is of interest (Figure 6.5).

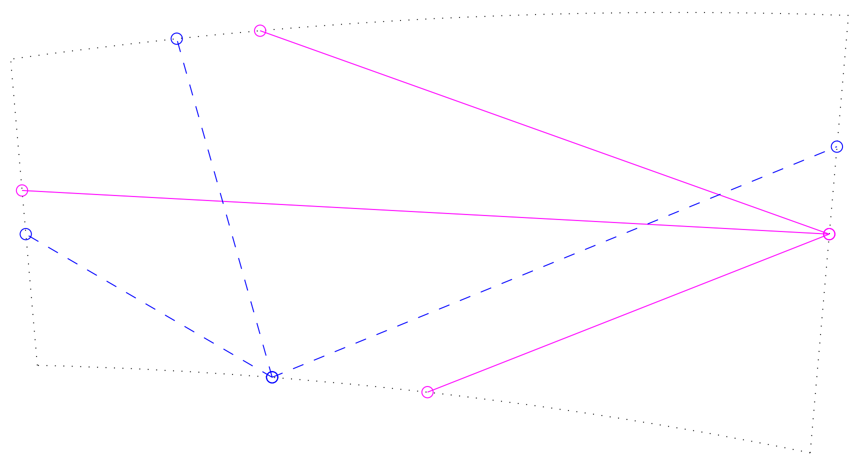

Figure 6.5: Example on propagation path steps starting from a latitude end face (solid lines), or the lower pressure surface (dashed lines), to all other grid cell faces. The distortion of the grid cell from cylinder segment is highly exaggerated compared to a real case. The relationship between vertical and horisontal size deviates also from normal real cases. Typical values for the vertical extension is around 500 m, while the horisontal length is normally $>$10 km.

6.7.3 3D

Geometrical 3D propagation path steps are determined by the function `do_gridcell_3d`. It was first tested to use different analytical expressions to calculate the length between a point and the crossing of some radius, latitude or longitude. However, the expressions found include the trigonometric functions and the squaring of radii, which resulted in a high sensitivity to the numerical inaccuracy. It was found that the numerical problems made the created algorithm impossible to use in practice. Equation 6.20 below is a reminiscence of that work. In addition, no simple solution to the problem of finding the crossing with a tilted 3D pressure surface using the analytical expressions was found.

A straightforward trail-and-error algorithm was then tested (Algorithm 5 and Figure 6.6). The main advantage of the algorithm is that a correction for the shift in position caused by the transformations back and fourth to a cartesian coordinate system can be applied. The correction term assures that the position is not changed for a step of zero length, and is not moved outside the grid cell due to the numerical problems. The algorithm was further found to be sufficiently fast to be accepted. A simple bisection search to find the length of the propagation path step is used. Both the position and the line-of-sight for the other end point of the path step are calculated using a transformation to cartesian coordinates. The cartesian coordinate system used here is defined as:

x-axis is along latitude 0°and longitude 0°

y-axis is along latitude +90°

z-axis is along latitude 0°and longitude +90°

Algorithm 5 The method applied in `do_gridcell_3d` to find the total length of the path step to be calculated. The symbol S signifies here conversion from cartesian to spherical coordinates (Equation 6.13).

```
calculate the spherical position $(x_0, y_0, z_0)$ and LOS vector $(\mathrm{d}x, \mathrm{d}y, \mathrm{d}z)$
calculate $(r_c, \alpha_c, \beta_c) = S(x_0, y_0, z_0) - (r_0, \alpha_0, \beta_0)$, the position correction term
set $l_{in} = 0$
set $l_{out} = 1$
**if** LOS is downwards **then**
    calculate length to the tangent point, $l_{tan}$
**else**
    set $l_{tan} = 99 \cdot 10^6$ m
**end if**
**while** $S(x_0 + l_{out}\mathrm{d}x, y_0 + l_{out}\mathrm{d}y, z_0 + l_{out}\mathrm{d}z) - (r_c, \alpha_c, \beta_c)$ is inside grid cell **do**
    **if** $l_{out} < l_{tan}$ and $10 l_{out} > l_{tan}$ **then**
        $l_{out} = l_{tan}$ (to assure that tangent point is included in search)
    **else**
        $l_{out} \leftarrow 10 * l_{out}$
    **end if**
**end while**
set $l_{end} = (l_{in} + l_{out})/2$
set accuracy flag to false
**while** accuracy flag is false **do**
    calculate $(r, \alpha, \beta) = S(x_0 + l_{end}\mathrm{d}x, y_0 + l_{end}\mathrm{d}y, z_0 + l_{end}\mathrm{d}z) - (r_c, \alpha_c, \beta_c)$
    **if** $(r, \alpha, \beta)$ is inside grid cell **then**
        $l_{in} = l_{end}$
    **else**
        $l_{out} = l_{end}$
    **end if**
    **if** $(l_{out} - l_{in})$ smaller than specified accuracy **then**
        set accuracy flag to true
    **else**
        $l_{end} = (l_{in} + l_{out})/2$
    **end if**
**end while**
$(r, \alpha, \beta) \leftarrow (r, \alpha, \beta) + (r_c, \alpha_c, \beta_c)$
```

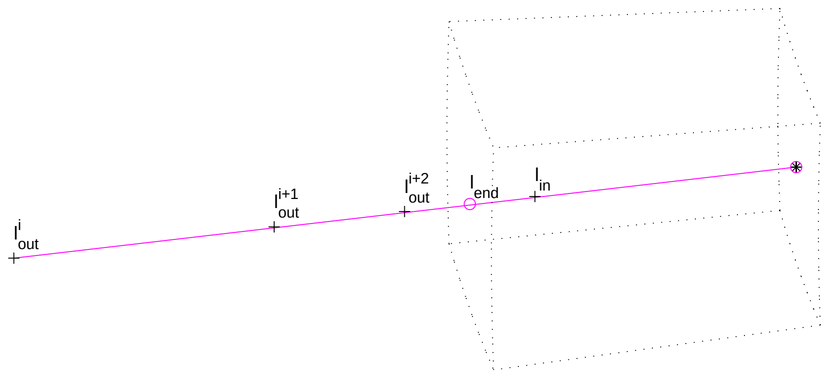

Figure 6.6: Schematic of Algorithm 5. The figure shows two iterations of the algorith to search for the total length of the path step. The asterisk $(*)$ gives the start point for the calculations and the circles $(\circ)$ are the final end points of the path step. The plus signs $(+)$ shows the position of the different lengths tested during the iterations.

This definition results in the following relationships between the spherical (r, α, β) and cartesian (x, y, z) coordinates

$$\begin{aligned} x &= r\cos(\alpha)\cos(\beta) \\ y &= r\sin(\alpha) \\ z &= r\cos(\alpha)\sin(\beta) \end{aligned} \tag{6.12}$$

and

$$\begin{aligned} r &= \sqrt{x^2+y^2+z^2} \\ \alpha &= \arcsin(y/r) \\ \beta &= \arctan(z/x) \qquad \text{(implemented by the atan2 function)} \end{aligned} \tag{6.13}$$

The functions performing these transformations are `sph2cart` and `cart2sph`.

The first step to transform a line-of-sight, given by the zenith (ψ) and the azimuth (ω) angle, to cartesian coordinates is to determine the corresponding vector with unit length in the spherical coordinate system:

$$\begin{bmatrix} \mathrm{d}r \\ \mathrm{d}\alpha \\ \mathrm{d}\beta \end{bmatrix} = \begin{bmatrix} \cos(\psi) \\ \sin(\psi)\cos(\omega)/r \\ \sin(\psi)\sin(\omega)/(r\cos(\alpha)) \end{bmatrix} \tag{6.14}$$

This vector is then translated to the cartesian coordinate system as

$$\begin{bmatrix} \mathrm{d}x \\ \mathrm{d}y \\ \mathrm{d}z \end{bmatrix} = \begin{bmatrix} \cos(\alpha)\cos(\beta) & -r\sin(\alpha)\cos(\beta) & -r\cos(\alpha)\sin(\beta) \\ \sin(\alpha) & r\cos(\alpha) & 0 \\ \cos(\alpha)\sin(\beta) & -r\sin(\alpha)\sin(\beta) & r\cos(\alpha)\cos(\beta) \end{bmatrix} \begin{bmatrix} \mathrm{d}r \\ \mathrm{d}\alpha \\ \mathrm{d}\beta \end{bmatrix} \tag{6.15}$$

Note that the radial terms (r) in Equations 6.14 and 6.15 cancel each other. These calculations are performed in `poslos2cart`. Special expressions must be used for positions at the north and south pole (see the code) as the azimuth angle has there a special definition (Section 3.7.2).

The cartesian position of a point along the geometrical path at a distance l is then simply

$$\begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix} = \begin{bmatrix} x_1 + l\mathrm{d}x \\ y_1 + l\mathrm{d}y \\ z_1 + l\mathrm{d}z \end{bmatrix} \tag{6.16}$$

The cartesian viewing vector $[\mathrm{d}x, \mathrm{d}y, \mathrm{d}z]^T$ is constant along a geometrical path. The new position is converted to spherical coordinates by Equation 6.13 and the new spherical viewing vector is calculated as

$$\begin{bmatrix} \mathrm{d}r \\ \mathrm{d}\alpha \\ \mathrm{d}\beta \end{bmatrix} = \begin{bmatrix} \cos(\alpha)\cos(\beta) & \sin(\alpha) & \cos(\alpha)\sin(\beta) \\ -\sin(\alpha)\cos(\beta)/r & \cos(\alpha)/r & -\sin(\alpha)\sin(\beta)/r \\ -\sin(\beta)/(r\cos(\alpha)) & 0 & \cos(\beta)/(r\cos(\alpha)) \end{bmatrix} \begin{bmatrix} \mathrm{d}x \\ \mathrm{d}y \\ \mathrm{d}z \end{bmatrix} \tag{6.17}$$

which is converted to a zenith and azimuth angle as

$$\begin{aligned} \psi &= \arccos(\mathrm{d}r) \\ \omega &= \arccos(r\mathrm{d}\alpha/\sin(\psi)), \qquad \text{for} \quad \mathrm{d}\beta >= 0 \\ \omega &= -\arccos(r\mathrm{d}\alpha/\sin(\psi)), \qquad \text{for} \quad \mathrm{d}\beta < 0 \end{aligned} \tag{6.18}$$

Special expressions must be used for positions at the north and south pole (see the code) as the azimuth angle has there a special definition (Section 3.7.2). These calculations are performed in `cart2poslos`.

For sensor positions outside the atmosphere, the calculations made in `ppath_start_stepping` involve the problem of finding the position where the path leaves the atmosphere. This position is found by an iterative search. The maximum radius of the uppermost pressure surface is taken as first guess for the radius of the exit point. The exit latitude and longitude for this radius is determined (as discussed below), and the radius for the top of the atmosphere for the found position is used as radius for next iteration. This procedure is repeated until the change from one iteration to next for both latitude and longitude is smaller than $1\cdot 10^{-6}$. The exit position for a given radius, r, is found by solving the following equation system:

$$\begin{aligned} r\cos(\alpha)\cos(\beta) &= x + l\mathrm{d}x \\ r\sin(\alpha) &= y + l\mathrm{d}y \\ r\cos(\alpha)\sin(\beta) &= z + l\mathrm{d}z \end{aligned} \tag{6.19}$$

where (x, y, z) is the position of the sensor, $(\mathrm{d}x, \mathrm{d}y, \mathrm{d}z)$ the sensor LOS, and l, α and β are the variables to be determined. The first step is to determine the distance l to the exit point, which is found by adding the square of all three equations:

$$r^2 = (x + l\mathrm{d}x)^2 + (y + l\mathrm{d}y)^2 + (z + l\mathrm{d}z)^2 \tag{6.20}$$

Once l is determined, the latitude and longitude are easily calculated by Equations 6.16 and 6.13. These calculations are implemented in the function `psurface_crossing_3d`. Similar expressions were derived to find the position for the crossing of a given latitude

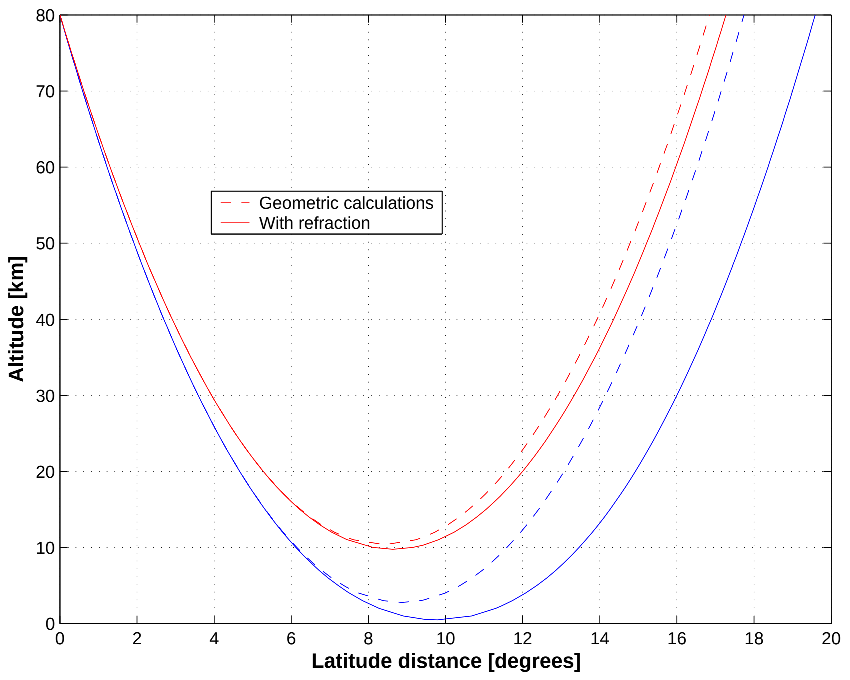

Figure 6.7: Comparison of propagation paths calculated geometrically and with refraction considered, for the same zenith angle of the sensor line-of-sight. The figure include two pair of paths, with refracted tangent altitude of about 0 and 10 km, respectively. The horisontal coordinate is the latitude distance from the point where the path exits the model atmosphere (at 80 km). The model atmosphere used had a spherical symmetry (that is, 1 D case, but the calculations were performed in 2D mode).

or longitude but those expressions were removed from the code as they are not used with present algorithms.[1]

6.8 Refraction with simple Euler scheme

Refraction affects the radiative transfer in several ways. The distance through a layer of a fixed vertical thickness will be changed, and for a limb sounding observation the tangent point is moved both vertically and horizontally. If the atmosphere is assumed to be horizontally stratified (1D), a horizontal displacement is of no importance but for 2D and 3D calculations this effect must be considered. For limb sounding and a fixed zenith angle, the tangent point is moved downwards compared to the pure geometrical case (Figure 6.7), resulting in that inclusion of refraction in general gives higher intensities. However, the propagation path is still symmetric around tangent and surface points.

The refraction causes a bending of the path, which gives a deviation from the geometrical approximation of propagation along a straight line. The bending of the path is obtained

[1]The expressions mentioned can be extracted from the function `gridcell_crossing_3d` in ARTS version 1-1-440.

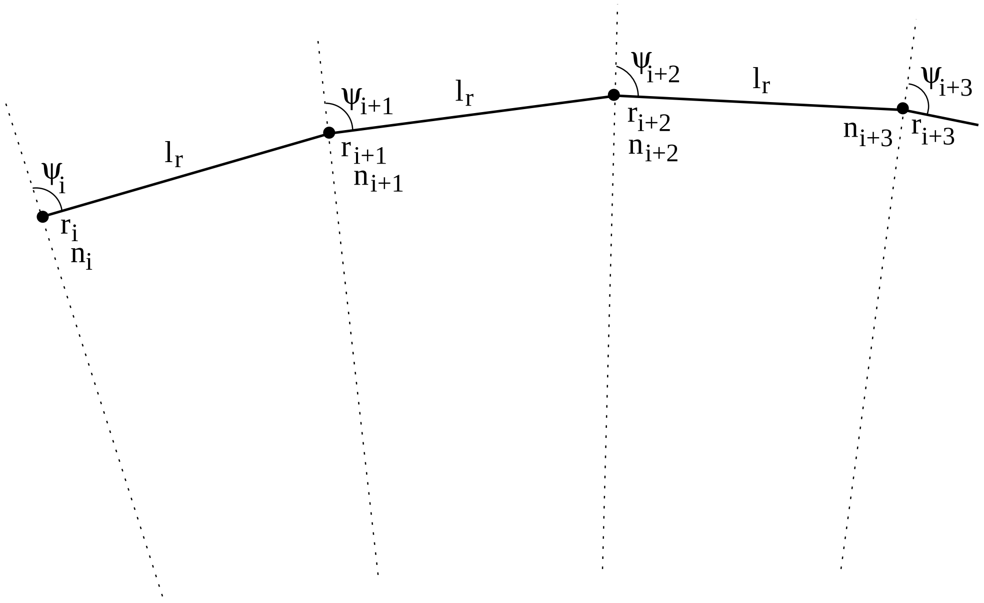

Figure 6.8: Schematic of the Euler ray tracing scheme. The ray tracing step length is l_r.

by the relationship

$$\frac{\mathrm{d}x}{\mathrm{d}l} = \frac{1}{n}\left(\frac{\partial n}{\partial y}\right)_x \tag{6.21}$$

where x is the direction of propagation, l the distance along the path, n the refractive index[2], and y is the coordinate perpendicular to the path. See further Section 9.4 in *Rodgers* [2000].

The workspace method ppath_stepRefractionEuler takes refraction into consideration by probably the most simple (from the viewpoint of implementation) algorithm possible. This does not mean that it is the best way to consider refraction, it is rather inefficient regarding computational burden, and if the step length for the ray tracing (see below) is made very small, the result can be completely wrong due to numerical problems.

The approach taken in ppath_stepRefractionEuler is to take a geometrical ray tracing step from the present point of the path (and in the direction of present line-of-sight). Refraction is considered only when the line-of-sight at the new point is determined (Figure 6.8). The found line-of-sight is used to calculate the next ray tracing step etc. This can be seen as an Euler solution to the differential problem given by Equation 6.21. The main difference between handling 1D, 2D or 3D cases is how the line-of-sight for the new point is corrected to compensate for the bending due to refraction. The calculation of propagation paths including the effect of refraction is often denoted as ray tracing.

The length of the calculation steps is set by the generic input `lraytrace`. This length shall not be confused with the final distance between the points that define the path, which is controlled by the generic input `lmax`. The path is first determined in steps of `lraytrace`. The found ray tracing points are then used for an interpolation to create a path step defined

[2]The refractive index is here assumed to have no imaginary part

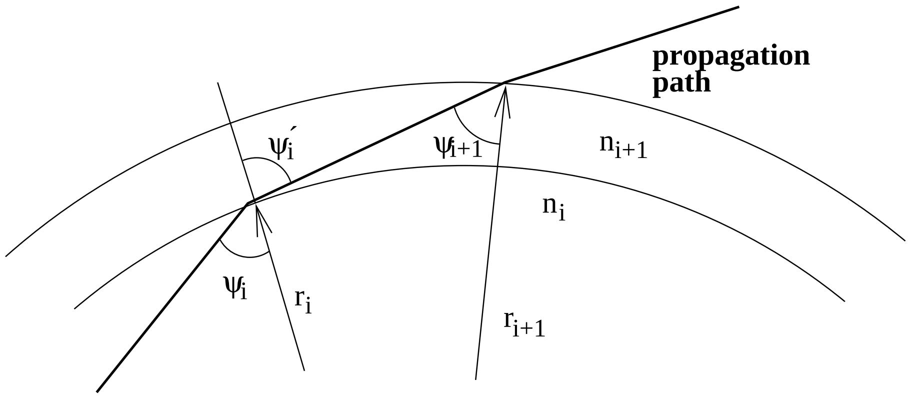

Figure 6.9: Geometry to derive Snell's law for a spherical atmosphere.

exactly as for geometrical calculations. The normal situation is that the ray tracing step length is considerably shorter than the final spacing between the path points. Suitable values for `lraytrace` have not yet been investigated in detail, but for limb sounding values in around 1–10 km should be appropriate. Shorter ray tracing steps (down to a level where rounding errors will start to have an impact) will of course give a propagation path more accurately determined, but on the cost of more time consuming calculations.

6.8.1 1D

When determining the propagation path through the atmosphere geometrical optics can be applied because the change of the refractive index over a wavelength can be neglected. Applying Snell's law to the geometry shown in Figure 6.9 gives

$$n_i \sin(\psi_i) = n_{i+1} \sin(\psi_{i'}) \tag{6.22}$$

Using the same figure, the law of sines gives the relationship

$$\frac{\sin(\psi_{i+1})}{r_i} = \frac{\sin(180^\circ - \psi'_{i+1})}{r_{i+1}} = \frac{\sin(\psi_{i'})}{r_{i+1}} \tag{6.23}$$

By combining the two equations above, the Snell's law for a spherical atmosphere (that is, 1D cases) is derived [e.g. *Kyle*, 1991; *Balluch and Lary*, 1997]:

$$p_c = r_i n_i \sin(\psi_i) = r_{i+1} n_{i+1} \sin(\psi_{i+1}) \tag{6.24}$$

where c is a constant. With other words, the Snell's law for spherical atmospheres states that the product of n, r and $\sin(\psi)$ is constant along the propagation path. It is noteworthy that with $n = 1$, Equations 6.1 and 6.24 are identical.

The Snell's law for a spherical atmosphere makes it very easy to determine the zenith angle of the path for a given radius. A rearrangement of Equation 6.24 gives

$$\psi = \arcsin(rn/p_c) \tag{6.25}$$

This relationship makes it possible to handle refraction for 1D without calculating any gradients of the refractive index, which is needed for 2D and 3D. These calculations are implemented in the function `raytrace_1d_linear_euler`. Figure 6.10 shows the vertical variation of the refractive index.

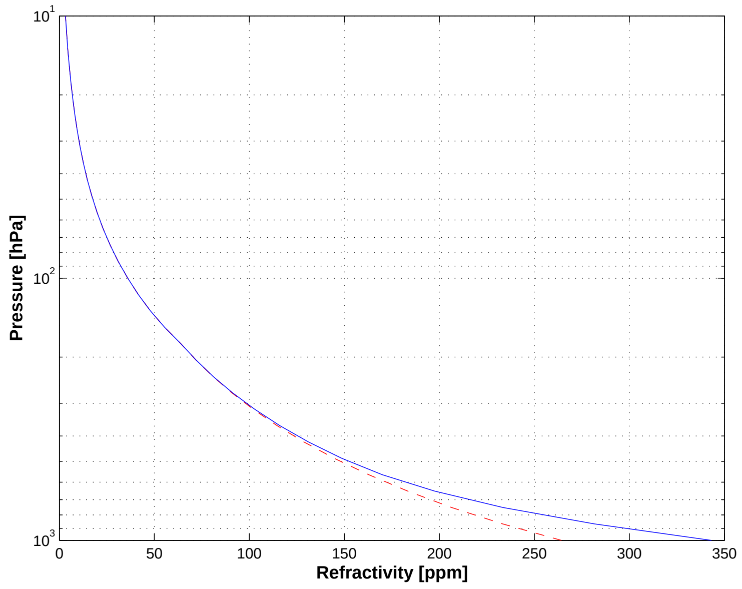

Figure 6.10: Vertical variation of refractivity $(n-1)\cdot 10^6$. Calculated for a mid-latitude summer climatology (FASCODE), where the dashed line is for a completely dry atmosphere, and the solid line includes also contribution from water vapour.

6.8.2 2D

Equation 6.21 expressed in polar coordinates is [*Rodgers*, 2000, Eq. 9.30]

$$\frac{\mathrm{d}(\alpha+\psi)}{\mathrm{d}l} = -\frac{\sin\psi}{n}\left(\frac{\partial n}{\partial r}\right)_{\alpha} + \frac{\cos\psi}{nr}\left(\frac{\partial n}{\partial \alpha}\right)_{r} \tag{6.26}$$

If the gradients are zero (corresponding to the geometrical case) we find that the sum of the zenith angle and the latitude is constant along a 2D geometrical path, which is also made clear by Equation 6.7. The geometrical zenith angle at ray tracing point $i+1$ is accordingly $\psi_{i+1} = \psi_i - (\alpha_{i+1} - \alpha_i)$. If then also the refraction is considered, we get the following expression:

$$\psi_{i+1} = \psi_i - (\alpha_{i+1} - \alpha_i) + \frac{l_g}{n_i}\left[-\sin\psi_i\left(\frac{\partial n}{\partial r}\right)_{\alpha_i} + \frac{\cos\psi_i}{r_i}\left(\frac{\partial n}{\partial \alpha}\right)_{r_i}\right] \tag{6.27}$$

The gradients of the refractive index for 2D are calculated by the function `refr_gradients_2d`. This function returns the gradients as the change of the refractive index over 1 m. The conversion for the latitude gradient corresponds to the $1/r$ term found in Equation 6.27, and this term is accordingly left out in `raytrace_2d_linear_euler`, which is the function of this section.

The radial and latitudinal gradients of the refractive index are calculated in pure numerical way, by shifting the position slightly from the position of concern. Figures 6.11 and 6.12 show example on gradients of the refractive index.

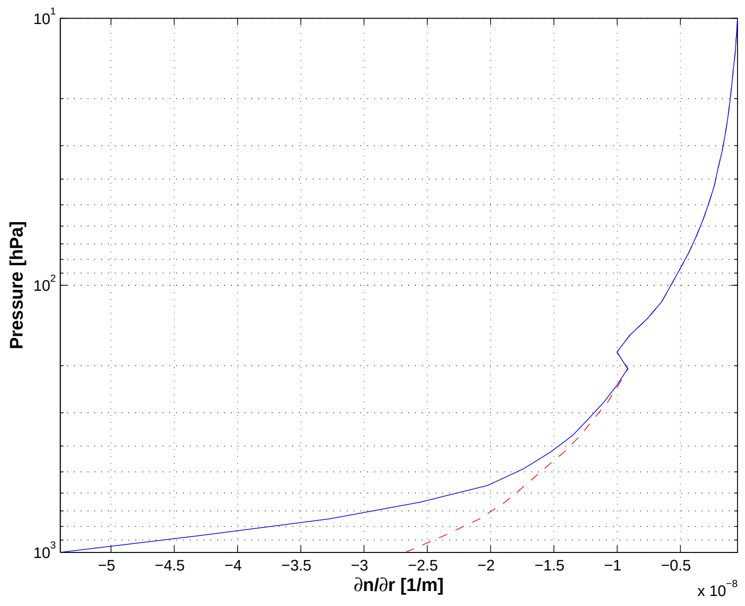

Figure 6.11: Vertical gradient of the refractive index. Calculated for a mid-latitude summer climatology (FASCODE), where the dashed line is for a completely dry atmosphere, and the solid line includes also contribution from water vapour.

6.8.3 3D

For 3D, the geomtrical expressions are used to calculate the geometrical zenith and azimuth angles at the end of the ray tracing step. Following the methodology for 2D, the geometrical zenith and azimuth angles are then corrected to incorporate the influence of refraction. The zenith angle is calculated as

$$\begin{aligned} \psi_{i+1} &= \psi_g - \frac{l_g \sin\psi_i}{n_i}\left(\frac{\partial n}{\partial r}\right)_{(\alpha_i,\beta_i)} + \\ &\quad + \frac{l_g \cos\psi_i}{r_i n_i}\left[\cos\omega_i \left(\frac{\partial n}{\partial \alpha}\right)_{(r_i,\beta_i)} + \frac{\sin\omega_i}{\cos\alpha_i}\left(\frac{\partial n}{\partial \beta}\right)_{(r_i,\alpha_i)}\right] \end{aligned} \tag{6.28}$$

where ψ_g is the zenith angle obtained from the geometrical expressions. In similar manner, the geometrical azimuth angle, ω_g, is corrected as

$$\omega_{i+1} = \omega_g + \frac{l_g \sin\psi_i}{r_i n_i}\left[-\sin\omega_i \left(\frac{\partial n}{\partial \alpha}\right)_{(r_i,\beta_i)} + \frac{\cos\omega_i}{\cos\alpha_i}\left(\frac{\partial n}{\partial \beta}\right)_{(r_i,\alpha_i)}\right] \tag{6.29}$$

This expression, slightly modified, is found in `raytrace_3d_linear_euler`. The terms of Equation 6.29 missing in that function, are part of `refr_gradients_3d` to convert the gradients to the same unit. The longitude gradient is converted to the unit [1/m] by multiplication with the term $1/(r\cos\alpha)$.

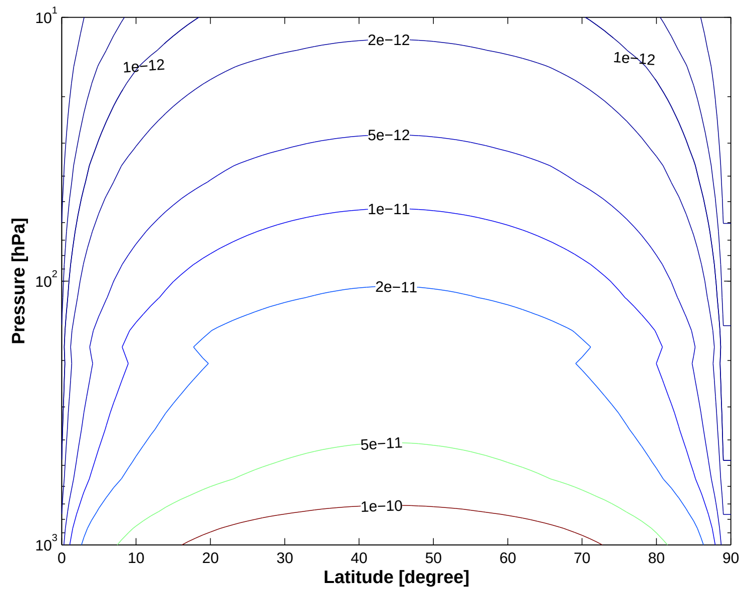

Figure 6.12: Latitude gradient of the refractive index due to varying radius of the geoid. The gradient is given as the change in refractive index over 1 m, which allows direct comparison with the values in Figure 6.11e. The wet atmosphere from Figure 6.11 was used for all latitudes, and the the plotted gradient is only caused by the fact that the radius of the geoid is not constant. The gradient is positive on the southern hemisphere (shown), and negative on the northern hemisphere.

Chapter 7

Atmospheric winds

Atmospheric transport is not considered by ARTS, but winds can still be of importance due to the Doppler effect they can cause. This effect is most significant at high altitudes where the line shape is narrow, and a frequency shift of absorption and emission is most easily discerned. The effect depends also on the angle between the wind vector and the line-of-sight.

7.1 Definitions and practicalities

7.1.1 Needed input

The workspace variables to specify winds are wind_u_field, wind_v_field and wind_w_field, below denoted as v_u, v_v and v_w, respectively. The user need to set all these three variables. It is allowed for all three wind components to set the variable to be empty, which is short-hand for saying that the wind is zero throughout the atmosphere. Otherwise, the size of the variable is required to match the atmospheric grids. The variable wind_u_field is not considered for 1D and 2D.

No further input is required, a Doppler shift is added as soon as any of the winds is non-zero (with exceptions discussed in Section 7.1.4).

7.1.2 Definition for 3D atmospheres

The definition of the winds should be the standard one for 3D atmospheres:

v_u The zonal wind, where a positive wind is defined as air moving from west to east (i.e. towards higher longitudes).

v_v The meridional wind, where a positive wind is defined as air moving from south to north (i.e. towards higher latitudes).

v_w The vertical wind, where a positive wind is defined as air moving upwards.

History

110622 Created by Patrick Eriksson.

7.1.3 Definition for 1D and 2D atmospheres

As these atmospheric dimensionalities do not have an azimuthal dimensions, it only makes sense to operate with a single horizontal wind component and wind_v_field is used for this purpose, wind_u_field is ignored. The v_v-component is assumed to be totally aligned azimuthally with the line-of-sight. That is, the horizontal wind is radial, centred around the sensor. For 1D a positive wind signifies air moving away from the sensor. For 2D, a positive wind is air moving against higher latitudes (as for 3D). The vertical component is defined as for 3D.

7.1.4 Limitations

A single Doppler shift is applied for all frequencies. That is, broadband calculations are treated in an approximative manner. So far, non-zero winds require that the absorption is calculated "on-the-fly" (Section 4). The scattering algorithms do not apply a Doppler shift, i.e. the winds are ignored.

7.2 Equations

The main equations for deriving the Doppler shift from the winds are given in this section. The total wind, v, is

$$v = \sqrt{v_u^2 + v_v^2 + v_w^2}. \tag{7.1}$$

The zenith angle of the wind direction is

$$\psi_v = \arccos(v_w/v), \tag{7.2}$$

and the azimuth angle (for 3D only) is

$$\omega_v = \arctan(v_u/v_v). \qquad \text{(implemented by the atan2 function)} \tag{7.3}$$

For 1D and 2D, ω_v is set to 0° if $v_v \geq 0$, otherwise to 180°.

The cosine of the angle between the wind vector and the line-of-sight is

$$\cos\gamma = \cos\psi_v \cos\psi_l + \sin\psi_v \sin\psi_l \cos(\omega_v - \omega_l), \tag{7.4}$$

where ψ_l and ω_l are the angles of the line-of-sight. In this equation, the azimuth angle ω_l is treated to be 0° for 1D and 2D with zenith angles above 0, and 180°for 2D with negative zenith angles.

Finally, as the winds do not reach relativistic values, the Doppler shift can be calculated as

$$\Delta\nu = \frac{-v\nu_0 \cos\gamma}{c}, \tag{7.5}$$

where ν_0 is the rest frequency and c is the speed of light. In practise, ν_0 is taken as the mean of the end values of f_grid. The sign of the expression can be understood by the fact that if the wind and line-of-sight are aligned and going in the same direction, the air movement is directly away from the sensor, that gives a shift towards lower frequencies. As workspace variable $\Delta\nu$ is denoted as doppler_shift, and is applied on f_grid when extracting the (non-polarised) absorption (where a single $\Delta\nu$ is applied on the complete f_grid, as also mentioned in Section 7.1.4).

Chapter 8

Batch calculations

Quite often one wants to repeat a large number of calculations with only a few variables changed. Examples of such cases are to perform 1D calculations for a number of atmospheric states taken from some atmospheric model, generate a set of spectra to create a training database for regression based inversions or perform a numeric inversion error analysis. For such calculations it is inefficient to perform the calculations by calling ARTS repeatedly. For example, as data must be imported for each call even if the data are identical between the cases. Cases such as the ones described above are here denoted commonly as batch calculations.

8.1 Batch calculations of measurement vector y

Batch calculations of measurement vectors are done by the WSM ybatchCalc, which takes three inputs, the indices ybatch_start and ybatch_n, and the agenda ybatch_calc_agenda.

The method ybatchCalc will do ybatch_n calculations, starting at index ybatch_start. It will run the agenda ybatch_calc_agenda for each case individually. The agenda gets an input ybatch_index, which it should use to extract the right input data from one or more arrays or matrices of input. Execution of ybatch_calc_agenda must result in a new spectrum vector, y, most likely by a call of yCalc.

The WSV ybatch_start is set to a default of 0 in `general.arts`, so you do not have to set this variable unless you want to start somewhere in the middle of your batch input data.

The next section gives some examples of what could be put inside the ybatch_calc_agenda.

History

090330	Description of ybatch_start added by Stefan Buehler.
070726	Copy-edited by Stefan Buehler.
070618	Updated by Oliver Lemke.
040916	Created by Patrick Eriksson.

8.2 Control file examples

The WSV Extract can be used to extract an element from an Array, a row from a Matrix, a Matrix from a Tensor3, and so on. The selection is always done on the first dimension. So, for a Tensor3 as input, it copies the page with the given index from the input Tensor3 variable to the output Matrix.

The idea here is to store the batch cases in tensors or arrays with one dimension extra compared to corresponding workspace variables. For example, a set of t_field (which is of type Tensor3) is stored as Tensor4.

If the dimension is the same for all batch cases, tensors are appropriate. If the dimensions vary (for example you have a different number of vertical levels for each case), then arrays are appropriate.

In the following 1D example, five atmospheric scenarios have been put into the three first loaded files, and a random vector of zenith angles is found in the last file. The batch calculations are then performed as:

```
ReadXML( tensor4_1, "batch_t_field.xml" )
ReadXML( tensor4_2, "batch_z_field.xml" )
ReadXML( tensor5_1, "batch_vmr_field.xml" )
ReadXML( tensor3_1, "batch_za.xml" )
IndexSet( ybatch_n, 5 )

AgendaSet ( ybatch_calc_agenda ){
  Print( ybatch_index, 0 )
  Extract( t_field,    tensor4_1, ybatch_index )
  Extract( z_field,    tensor4_2, ybatch_index )
  Extract( vmr_field,  tensor5_1, ybatch_index )
  Extract( sensor_los, tensor3_1, ybatch_index )

  yCalc
}

ybatchCalc

WriteXML( "ascii", ybatch )
```

If you then want to repeat the calculations, for example with another propagation path step length (e.g. 25 km), it is sufficient to add the lines:

```
AgendaSet( ppath_step_agenda ){
   ppath_stepGeometric( ppath_step, atmosphere_dim, p_grid,
                        lat_grid, lon_grid,
                        z_field, r_geoid, z_surface,
                        25e3 )
}

ybatchCalc

WriteXML( "ascii", ybatch, "ybatch_run2.xml" )
```

Chapter 9

Description of clouds

9.1 Introduction

In the Earth's atmosphere we find liquid water clouds consisting of approximately spherical water droplets and cirrus clouds consisting of ice particles of diverse shapes and sizes. We also find different kinds of aerosols. In order to take into account this variety, the model allows to define several *particle types*. A particle type is either a specified particle or a specified particle distribution, for example a particle ensemble following a gamma size distribution. The particles can be completely randomly oriented, azimuthally randomly oriented or arbitrarily oriented. For each particle type being a part of the modeled cloud field, a data file containing the single scattering properties ($\langle \mathbf{K}_i \rangle$, $\langle \mathbf{a}_i \rangle$, and $\langle \mathbf{Z}_i \rangle$), and the appropriate particle number density field is required. The particle number density fields are stored as `GField3`, including the field stored in a three-dimensional tensor and also the appropriate atmospheric grids (pressure, latitude and longitude grid). For each grid point in the cloud box the single scattering properties are averaged using the particle number density fields. In the scattering database the single scattering properties are not always stored in the same coordinate system. For instance for randomly oriented particles it makes sense to store the single scattering properties in the so-called scattering frame in order to reduce memory requirements. The following section describes in detail the `SingleScatteringData` class. The consequent section describes how to realize different kinds of size distributions in the ARTS frame by defining appropriate particle number density fields.

9.2 Single scattering properties

9.2.1 Coordinate systems: The laboratory frame and the scattering frame

For radiative transfer calculations we need a coordinate system to describe the direction of propagation. For this purpose we use the laboratory frame, which is also shown in *ARTS Theory*, Figure 5.1. The z-axis corresponds to the local zenith direction and the x-axis points towards the north-pole. The propagation direction is described by the local zenith angle θ

History

050913	Created and written by Claudia Emde

and the local azimuth angle ϕ. This coordinate system is the most appropriate frame to describe the propagation direction and the polarization state of the radiation. However, in order to describe scattering of radiation by a particle or a particle ensemble, it makes sense to define another coordinate system taking into consideration the symmetries of the particle or the scattering medium, as one gets much simpler expressions for the single scattering properties. For macroscopically isotropic and mirror-symmetric scattering media it is convenient to use the scattering frame, in which the incidence direction is parallel to the z-axis and the x-axis coincides with the scattering plane, that is, the plane through the unit vectors $\hat{\mathbf{n}}^{\mathrm{inc}}$ and $\hat{\mathbf{n}}^{\mathrm{sca}}$. The scattering frame is illustrated in Figure 9.1. For symmetry reasons the single scattering properties defined with respect to the scattering frame can only depend on the scattering angle Θ,

$$\Theta = \arccos(\hat{\mathbf{n}}^{\mathrm{inc}} \cdot \hat{\mathbf{n}}^{\mathrm{sca}}), \tag{9.1}$$

between the incident and the scattering direction.

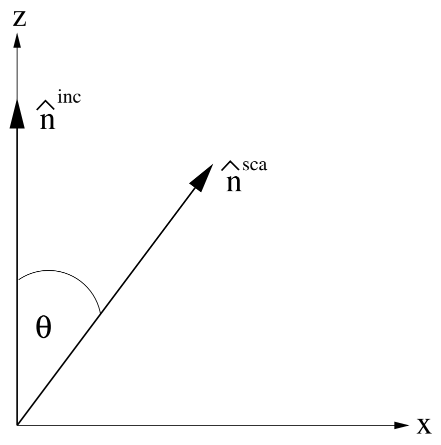

Figure 9.1: Illustration of the scattering frame. The z-axis coincides with the incident direction $\hat{\mathbf{n}}^{\mathrm{inc}}$. The scattering angle Θ is the angle between $\hat{\mathbf{n}}^{\mathrm{inc}}$ and $\hat{\mathbf{n}}^{\mathrm{sca}}$.

9.2.2 Scattering datafile structure

The single scattering properties are pre-calculated, for example by using the T-matrix code by *Mishchenko et al.* [2002], and stored in data-files. Different methods for the calculation of single scattering properties are reviewed in *Emde* [2005].

The format of the scattering database allows space reduction due to symmetry for certain special cases, e.g. random orientation or horizontal alignment. The file format is XML. The data is stored in a class called `SingleScatteringData`, which resides in the files `optproperties.h`. The class consists of the following fields (compare also Table 9.2.2):

- *enum* `ptype`: An attribute which contains information about the data type, which is the classification of the kind of hydrometeor species (randomly oriented, general case ...). This attribute is needed in the radiative transfer function to be able to extract the physical phase matrix, the physical extinction matrix and the physical absorption vector from the data.

Possible values of ptype are:

`PARTICLE_TYPE_GENERAL` = 10
`PARTICLE_TYPE_MACROS_ISO` = 20
`PARTICLE_TYPE_HORIZ_AL` = 30

A more detailed description of the different cases is given below.

- *String* `description`: Here the particle type should be specified explicitly. We can have the case randomly oriented particles, but furthermore we also have to specify the exact particle properties (i.e. size and shape distribution). This can be a longer text describing how the scattering properties were generated. It should be formatted for direct printout to screen or file.
- *Vector* `f_grid`: Frequency grid [Unit: Hz].
- *Vector* `T_grid`: Temperature grid [Unit: K].
- *Vector* `za_grid`:
  1. `p10, p30`: Zenith angle grid (Range: $0.0° \leq$ za $\leq 180.0°$).
  2. `p20`: Scattering angle grid (Range: $0.0° \leq$ za $\leq 180.0°$).
- *Vector* `aa_grid`: Azimuth angle grid.
  1. `p10`: Range: $-180.0° \leq$ aa $\leq 180.0°$
  2. `p20`: Not needed, since optical properties depend only on scattering angle (dummy grid).
  3. `p30`: Only half of the grid is required (Range: $0.0° \leq$ aa $\leq 180.0°$)

  The angular grids have to satisfy the following conditions:

  - They have to be equidistant.
  - The value of the data must be the same for the first and the last grid-point. This condition is required for the integration routine.
  - If we only have to store a part of the grid, for example `za_grid` only from 0°to 90°, these two values (0°, 90°) must be grid-points.
- *Tensor7* `pha_mat_data`: Phase matrix data $\langle \mathbf{Z} \rangle$ [Unit: m^2]. The dimensions of the data array are:

  `[frequency temperature za_sca aa_sca za_inc aa_inc matrix_element]`

  The order of matrix elements depends on the chosen case. For most cases we do not need all matrix elements (see description of cases below).

Table 9.1: Structure of single scattering data files

Symbol	Type	Dimensions	Description
	enum		specification of particle type
	String		short description of particle type
ν	Vector	(ν)	frequency grid
T	Vector	(T)	temperature grid
ψ	Vector	(ψ)	zenith angle grid
ω	Vector	(ω)	azimuth angle grid
$\langle \mathbf{Z} \rangle$	Tensor7	$(\nu, T, \psi, \omega, \psi', \omega', i)$	phase matrix
$\langle \mathbf{K} \rangle$	Tensor5	$(\nu, T, \psi, \omega, i)$	extinction matrix
$\langle \mathbf{a} \rangle$	Tensor5	$(\nu, T, \psi, \omega, i)$	absorption vector

- *Tensor5* `ext_mat_data`: Extinction matrix data $\langle \mathbf{K} \rangle$ [Unit: m^2]. The dimensions are:

  `[frequency temperature za_inc aa_inc matrix_element]`

  Again, the order of matrix elements depends on the chosen case.

- *Tensor5* `abs_vec_data`: Absorption vector data $\langle \mathbf{a} \rangle$ [Unit: m^2].

  The absorption vector is also precalculated. It could be calculated from extinction matrix and phase matrix. But this calculation takes long computation time, as it requires an angular integration over the phase matrix. For the cases with symmetries (e.g., random orientation) the data files will not become too large even if we store additionally the absorption vector. The dimensions are:

  `[frequency temperature za_inc aa_inc vector_element]`

9.2.3 Definition of particle types

Macroscopically isotropic and mirror-symmetric scattering media (p20)

For macroscopically isotropic and mirror-symmetric scattering media (totally randomly oriented particles) the optical properties are calculated in the so-called scattering frame as shown in Figure 9.1. In this coordinate system the z-axis corresponds to the incident direction and the xz-plane coincides with the scattering plane. Using this frame only the scattering angle, which is the angle between incident and scattered direction is needed. Furthermore the number of matrix elements of both matrices, phase matrix and extinction matrix, can be reduced (see *Mishchenko et al.* [2002], p.90). To calculate the particle optical properties it is convenient to use Mishchenko's T-matrix code for randomly oriented particles [*Mishchenko and Travis*, 1998] which returns the averaged phase matrix and extinction matrix. The only drawback is that the single scattering data has to be transformed from the particle frame representation to the laboratory frame representation. These transformations are described in the appendix of *Emde* [2005].

Only six elements of the transformed phase matrix, which is commonly called scattering matrix **F**, are different. Therefore the size of `pha_mat_data` is:

```
[N_f N_T N_za_sca 1 1 1 6]
```

The order of the matrix elements is as follows: *F11, F12, F22, F33, F34, F44*
The extinction matrix is in this case diagonal and independent of direction and polarization. That means that we need to store only one element for each frequency. Hence the size of `ext_mat_data` is

```
[N_f N_T 1 1 1]
```

The absorption vector is also direction and polarization independent. Therefore the size of `abs_vec_data` for this case is the same as `ext_mat_data`:

```
[N_f N_T 1 1 1]
```

Horizontally aligned plates and columns (p30) For particle distributions of horizontally aligned plates and columns that are oriented randomly in the azimuth the angular dimension can be reduced by one, if we rotate the coordinate system appropriately. For this case we use the T-matrix code for single particles in fixed orientation and average phase matrix and extinction matrix manually like in the general case.

The phase matrix (and also extinction matrix and absorption vector) become independent of the incident azimuth angle in this frame. Furthermore, regarding the symmetry of this case, it can be shown that for the scattered directions we need only half of the angular grids, as the two halves must contain the same data. `pha_mat_data` therefore has the following size:

```
[N_f N_T N_za_sca N_aa_sca N_za_inc/2+1 1 16]
```

We store `za_sca` for all grid points from 0°to 180°, `aa_sca` from 0°to 180°, and `za_inc` from 0°to 90°. This means that the zenith angle grid has to include 90°as grid-point. The order of the matrix elements is the same as in the general case. For this case it can be shown that the extinction matrix has only three elements *Kjj*, *K12(=K21)*, and *K34(=-K43)*. Because of azimuthal symmetry, it can not depend on the azimuth angle. Hence the size of `ext_mat_data` is

```
[N_f N_T N_za/2+1 1 3]
```

The absorption coefficient vector has only two elements *a1* and *a2*. This means that the size of `abs_vec_data` is

```
[N_f N_T N_za/2+1 1 2]
```

General case (p10) If there are no symmetries at all we have to store all 16 elements of the phase matrix. The average phase matrix has to be generated from all individual phase matrices of the particles in the distribution outside ARTS. The individual phase matrices are calculated using Mishchenko's T-matrix code for single particles in fixed orientation [*Mishchenko*, 2000]. We have to store all elements for all angles in the grids. The size of `pha_mat_data` is therefore:

```
[N_f N_T N_za_sca N_aa_sca N_za_inc N_aa_inc 16]
```

The matrix elements have to be stored in the following order: *Z11, Z12, Z13, Z14, Z21, Z22, ...* Seven extinction matrix elements are independent (cp. *Mishchenko et al.* [2002], p.55). The elements being equal for single particles should still be the equal for a distribution as we get the total extinction just by adding. Here we need only the incoming grids, so the size of ext_mat_data is:

```
    [N_f N_T N_za_inc N_aa_inc 7]
```

The absorption vector in general has four components (cp. Equation (2.186) in *Mishchenko et al.* [2002]). The size of abs_vec_data is accordingly:

```
    [N_f N_T N_za_inc N_aa_inc 4]
```

Generating single scattering properties It is very convenient to use the Python module PyARTS, which has been developed especially for ARTS and which is freely available at `http://www.sat.ltu.se/arts/tools/`. This module can be used to generate single scattering properties for horizontally aligned as well as for randomly oriented particles in the ARTS data-file-format. PyARTS has been developed by C. Davis, who has implemented the Monte Carlo scattering algorithm in ARTS (see *ARTS Theory*, Section 6). The ATMLAB package includes functions to generate single scattering properties for spherical particles (Mie-Theory).

9.3 Representation of the particle size distribution

The particle size has an important impact on the scattering and absorption properties of cloud particles as shown for instance in [*Emde et al.*, 2004]. Clouds contain a whole range of different particle sizes, which can be described by a size distribution giving the number of particles per unit volume per unit radius interval as a function of radius. It is most convenient to parameterize the size distribution by analytical functions, because in this case optical properties can be calculated much faster than for arbitrary size distributions. The T-matrix code for randomly oriented particles includes several types of analytical size distributions, e.g., the gamma distribution or the log-normal distribution. This section presents the size distribution parameterizations, which were used for the ARTS simulations included in this thesis.

9.3.1 Mono-disperse particle distribution

The most simple assumption is, that all particles in the cloud have the same size. In order to study scattering effects like polarization or the influence of particle shape, it makes sense to use this most simple assumption, because one can exclude effects resulting from the particle size distribution itself.

Along with the single scattering properties we need the particle number density field, which specifies the number of particles per cubic meter at each grid point, for ARTS scattering simulations. For a given IMC and mono-disperse particles the particle number density n^p is simply

$$n^p(IMC, r) = \frac{IMC}{m} = \frac{IMC}{V\rho} = \frac{3}{4\pi}\frac{IMC}{\rho r^3}, \tag{9.2}$$

where m is the mass of a particle, r is its equal volume sphere radius, ρ is its density, and V is its volume.

9.3.2 Gamma size distribution

A commonly used distribution for radiative transfer modeling in cirrus clouds is the *gamma distribution*

$$n(r) = ar^{\alpha} \exp(-br). \tag{9.3}$$

The dimensionless parameter α describes the width of the distribution. The other two parameters can be linked to the effective radius R_{eff} and the ice mass content IMC as follows:

$$b = \frac{\alpha+3}{R_{eff}}, \tag{9.4}$$

$$a = \frac{IMC}{4/3\pi\rho b^{-(\alpha+4)}\Gamma[\alpha+4]}, \tag{9.5}$$

where ρ is the density of the scattering medium and Γ is the gamma function. For cirrus clouds ρ corresponds to the bulk density of ice, which is approximately 917 kg/m^3.

Generally, the effective radius R_{eff} is defined as the average radius weighted by the particle cross-section

$$R_{eff} = \frac{1}{\langle A \rangle} \int_{r_{min}}^{r_{max}} A(r) r n(r) \mathrm{d}r, \tag{9.6}$$

where A is the area of the geometric projection of a particle. The minimal and maximal particle sizes in the distribution are given by r_{min} and r_{max} respectively. In the case of spherical particles $A = \pi r^2$. The average area of the geometric projection per particle $\langle A \rangle$ is given by

$$\langle A \rangle = \frac{\int_{r_{min}}^{r_{max}} A(r) n(r) \mathrm{d}r}{\int_{r_{min}}^{r_{max}} n(r) \mathrm{d}r}. \tag{9.7}$$

The question is how well a gamma distribution can represent the true particle size distribution in radiative transfer calculations. This question is investigated by Evans et al. [1998]. The authors come to the conclusion that a gamma distribution represents the distribution of realistic clouds quite well, provided that the parameters R_{eff}, IMC and α are chosen correctly. They show that setting $\alpha = 1$ and calculating only R_{eff} gave an agreement within 15% in 90% of the considered measurements obtained during the First ISCCP Regional Experiment (FIRE). Therefore, for all calculations including gamma size distributions for ice particles, $\alpha = 1$ was assumed.

The particle number density for size distributions is obtained by integration of the distribution function over all sizes:

$$n^p(IMC, R_{eff}) = \int_0^{\infty} n(r) \mathrm{d}r \tag{9.8}$$

$$= \int_0^{\infty} ar^{\alpha} \exp(-br) \mathrm{d}r = a\frac{\Gamma(\alpha+1)}{b^{\alpha+1}}. \tag{9.9}$$

After setting α = 1, inserting Equation 9.5 and some simple algebra we obtain

$$n^p(IMC, R_{eff}) = \frac{2}{\pi} \frac{IMC}{\rho R_{eff}^3}. \tag{9.10}$$

Comparing Equation 9.2 and 9.10, we see that the particle number density for monodisperse particles with a particle size of R is smaller than the particle number density for gamma distributed particles with $R_{eff} = R$. The reason is that in the gamma distribution most particles are smaller than R_{eff}.

9.3.3 Ice particle size parameterization by McFarquhar and Heymsfield

A more realistic parameterization of tropical cirrus ice crystal size distributions was derived by *McFarquhar and Heymsfield* [1997], who derived the size distribution as a function of temperature and IMC. The parameterization was made based on observations during the Central Equatorial Pacific Experiment (CEPEX). Smaller ice crystals with an equal volume sphere radius of less than 50 μm are parametrized as a sum of first-order gamma functions:

$$n(r) = \frac{12 \cdot IMC_{<50}\alpha^5_{<50} r}{\pi\rho\Gamma(5)} \exp(-2\alpha_{<50} r), \tag{9.11}$$

where $\alpha_{<50}$ is a parameter of the distribution, and $IMC_{<50}$ is the mass of all crystals smaller than 50 μm in the observed size distribution. Large ice crystals are represented better by a log-normal function

$$\begin{aligned} n(r) = &\frac{3 \cdot IMC_{>50}}{\pi^{3/2}\rho\sqrt{2}\exp(3\mu_{>50} + (9/2)\sigma^2_{>50}) r \sigma_{>50} r_0^3} \\ &\cdot \exp\left[-\frac{1}{2}\left(\frac{\log\frac{2r}{r_0} - \mu_{>50}}{\sigma_{>50}}\right)^2\right], \end{aligned} \tag{9.12}$$

where $IMC_{>50}$ is the mass of all ice crystals greater than 50 μm in the observed size distribution, r_0 = 1 μm is a parameter used to ensure that the equation does not depend on the choice of unit for r, $\sigma_{>50}$ is the geometric standard deviation of the distribution, and $\mu_{>50}$ is the location of the mode of the log-normal distribution. The fitted parameters of the distribution can be looked up in the article by *McFarquhar and Heymsfield* [1997]. The particle number density field is obtained by numerical integration over a discrete set of size bins. This parameterization of particle size has been implemented in the PyARTS package, which was introduced in Section 9.2.2. Using PyARTS one can calculate the size distributions, the corresponding single scattering properties and the particle number density fields for given IMC and temperature.

9.4 Implementation

The workspace methods related to the description of clouds in ARTS are implemented in the file `m_cloudbox.cc`. Work space methods related to the optical properties of the clouds are implemented in the file `m_optproperties.cc`. The coordinate system transformations described above reside in the file `optproperties.cc`.

9.4.1 Work space methods and variables

The following controlfile section illustrates how a simple cloud can be included in an ARTS calculation.

First we have to define the cloudbox region, i.e. the region where scattering objects are found. To do this we use the method cloudboxSetManuallyAltitude:

```
cloudboxSetManuallyAltitude( cloudbox_on, cloudbox_limits,
                             atmosphere_dim, p_grid,
                             lat_grid, lon_grid,
                              8000, 120000,
                              0, 0, 0, 0 )
```

If we want to do a simulation for a cirrus cloud at an altitude from 9 to 11 km the cloudbox limits can be set to 8 and 12 km. The latitude and longitude limits are set to an arbitrary value for a 1D calculation. For 3D calculations they are also needed. Alternatively one can use the method cloudboxSetManually, where one has to provide pressure instead of altitude limits.

Now we have to specify the cloud particles inside the scattering region:

```
# Initialisation
ParticleTypeInit
# Only one particle type is added in this example
ParticleTypeAdd( scat_data_raw, pnd_field_raw,
                 atmosphere_dim, f_grid, p_grid,
                 lat_grid, lon_grid, cloudbox_limits,
                 "ssd_sphere_50um_p20.xml",
                 "pnd_sphere_50um_p20.xml" )
```

In the workspace method ParticleTypeAdd the single scattering properties for one particle type are read. The generic input `filename_scat_data` must be set to the filename of a datafile including scattering data (class `SingleScatteringData`) in xml-format. The generic input `filename_pnd_field` must contain the filename of the corresponding particle number density field in xml-format (class `GField3`). If the cloud is composed of several different particle types ParticleTypeAdd can be used repeatedly for all particle types, for instance one could add to the randomly oriented spherical particles above horizontally aligned cylindrical particles:

```
ParticleTypeAdd( scat_data_raw, pnd_field_raw,
                 atmosphere_dim, f_grid, p_grid,
                 lat_grid, lon_grid, cloudbox_limits,
                 "ssd_cylinder_30um_p30.xml",
                 "pnd_cylinder_30um_p30.xml" )
```

Alternatively it is possible to use the method ParticleTypeAddAll, which is convenient to generate a size distribution using several size bins. In this case one needs to define one particle type for each size bin. For many size bins the control file becomes very lengthy if one uses ParticleTypeAdd repeatedly. ParticleTypeAddAll requires as input an array of string including all filenames of the single scattering data files and the variable pnd_field_raw which includes the particle number density fields for all particle types. Using this function, one has to make sure that the order of the filenames containing the single scattering data corresponds to the order of the particle number density fields in pnd_field_raw. After reading the data the workspace variable pnd_field is calculated using pnd_fieldCalc:

```
# Calculate the particle number density field
pnd_fieldCalc
```

The definition of the single scattering data along with the corresponding particle number density fields is common in both scattering modules, the DOIT module described in Chapter 10 and the Monte Carlo module described in *ARTS Theory*, Chapter 6.

Chapter 10

Scattering - DOIT module

The Discrete Ordinate ITerative (DOIT) method is one of the scattering algorithms in ARTS. Besides the DOIT method a backward Monte Carlo scheme has been implemented (see *ARTS Theory*, Section 6). The DOIT method is unique because a discrete ordinate iterative method is used to solve the scattering problem in a spherical atmosphere. Although the DOIT module is implemented for 1D and 3D atmospheres, it is strongly recommended to use it only for 1D, because the Monte Carlo module (*ARTS Theory*, Chapter 6) is much more appropriate for 3D calulations. More appropriate in the sense that it is much more efficient. A literature review about scattering models for the microwave region, which is presented in *Emde and Sreerekha* [2004], shows that former implementations of discrete ordinate schemes are only applicable for (1D-)plane-parallel or 3D-cartesian atmospheres. All of these algorithms can not be used for the simulation of limb radiances. A description of the DOIT method, similar to what is presented in this chapter, has been published in *Emde et al.* [2004] and in *Emde* [2005].

10.1 The discrete ordinate iterative method

10.1.1 Radiation field

The Stokes vector depends on the position in the cloud box and on the propagation direction specified by the zenith angle (ψ) and the azimuth angle (ω). All these dimensions are discretized inside the model; five numerical grids are required to represent the radiation field $\mathcal{I}$:

$$\begin{aligned} \vec{P} &= \{P_1, P_2, ..., P_{N_P}\}, \\ \vec{\alpha} &= \{\alpha_1, \alpha_2, ..., \alpha_{N_\alpha}\}, \\ \vec{\beta} &= \{\beta_1, \beta_2, ..., \beta_{N_\beta}\}, \end{aligned} \tag{10.1}$$

History

020601	Created and written by Claudia Emde
050223	Rewritten by Claudia Emde, mostly taken from Chapter 4 of Claudia's PhD thesis
050929	Included technical part, example contol file

$$\begin{aligned}\vec{\psi} &= \{\psi_1, \psi_2, ..., \psi_{N_\psi}\},\\ \vec{\omega} &= \{\omega_1, \omega_2, ..., \omega_{N_\omega}\}.\end{aligned}$$

Here $\vec{P}$ is the pressure grid, $\vec{\alpha}$ is the latitude grid and $\vec{\beta}$ is the longitude grid. The radiation field is a set of Stokes vectors ($N_P \times N_\alpha \times N_\beta \times N_\psi \times N_\omega$ elements) for all combinations of positions and directions:

$$\begin{aligned}\mathcal{I} &= \{\mathbf{I}_1(P_1, \alpha_1, \beta_1, \psi_1, \omega_1), \mathbf{I}_2(P_2, \alpha_1, \beta_1, \psi_1, \omega_1), ...,\\ &\mathbf{I}_{N_P \times N_\alpha \times N_\beta \times N_\psi \times N_\omega}(P_{N_P}, \alpha_{N_\alpha}, \beta_{N_\beta}, \psi_{N_\psi}, \omega_{N_\omega})\}. \qquad (10.2)\end{aligned}$$

In the following we will use the notation

$$\mathcal{I} = \{\mathbf{I}_{ijklm}\} \qquad \begin{aligned} i &= 1 \dots N_P\\ j &= 1 \dots N_\alpha\\ k &= 1 \dots N_\beta.\\ l &= 1 \dots N_\psi\\ m &= 1 \dots N_\omega\end{aligned} \qquad (10.3)$$

10.1.2 Vector radiative transfer equation solution

Figure 10.1 shows a schematic of the iterative method, which is applied to solve the vector radiative transfer equation (compare *ARTS Theory*, Equation 5.35)

$$\begin{aligned}\frac{\mathrm{d}\mathbf{I}(\hat{\mathbf{n}}, \nu, T)}{\mathrm{d}s} = &-\langle \mathbf{K}(\hat{\mathbf{n}}, \nu, T)\rangle \mathbf{I}(\hat{\mathbf{n}}, \nu, T) + \langle \mathbf{a}(\hat{\mathbf{n}}, \nu, T)\rangle B(\nu, T) &(10.4)\\ &+ \int_{4\pi} \mathrm{d}\hat{\mathbf{n}}' \langle \mathbf{Z}(\hat{\mathbf{n}}, \hat{\mathbf{n}}', \nu, T)\rangle \mathbf{I}(\hat{\mathbf{n}}', \nu, T), &(10.5)\end{aligned}$$

where $\mathbf{I}$ is the specific intensity vector, $\langle \mathbf{K}\rangle$ is the ensemble-averaged extinction matrix, $\langle \mathbf{a}\rangle$ is the ensemble-averaged absorption vector, B is the Planck function and $\langle \mathbf{Z}\rangle$ is the ensemble-averaged phase matrix. Furthermore ν is the frequency of the radiation, T is the temperature, $\mathrm{d}s$ is a path-length-element of the propagation path and $\hat{\mathbf{n}}$ the propagation direction. The vector radiative transfer equation is explained in more detail in *ARTS Theory*, Section 5.4.

The *first guess field*

$$\mathcal{I}^{(0)} = \left\{\mathbf{I}^{(0)}_{ijklm}\right\}, \qquad (10.6)$$

is partly determined by the boundary condition given by the radiation coming from the clear sky part of the atmosphere traveling into the cloud box. Inside the cloud box an arbitrary field can be chosen as a first guess. In order to minimize the number of iterations it should be as close as possible to the solution field.

The next step is to solve the scattering integrals

$$\left\langle \mathbf{S}^{(0)}_{ijklm}\right\rangle = \int_{4\pi} \mathrm{d}\hat{\mathbf{n}}' \langle \mathbf{Z}_{ijklm}\rangle \mathbf{I}^{(0)}_{ijklm}, \qquad (10.7)$$

using the first guess field, which is now stored in a variable reserved for the *old radiation field*. For the integration we use equidistant angular grids in order to save computation time (cf. Section 10.3.1). The radiation field, which is generally defined on finer angular grids

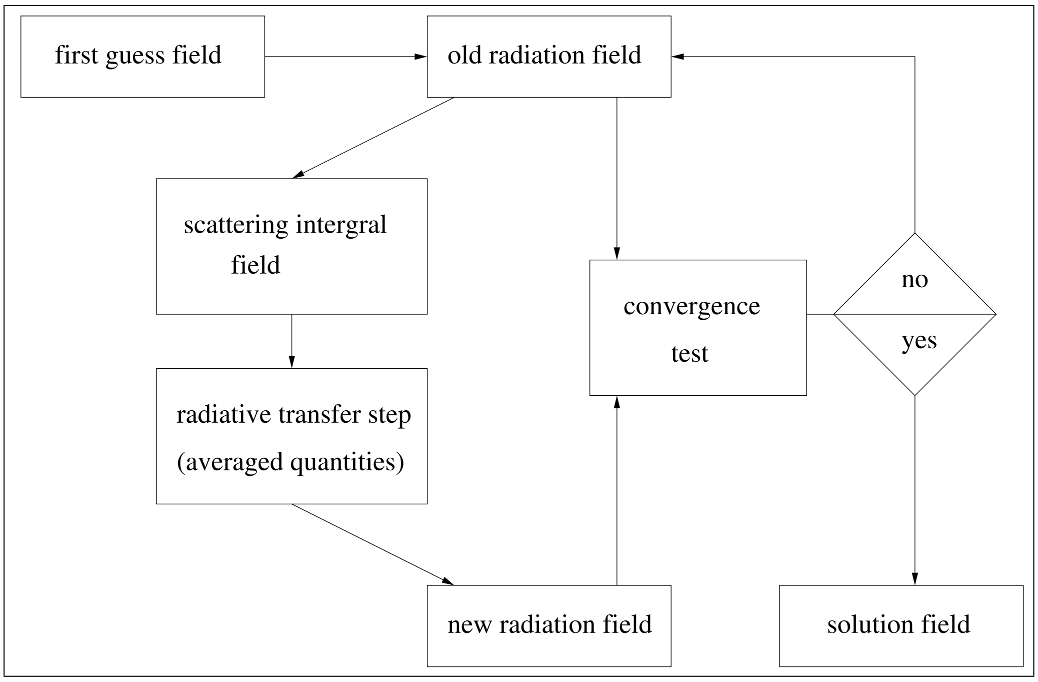

Figure 10.1: Schematic of the iterative method to solve the VRTE in the cloud box.

$(\vec{\omega}, \vec{\psi})$, is interpolated on the equidistant angular grids. The integration is performed over all incident directions $\hat{\mathbf{n}}'$ for each propagation direction $\hat{\mathbf{n}}$. The evaluation of the scattering integral is done for all grid points inside the cloud box. The obtained integrals are interpolated on $\vec{\omega}$ and $\vec{\psi}$. The result is the first guess *scattering integral field* $\mathcal{S}^0$:

$$\mathcal{S}^{(0)} = \left\{ \left\langle \mathbf{S}^{(0)}_{ijklm} \right\rangle \right\}. \tag{10.8}$$

Figure 10.2 shows a propagation path step from a grid point $\mathbf{P} = (P_i, \alpha_j, \beta_k)$ into direction $\hat{\mathbf{n}} = (\psi_l, \omega_m)$. The radiation arriving at $\mathbf{P}$ from the direction $\hat{\mathbf{n}}'$ is obtained by solving the linear differential equation:

$$\frac{\mathrm{d}\mathbf{I}^{(1)}}{\mathrm{d}s} = -\overline{\langle\mathbf{K}\rangle}\mathbf{I}^{(1)} + \overline{\langle\mathbf{a}\rangle}\,\bar{B} + \overline{\langle\mathbf{S}^{(0)}\rangle}, \tag{10.9}$$

where $\overline{\langle\mathbf{K}\rangle}$, $\overline{\langle\mathbf{a}\rangle}$, $\bar{B}$ and $\overline{\langle\mathbf{S}^{(0)}\rangle}$ are *averaged quantities*. This equation can be solved analytically for constant coefficients. Multi-linear interpolation gives the quantities $\mathbf{K}', \mathbf{a}', \mathbf{S}'$ and T' at the intersection point $\mathbf{P}'$. To calculate the radiative transfer from $\mathbf{P}'$ towards $\mathbf{P}$ all quantities are approximated by taking the averages between the values at $\mathbf{P}'$ and $\mathbf{P}$. The average value of the temperature is used to get the averaged Planck function $\bar{B}$.

The solution of Equation 10.9 is found analytically using a matrix exponential approach:

$$\mathbf{I}^{(1)} = e^{-\overline{\langle\mathbf{K}\rangle}s}\mathbf{I}^{(0)} + \left(\mathbf{I} - e^{-\overline{\langle\mathbf{K}\rangle}s}\right)\overline{\langle\mathbf{K}\rangle}^{-1}\left(\overline{\langle\mathbf{a}\rangle}\,\bar{B} + \overline{\langle\mathbf{S}^{(0)}\rangle}\right), \tag{10.10}$$

where $\mathbf{I}$ denotes the identity matrix and $\mathbf{I}^{(0)}$ the initial Stokes vector. The *radiative transfer step* from $\mathbf{P}'$ to $\mathbf{P}$ is calculated, therefore $\mathbf{I}^{(0)}$ is the incoming radiation at $\mathbf{P}'$ into direction

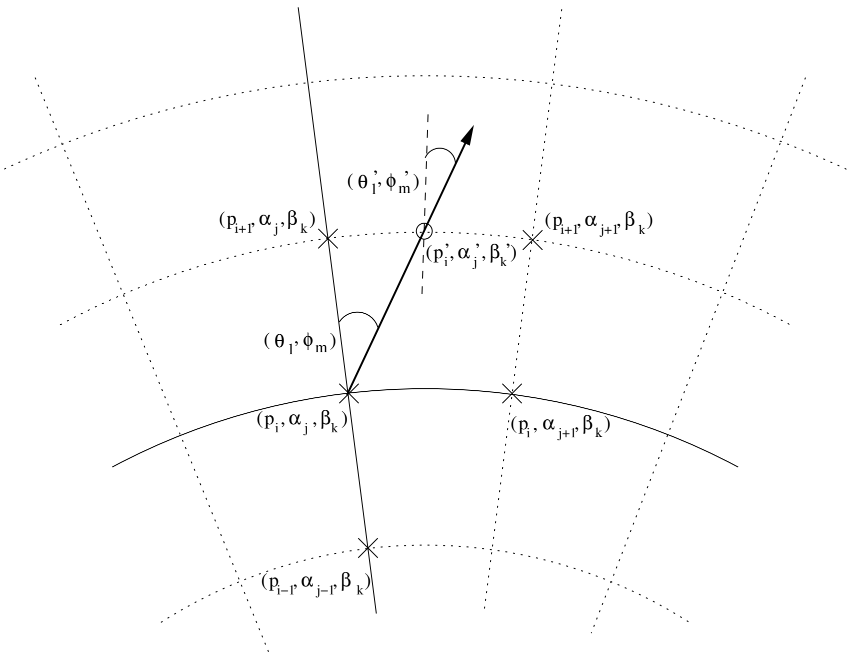

Figure 10.2: Path from a grid point ((P_i, α_j, β_k) - ($\times$)) to the intersection point ($(P'_i, \alpha'_j, \beta'_k)$ - ($\circ$)) with the next grid cell boundary. Viewing direction is specified by (ψ_l, ω_m) at ($\times$) or (ψ'_l, ω'_m) at ($\circ$).

(ψ'_l, ω'_m), which is the first guess field interpolated on $\mathbf{P}'$. This radiative transfer step calculation is done for all points inside the cloud box in all directions. The resulting set of Stokes vectors ($\mathbf{I}^{(1)}$ for all points in all directions) is the first order iteration field $\mathcal{I}^{(1)}$:

$$\mathcal{I}^{(1)} = \left\{ \mathbf{I}^{(1)}_{ijklm} \right\}. \tag{10.11}$$

The first order iteration field is stored in a variable reserved for the *new radiation field*.

In the *convergence test* the *new radiation field* is compared to the *old radiation field*. For the difference field, the absolute values of all Stokes vector elements for all cloud box positions are calculated. If one of the differences is larger than a requested accuracy limit, the convergence test is not fulfilled. The user can define different convergence limits for the different Stokes components.

If the convergence test is not fulfilled, the first order iteration field is copied to the variable holding the *old radiation field*, and is then used to evaluate again the scattering integral at all cloud box points:

$$\left\langle \mathbf{S}^{(1)}_{ijklm} \right\rangle = \int_{4\pi} \mathrm{d}\hat{\mathbf{n}}' \, \langle \mathbf{Z} \rangle \, \mathbf{I}^{(1)}_{ijklm}. \tag{10.12}$$

The second order iteration field

$$\mathcal{I}^{(2)} = \left\{ \mathbf{I}^{(2)}_{ijklm} \right\}, \tag{10.13}$$

is obtained by solving

$$\frac{\mathrm{d}\mathbf{I}^{(2)}}{\mathrm{d}s} = -\overline{\langle \mathbf{K} \rangle} \mathbf{I}^{(2)} + \overline{\langle \mathbf{a} \rangle} \, \bar{B} + \overline{\langle \mathbf{S}^{(1)} \rangle}, \tag{10.14}$$

for all cloud box points in all directions. This equation contains already the averaged values and is valid for specified positions and directions.

As long as the convergence test is not fulfilled the scattering integral fields and higher order iteration fields are calculated alternately.

We can formulate a differential equation for the n-th order iteration field. The scattering integrals are given by

$$\left\langle \mathbf{S}^{(n-1)}_{ijklm} \right\rangle = \int_{4\pi} \mathrm{d}\hat{\mathbf{n}}' \left\langle \mathbf{Z} \right\rangle \mathbf{I}^{(n-1)}_{ijklm}, \tag{10.15}$$

and the differential equation for a specified grid point into a specified direction is

$$\frac{\mathrm{d}\mathbf{I}^{(n)}}{\mathrm{d}s} = -\overline{\langle \mathbf{K} \rangle}\mathbf{I}^{(n)} + \overline{\langle \mathbf{a} \rangle}\,\bar{B} + \overline{\langle \mathbf{S}^{(n-1)} \rangle}. \tag{10.16}$$

Thus the *n-th order iteration field*

$$\mathcal{I}^{(n)} = \left\{ \mathbf{I}^{(n)}_{ijklm} \right\}, \tag{10.17}$$

is given by

$$\mathbf{I}^{(n)} = e^{-\overline{\langle \mathbf{K} \rangle}s} + \cdot\mathbf{I}^{(n-1)}(\mathbf{I} - e^{-\overline{\langle \mathbf{K} \rangle}s})\overline{\langle \mathbf{K} \rangle}^{-1}(\overline{\langle \mathbf{a} \rangle}\,\bar{B} + \overline{\langle \mathbf{S}^{(n-1)} \rangle}), \tag{10.18}$$

for all cloud box points and all directions defined in the numerical grids.

If the convergence test

$$\left| \mathbf{I}^{(N)}_{ijklm}\left(P_i, \alpha_j, \beta_k, \psi_l, \omega_m\right) - \mathbf{I}^{(N-1)}_{ijklm}\left(P_i, \alpha_j, \beta_k, \psi_l, \omega_m\right) \right| < \epsilon, \tag{10.19}$$

is fulfilled, a solution to the vector radiative transfer equation has been found:

$$\mathcal{I}^{(N)} = \left\{ \mathbf{I}^{(N)}_{ijklm} \right\}. \tag{10.20}$$

10.1.3 Scalar radiative transfer equation solution

In analogy to the *scattering integral* vector field the scalar scattering integral field is obtained:

$$\left\langle S^{(0)}_{ijklm} \right\rangle = \int_{4\pi} \mathrm{d}\hat{\mathbf{n}}' \left\langle Z_{11} \right\rangle I^{(0)}_{ijklm}. \tag{10.21}$$

The *scalar radiative transfer* equation (compare *ARTS Theory*, 5.44) with a fixed scattering integral is

$$\frac{\mathrm{d}I^{(1)}}{\mathrm{d}s} = -\left\langle K_{11} \right\rangle I^{(1)} + \left\langle a_1 \right\rangle B + \left\langle S^{(0)} \right\rangle. \tag{10.22}$$

Assuming constant coefficients this equation is solved analytically after averaging extinction coefficients, absorption coefficients, scattering vectors and the temperature. The averaging procedure is done analogously to the procedure described for solving the VRTE. The solution of the averaged differential equation is

$$I^{(1)} = I^{(0)}e^{-\overline{\langle K_{11} \rangle}s} + \frac{\overline{\langle a_1 \rangle}\,\bar{B} + \overline{\langle S^{(0)} \rangle}}{\overline{\langle K_{11} \rangle}}\left(1 - e^{-\overline{\langle K_{11} \rangle}s}\right), \tag{10.23}$$

where $I^{(0)}$ is obtained by interpolating the initial field, and $\overline{\langle K_{11}\rangle}$, $\overline{\langle a_1\rangle}$, $\bar{B}$ and $\overline{\langle S^{(0)}\rangle}$ are the averaged values for the extinction coefficient, the absorption coefficient, the Planck function and the scattering integral respectively. Applying this equation leads to the first iteration scalar intensity field, consisting of the intensities $I^{(1)}$ at all points in the cloud box for all directions.

As the solution to the vector radiative transfer equation, the solution to the scalar radiative transfer equation is found numerically by the same iterative method. The convergence test for the scalar equation compares the values of the calculated intensities of two successive iteration fields.

10.1.4 Single scattering approximation

The DOIT method uses the single scattering approximation, which means that for one propagation path step the optical depth is assumed to be much less than one so that multiple-scattering can be neglected along this propagation path step. It is possible to choose a rather coarse grid inside the cloud box. The user can define a limit for the maximum propagation path step length. If a propagation path step from one grid cell to the intersection point with the next grid cell boundary is greater than this value, the path step is divided in several steps such that all steps are less than the maximum value. The user has to make sure that the optical depth due to cloud particles for one propagation path sub-step is is sufficiently small to assume single scattering. The maximum optical depth due to ice particles is

$$\tau_{max} = \langle \mathbf{K}^p \rangle \cdot \Delta s, \tag{10.24}$$

where Δs is the length of a propagation path step. In all simulations presented in *Emde* [2005], $\tau_{max} \ll 0.01$ is assumed. This threshold value is also used in *Czekala* [1999]. The radiative transfer calculation is done along the propagation path through one grid cell. All coefficients of the VRTE are interpolated linearly on the propagation path points.

10.2 Sequential update

In the previous Section 10.1 the iterative solution method for the VRTE has been described. For each grid point inside the cloud box the intersection point with the next grid cell boundary is determined in each viewing direction. After that, all the quantities involved in the VRTE are interpolated onto this intersection point. As described in the sections above, the intensity field of the previous iteration is taken to obtain the Stokes vector at the intersection point. Suppose that there are N pressure levels inside the cloud box. If the radiation field is updated taking into account for each grid point only the adjacent grid cells, at least N-1 iterations are required until the scattering effect from the lower-most pressure level has propagated throughout the cloud box up to the uppermost pressure level. From these considerations, it follows, that the number of iterations depends on the number of grid points inside the cloud box. This means that the original method is very ineffective where a fine resolution inside the cloud box is required to resolve the cloud inhomogeneities.

A solution to this problem is the "sequential update of the radiation field", which is shown schematically in Figure 10.3. For simplicity it will be explained in detail for a 1D cloud box. We divide the update of the radiation field, i.e., the radiative transfer step calculations for all positions and directions inside the cloud box, into three parts: Update for "up-looking" zenith angles (0° $\leq \psi_{\rm up} \leq$ 90°), for "down-looking" angles ($\psi_{\rm limit} \leq \psi_{\rm down}$ $\leq$ 180°) and for "limb-looking" angles (90° $< \psi_{\rm limb} < \psi_{\rm limit}$). The "limb-looking" case is needed, because for angles between 90° and $\psi_{\rm limit}$ the intersection point is at the same pressure level as the observation point. The limiting angle $\psi_{\rm limit}$ is calculated geometrically. Note that the propagation direction of the radiation is opposite to the viewing direction or the direction of the line of sight, which is indicated by the arrows. In the 1D case the radiation field is a set of Stokes vectors each of which depend upon the position and direction:

$$\mathcal{I} = \{\mathbf{I}(P_i, \psi_l)\}\,. \tag{10.25}$$

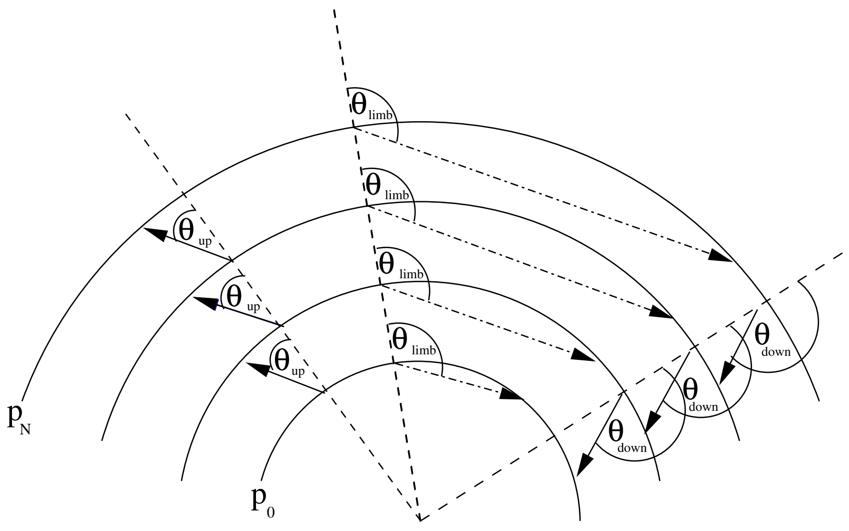

Figure 10.3: Schematic of the sequential update (1D) showing the three different parts: "up-looking" corresponds to zenith angles $\psi_{\rm up}$, "limb-looking" corresponds to $\psi_{\rm limb}$ "down-looking" corresponds to $\psi_{\rm down}$.

The *boundary condition* for the calculation is the incoming radiation field on the cloud box boundary $\mathcal{I}^{bd}$:

$$\begin{aligned} \mathcal{I}^{bd} = \{\mathbf{I}(P_i, \psi_l)\} \text{ where } \quad & P_i = P_N \,\forall\, \psi_l \in [0, \psi_{\rm limit}] \\ & P_i = P_0 \,\forall\, \psi_l \in (\psi_{\rm limit}, 180°], \end{aligned} \tag{10.26}$$

where P_0 and P_N are the pressure coordinates of the lower and upper cloud box boundaries respectively. For down-looking directions, the intensity field at the lower-most cloud box boundary and for up- and limb-looking directions the intensity field at the uppermost cloud box boundary are the required boundary conditions respectively.

10.2.1 Up-looking directions

The first step of the sequential update is to calculate the intensity field for the pressure coordinate P_{N-1}, the pressure level below the uppermost boundary, for all up-looking directions. Radiative transfer steps are calculated for paths starting at the uppermost boundary and propagating to the $(N-1)$ pressure level. The required input for this radiative transfer step are the averaged coefficients of the uppermost cloud box layer and the Stokes vectors at the uppermost boundary for all up-looking directions. These are obtained by interpolating the boundary condition $\mathcal{I}^{bd}$ on the appropriate zenith angles. Note that the zenith angle of the propagation path for the observing direction ψ_l does not equal ψ_l' at the intersection point due to the spherical geometry. If ψ_l is close to 90° this difference is most significant.

To calculate the intensity field for the pressure coordinate P_{N-2}, we repeat the calculation above. We have to calculate a radiative transfer step from the $(N-1)$ to the $(N-2)$ pressure level. As input we need the interpolated intensity field at the $(N-1)$ pressure level, which has been calculated in the last step.

For each pressure level $(m-1)$ we take the interpolated field of the layer above $(\mathcal{I}(P_m)^{(1)})$. Using this method, the scattering influence from particles in the upper-most cloud box layer can propagate during one iteration down to the lower-most layer. This means that the number of iterations does not scale with the number of pressure levels, which would be the case without sequential update.

The radiation field at a specific point in the cloud box is obtained by solving Equation 10.10. For up-looking directions at position P_{m-1} we may write:

$$\begin{aligned} \mathbf{I}\left(P_{m-1},\psi_{\rm up}\right)^{(1)} &= e^{-\overline{\langle \mathbf{K}(\psi_{\rm up})\rangle} s}\mathbf{I}\left(P_m,\psi_{\rm up}\right)^{(1)} \\ &+ \left(\mathbf{I} - e^{-\overline{\langle \mathbf{K}(\psi_{\rm up})\rangle} s}\right)\overline{\langle \mathbf{K}(\psi_{\rm up})\rangle}^{-1}\left(\overline{\langle \mathbf{a}(\psi_{\rm up})\rangle}\,\bar{B} + \overline{\left\langle \mathbf{S}\left(\psi_{\rm up}\right)^{(0)}\right\rangle}\right). \end{aligned} \tag{10.27}$$

For simplification we write

$$\mathbf{I}(P_{m-1},\psi_{\rm up})^{(1)} = \mathbf{A}(\psi_{\rm up})\mathbf{I}\left(P_m,\psi_{\rm up}\right)^{(1)} + \mathbf{B}(\psi_{\rm up}). \tag{10.28}$$

Solving this equation sequentially, starting at the top of the cloud and finishing at the bottom, we get the updated radiation field for all up-looking angles.

$$\mathcal{I}(P_i,\psi_{\rm up})^{(1)} = \left\{\mathbf{I}^{(1)}\left(P_i,\psi_l\right)\right\} \qquad \forall\, \psi_l \in [0, 90°]. \tag{10.29}$$

10.2.2 Down-looking directions

The same procedure is done for down-looking directions. The only difference is that the starting point is the lower-most pressure level P_1 and the incoming clear sky field at the lower cloud box boundary, which is interpolated on the required zenith angles, is taken as boundary condition. The following equation is solved sequentially, starting at the bottom of the cloud box and finishing at the top:

$$\mathbf{I}(P_m,\psi_{\rm down})^{(1)} = \mathbf{A}(\psi_{\rm down})\mathbf{I}\left(P_{m-1},\psi_{\rm down}\right)^{(1)} + \mathbf{B}(\psi_{\rm down}). \tag{10.30}$$

This yields the updated radiation field for all down-looking angles.

$$\mathcal{I}(P_i,\psi_{\rm down})^{(1)} = \left\{\mathbf{I}^{(1)}\left(P_i,\psi_l\right)\right\} \qquad \forall\, \psi_l \in [\psi_{\rm limit}, 180°]. \tag{10.31}$$

10.2.3 Limb directions

A special case for limb directions, which correspond to angles slightly above 90° had to be implemented. If the tangent point is part of the propagation path step, the intersection point is exactly at the same pressure level as the starting point. In this case the linearly interpolated clear sky field is taken as input for the radiative transfer calculation, because we do not have an already updated field for this pressure level:

$$\mathbf{I}(P_m, \psi_{\text{limb}})^{(1)} = \mathbf{A}(\psi_{\text{limb}})\mathbf{I}\left(P_m, \psi_{\text{limb}}\right)^{(0)} + \mathbf{B}(\psi_{\text{limb}}) \tag{10.32}$$

By solving this equation the missing part of the updated radiation field is obtained

$$\mathcal{I}(P_i, \psi_{\text{limb}})^{(1)} = \{\mathbf{I}\left(P_i, \psi_l\right)\} \qquad \forall\ \psi_l \in]90^\circ, \psi_{\text{limit}}[\tag{10.33}$$

For all iterations the sequential update is applied. Using this method the number of iterations depends only on the optical thickness of the cloud or on the number of multiple-scattering events, not on the number of pressure levels.

10.3 Numerical Issues

10.3.1 Grid optimization and interpolation

The accuracy of the DOIT method depends very much on the discretization of the zenith angle. The reason is that the intensity field strongly increases at about $\psi = 90^\circ$. For angles below 90° ("up-looking" directions) the intensity is very small compared to angles above 90° ("down-looking" directions), because the thermal emission from the lower atmosphere and from the ground is much larger than thermal emission from trace gases in the upper atmosphere. Figure 10.4 shows an example intensity field as a function of zenith angle for different pressure levels inside a cloud box, which is placed from 7.3 to 12.7 km altitude, corresponding to pressure limits of 411 hPa and 188 hPa respectively. The cloud box includes 27 pressure levels. The frequency of the sample calculation was 318 GHz. A midlatitude-summer scenario including water vapor, ozone, nitrogen and oxygen was used. The atmospheric data was taken from the FASCOD [*Anderson et al.*, 1986] and the spectroscopic data was obtained from the HITRAN database [*Rothman et al.*, 1998]. For simplicity this 1D set-up was chosen for all sample calculations in this section. As the intensity (or the Stokes vector) at the intersection point of a propagation path is obtained by interpolation, large interpolation errors can occur for zenith angles of about 90° if the zenith angle grid discretization is too coarse. Taking a very fine equidistant zenith angle grid leads to very long computation times. Therefore a zenith angle grid optimization method is required.

For the computation of the scattering integral it is possible to take a much coarser zenith angle resolution without losing accuracy. It does not make sense to use the zenith angle grid, which is optimized to represent the radiation field with a certain accuracy. The integrand is the product of the phase matrix and the radiation field. The peaks of the phase matrices can be at any zenith angle, depending on the incoming and the scattered directions. The multiplication smooths out both the radiation field increase at 90° and the peaks of the phase matrices. Test calculations have shown that an increment of 10° is sufficient. Taking the equidistant grid saves the computation time of the scattering integral to a very large extent, because much less grid points are required.

Zenith angle grid optimization

As a reference field for the grid optimization the DOIT method is applied for an empty cloud box using a very fine zenith angle grid. The grid optimization routine finds a reduced

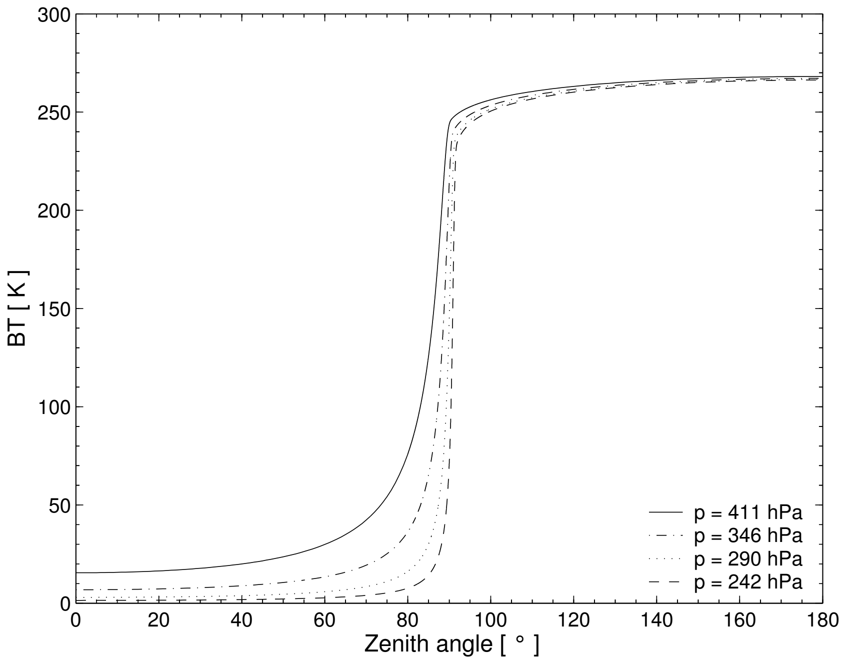

Figure 10.4: Intensity field for different pressure levels.

zenith angle grid which can represent the intensity field with the desired accuracy. It first takes the radiation at 0° and 180° and interpolates between these two points on all grid points contained in the fine zenith angle grid for all pressure levels. Then the differences between the reference radiation field and the interpolated field are calculated. The zenith angle grid point, where the difference is maximal is added to 0° and 180°. After that the radiation field is interpolated between these three points forming part of the reduced grid and again the grid point with the maximum difference is added. Using this method more and more grid points are added to the reduced grid until the maximum difference is below a requested accuracy limit.

The top panel of Figure 10.5 shows the clear sky radiation in all viewing directions for a sensor located at 13 km altitude. This result was obtained with a switched-off cloud box. The difference between the clear sky part of the ARTS model and the scattering part is that in the clear sky part the radiative transfer calculations are done along the line of sight of the instrument whereas inside the cloud box the RT calculations are done as described in the previous section to obtain the full radiation field inside the cloud box. In the clear sky part the radiation field is not interpolated, therefore we can take the clear sky solution as the exact solution.

The interpolation error is the relative difference between the exact clear sky calculation (cloud box switched off) and the clear sky calculation with empty cloud box. The bottom panels of Figure 10.5 show the interpolation errors for zenith angle grids optimized with three different accuracy limits (0.1%, 0.2% and 0.5%.). The left plot shows the critical region close to 90°. For a grid optimization accuracy of 0.5% the interpolation error becomes very large, the maximum error is about 8%. For grid accuracies of 0.2% and 0.1% the maximum interpolation errors are about 0.4% and 0.2% respectively. However for most angles

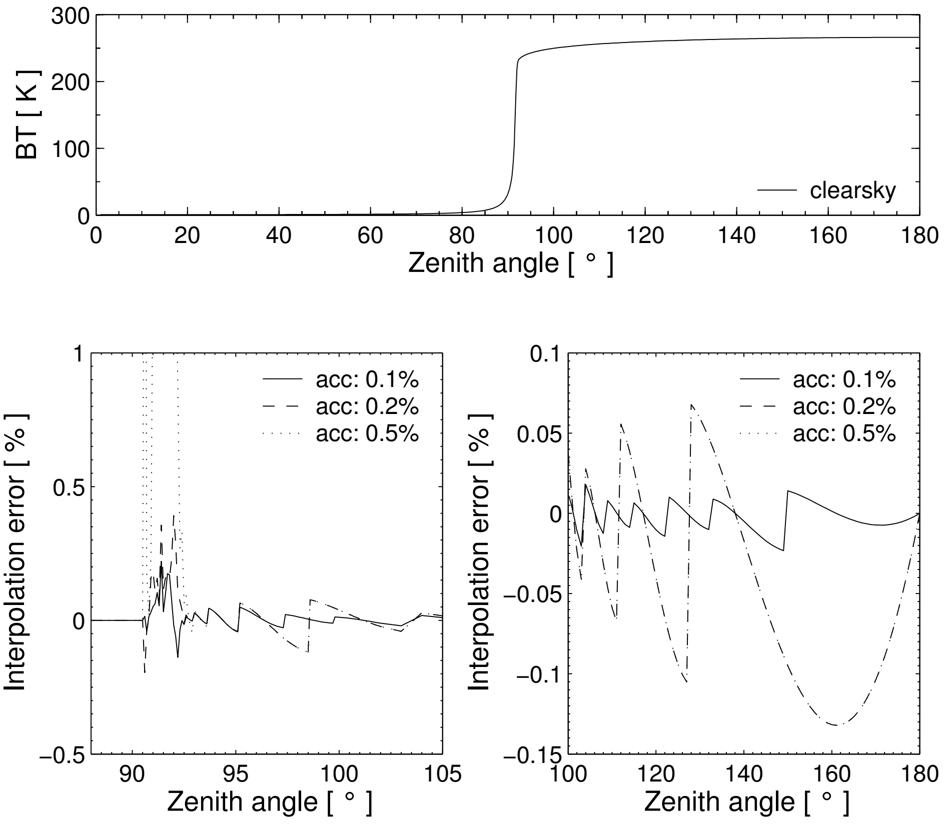

Figure 10.5: Interpolation errors for different grid accuracies. Top panel: Clear sky radiation simulated for a sensor at an altitude of 13 km for all viewing directions. Bottom left: Grid optimization accuracy for limb directions. Bottom right: Grid optimization accuracy for down-looking directions.

it is below 0.2%, for all three cases. For down-looking directions from 100° to 180° the interpolation error is at most 0.14% for grid accuracies of 0.2% and 0.5% and for a grid accuracy of 0.1% it is below 0.02%.

Interpolation methods

Two different interpolation methods can be chosen in ARTS for the interpolation of the radiation field in the zenith angle dimension: linear interpolation or three-point polynomial interpolation. The polynomial interpolation method produces more accurate results provided that the zenith angle grid is optimized appropriately. The linear interpolation method on the other hand is safer. If the zenith angle grid is not optimized for polynomial interpolation one should use the simpler linear interpolation method. Apart from the interpolation of the radiation field in the zenith angle dimension linear interpolation is used everywhere in the model. Figure 10.6 shows the interpolation errors for the different interpolation methods. Both calculations are performed on optimized zenith angle grids, for polynomial interpolation 65 grid points were required to achieve an accuracy of 0.1% and for linear interpolation 101 points were necessary to achieve the same accuracy. In the region of about 90° the interpolation errors are below 1.2% for linear interpolation and below 0.2% for polynomial interpolation. For the other down-looking directions the differences are below 0.08% for linear and below 0.02% for polynomial interpolation. It is obvious that polynomial interpolation gives more accurate results. Another advantage is that the calculation is faster because less grid points are required, although the polynomial interpolation method itself is slower than the linear interpolation method. Nevertheless, we have implemented the polynomial interpolation method so far only in the 1D model. In the 3D model, the grid optimization needs to be done over the whole cloud box, where it is not obvious that one can save grid points. Applying the polynomial interpolation method using non-optimized grids can yield much larger interpolation errors than the linear interpolation method.

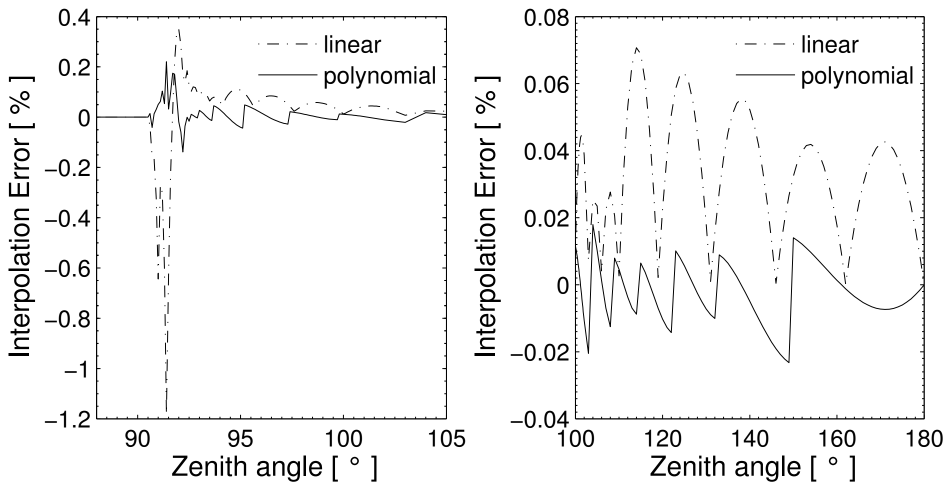

Figure 10.6: Interpolation errors for polynomial and linear interpolation.

Error estimates

The interpolation error for scattering calculations can be estimated by comparison of a scattering calculation performed on a very fine zenith angle grid (resolution 0.001° from 80° to 100°) with a scattering calculation performed on an optimized zenith angle grid with 0.1% accuracy. The cloud box used in previous test calculations is filled with spheroidal particles with an aspect ratio of 0.5 from 10 to 12 km altitude. The ice mass content is assumed to be $4.3 \cdot 10^{-3}$ g/m^3 at all pressure levels. An equal volume sphere radius of 75 μm is assumed. The particles are either completely randomly oriented (p20) or azimuthally randomly oriented (p30) (cf. Section 9.2.3). The top panels of Figure 10.7 show the interpolation errors of the intensity. For both particle orientations the interpolation error is in the same range as the error for the clear sky calculation, below 0.2 K. The bottom panels show the interpolation errors for Q. For the randomly oriented particles the error is below 0.5%. For the horizontally aligned particles with random azimuthal orientation it goes up to 2.5% for a zenith angle of about 91.5°. It is obvious that the interpolation error for Q must be larger than that for I because the grid optimization is accomplished using only the clear-sky field, where the polarization is zero. Only the limb directions about 90° are problematic, for other down-looking directions the interpolation error is below 0.2%.

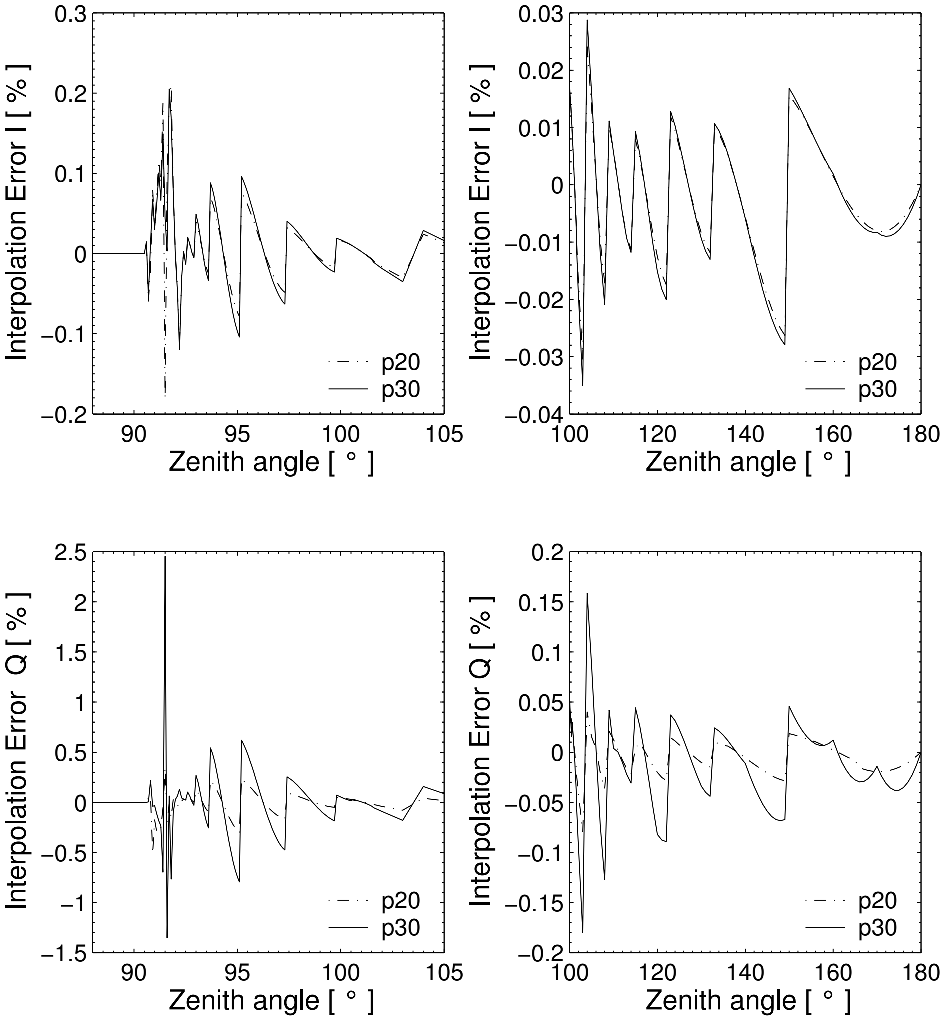

Figure 10.7: Interpolation errors for a scattering calculation. Left panels: Interpolation errors for limb directions. Right panels: Interpolation errors for down-looking directions. Top: Intensity I, Bottom: Polarization difference Q

10.4 Implementation

The workspace methods required for DOIT calculations are implemented in the files `m_scatrte.cc`, `m_cloudbox.cc` and `m_optproperties.cc`.

10.4.1 1D control file example

This example demonstrates how to set up a 1D DOIT calculation. A full running controlfile example for a DOIT calculation can be found in the ARTS package in the tests/DOIT directory. The file is called `TestDOIT.arts`. For detailed descriptions of the workspace methods and variables please refer also to the online help (`arts -d ...`).

10.4.2 DOIT frame

As a first step for a DOIT calculation we have to calculate the incoming field on the boundary of the cloudbox. This is done using the workspace method CloudboxGetIncoming:

```
CloudboxGetIncoming
```

The next step is the initialization of variables required for a DOIT calculation using DoitInit:

```
DoitInit
```

The grid discretization plays a very significant role in discrete ordinate methods. In spherical geometry the zenith angular grid is of particular importance (cf. Section 10.3.1). The angular discretization is defined in the workspace method DoitAngularGridsSet:

```
DoitAngularGridsSet( doit_za_grid_size,
                     scat_aa_grid, scat_za_grid,
                     19, 10, "doit_za_grid_opt.xml" )
```

For down-looking geometries it is sufficient to define the generic inputs:

`N_za_grid`	Number of grid points in zenith angle grid, recommended value: 19
`N_aa_grid`	Number of grid points in azimuth angle grid, recommended value: 37

From these numbers equally spaced grids are created and stored in the work space variables scat_za_grid and scat_aa_grid.

For limb simulations it is important to use an optimized zenith angle grid with a very fine resolution about 90° for the RT calculations. Such a grid can be generated using the workspace method doit_za_grid_optCalc. Please refer to the online documentation of this method. The filename of the optimized zenith angle grid can be given as a generic input. If a filename is given, the equidistant grid is taken for the calculation of the scattering integrals and the optimized grid is taken for the radiative transfer part. Otherwise, if no filename is specified (`za_grid_opt_file = ""`) the equidistant grid is taken for the calculation of the scattering integrals and for the radiative transfer calculations. This option makes sense for down-looking cases to speed up the calculation.

The main agenda for a DOIT calculation is doit_mono_agenda. The agenda is executed by the workspace method ScatteringDoit:

```
ScatteringDoit
```

The DOIT main agenda

The agenda doit_main_agenda requires the incoming clearsky field on the cloudbox boundary as an input and gives as output the scattered field on the cloudbox boundary if the sensor is placed outside the cloudbox or the full scattered field in the cloudbox if the sensor is placed inside the cloudbox.

```
AgendaSet( doit_mono_agenda ){
 # Prepare scattering data for DOIT calculation (Optimized method):
   DoitScatteringDataPrepare
 # Alternative method (needs less memory):
 # scat_data_monoCalc
 # Set first guess field:
   doit_i_fieldSetClearsky
 # Perform iterations: 1. scattering integral. 2. RT calculations with
 #   fixed scattering integral field, 3. convergence test
   doit_i_fieldIterate
 # Put solution into interface for clearsky calculation
   DoitCloudboxFieldPut
 }
```

The first method DoitScatteringDataPrepare reads the single scattering data and interpolates it on the requested frequency. It also performs the transformation from the scattering frame into the laboratory frame. Alternatively the method scat_data_monoCalc can be used. In this case only the frequency interpolation is done and the transformations are done later. The advantage is that this method needs less memory. For 1D calculation it is recommended to use DoitScatteringDataPrepare because it is much more efficient.

The method doit_i_fieldSetClearsky interpolates the incoming radiation field on all points inside the cloudbox to obtain the initial field (doit_i_field) for the DOIT calculation. As a test one can alternatively start with a constant radiation field using the method doit_i_fieldSetConst.

The iteration is performed in the method doit_i_fieldIterate, which includes the calculation of the scattering integral field (doit_scat_field), the radiative transfer calculations in the cloudbox with fixed scattering integral and the convergence test.

After convergence is obtained the radiation field inside the cloudbox is stored in the interface variable scat_i_p if the sensor is located outside the cloudbox, or in the variable doit_i_field1D_spectrum if the sensor is located inside the cloudbox. This is done by the workspace method DoitCloudboxFieldPut. In contrast to doit_i_field the interface variables include an additional dimension for the frequency.

Agendas used in doit_i_fieldIterate

There are several methods which can be used in doit_i_fieldIterate, for instance for the calculation of the scattering integral. The methods are selected in the control-file by defining several agendas.

Calculation of the scattering integral: To calculate the scattering integral (Equation 10.7) the phase matrix (pha_mat) is required. How the phase matrix is calculated is defined in the agenda pha_mat_spt_agenda:

```
# Calculation of the phase matrix
AgendaSet( pha_mat_spt_agenda ){
 # Optimized option:
   pha_mat_sptFromDataDOITOpt
 # Alternative option:
 #  pha_mat_sptFromMonoData
}
```

If in doit_mono_agenda the optimized method DoitScatteringDataPrepare is used we have to use here the corresponding method pha_mat_sptFromDataDOITOpt. Otherwise we have to use pha_mat_sptFromMonoData.

To do the integration itself, we have to define doit_scat_field_agenda:

```
AgendaSet( doit_scat_field_agenda ){
 doit_scat_fieldCalcLimb
 # Alternative:
 # doit_scat_fieldCalc
}
```

Here we have two options. One is doit_scat_fieldCalcLimb, which should be used for limb simulations, for which we need a fine zenith angle grid resolution to represent the radiation field. This method has to be used if a zenith angle grid file is given in DoitAngularGridsSet. The scattering integral can be calculated on a coarser grid resolution, hence in doit_scat_fieldCalcLimb, the radiation field is interpolated on the equidistant angular grids specified in DoitAngularGridsSet by the generic inputs `Nza` and `Naa`. Alternatively, one can use doit_scat_fieldCalc, where this interpolation is not performed. This function is efficient for simulations in up- or down-looking geometry, where the we do not need the fine zenith angle grid resolution about 90°.

Radiative transfer with fixed scattering integral term: With a fixed scattering integral field the radiative transfer equation can be solved (Equation 10.9). The workspace method to be used for this calculation is defined in doit_rte_agenda. The most efficient and recommended workspace method is doit_i_fieldUpdateSeq1D where the sequential update which is described in Section 10.2 is applied. The workspace method doit_i_fieldUpdate1D does the same calculation without sequential update and is therefore much less efficient because the number of iterations depends in this case on the number of pressure levels in the cloudbox. Other options are to use a plane-parallel approximation implemented in the workspace method doit_i_fieldUpdateSeq1DPP. This method is not much more efficient than doit_i_fieldUpdateSeq1D, therefore it is usually better to use doit_i_fieldUpdateSeq1D since it is more accurate.

```
AgendaSet( doit_rte_agenda ){
     doit_i_fieldUpdateSeq1D
   # Alternatives:
   # doit_i_fieldUpdateSeq1DPP
   # i_fieldUpdate1D
   }
```

The optical properties of the particles, i.e., extinction matrix and absorption vector are required for solving the radiative transfer equation. How they are calculated is specified in spt_calc_agenda. The workspace method opt_prop_sptFromMonoData requires that the raw data is already interpolated on the frequency of the monochromatic calculation. This requirement is fulfilled when DoitScatteringDataPrepare or scat_data_monoCalc is executed before doit_i_fieldIterate (see Section 10.4.2). The work space method ext_matAddPart and abs_vecAddPart are used to extract the absorption vector abs_vec and extinction matrix ext_mat from the workspace variable opt_prop_spt. The gas absorption is added internally.

```
AgendaSet( spt_calc_agenda ){
   opt_prop_sptFromMonoData
}
AgendaSet( opt_prop_part_agenda ){
   ext_matInit
   abs_vecInit
   ext_matAddPart
   abs_vecAddPart
}
```

Convergence test: After the radiative transfer calculations with a fixed scattering integral field are complete the newly obtained radiation field is compared to the old radiation field by a convergence test. The functions and parameters for the convergence test are defined in the agenda doit_conv_test_agenda. There are several options. The workspace methods doit_conv_flagAbsBT and doit_conv_flagAbs compare the absolute differences of the radiation field element-wise as described in Equation 10.19. The convergence limits are specified by the generic input `epsilon` which specifies the convergence limit. A limit must be given for each Stokes component. In doit_conv_flagAbsBT the limits must be specified in Rayleigh Jeans brightness temperatures whereas in doit_conv_flagAbs they must be defined in the basic radiance unit ([W/(m^{2}Hz sr)]. Another option is to perform a least square convergence test using the workspace method doit_conv_flagLsq. Test calculations have shown that this test is not safe, therefore the least square convergence test should only be used for test purposes.

```
AgendaSet( doit_conv_test_agenda) {
    doit_conv_flagAbsBT( doit_conv_flag, doit_iteration_counter,
                         doit_i_field, doit_i_field_old,
                         f_grid, f_index,
                         [0.1, 0.01, 0.01, 0.01] )
    # Alternative: Give limits in radiances
    #   doit_conv_flagAbs( doit_conv_flag, doit_iteration_counter,
    #                      doit_i_field, doit_i_field_old,
    #                      [0.1e-15, 0.1e-18, 0.1e-18, 0.1e-18] )
    #
    # If you want to look at several iteration fields, for example
    # to investigate the convergence behavior, you can use
    # the following workspace method:
    # DoitWriteIterationFields( doit_iteration_counter, doit_i_field,
    #                           [2, 4] )
}
```

10.4.3 Propagation of the DOIT result towards the sensor

In order to propagate the result of the scattering calculation towards the sensor, the fields needs to be interpolated on the direction of the sensor's line of sight. This is done in the workspace method iyInterpCloudboxField, which has to be put into the agenda iy_cloudbox_agenda:

```
AgendaSet( iy_cloudbox_agenda ){
    iyInterpCloudboxField
}
```

10.4.4 3D DOIT calculations

The DOIT method is implemented for 1D and 3D spherical atmospheres, but it is strongly recommended to use it only for 1D calculations, because there are several numerical difficulties related to the grid discretizations. It is difficult to find appropriate discretizations to get sufficiently accurate results in reasonable computation time. Therefore only experienced ARTS users should use DOIT for 3D calculations only for smaller cloud scenarios. Please refer to the online documentation for the workspace method for 3D scattering calculations (doit_i_fieldUpdateSeq3D). All other workspace methods adapt automatically to the atmospheric dimensionality.

Chapter 11

Weighting functions: clear-sky conditions

Inversions of both OEM and Tikhonov type require that the "weighting functions" can be provided by the forward model [see e.g. *Eriksson*, 2000]. A retrieval characterisation following *Rodgers* [1990, 2000] raises the same demand. A weighting function is defined as

$$\frac{\partial \mathbf{y}}{\partial x_p} \tag{11.1}$$

where $\mathbf{y}$ is the vector of measurement data and x_p is one forward model (scalar) variable. See further Sec. 1.3 of *ARTS Theory*. The nomenclature of that section is also used here.

A weighting function forms a column of the complete weighting function matrix, $\mathbf{K_x}$. The usage of the name weighting function is maybe restricted just to atmospheric sounding and the Jacobian is a more commonly encountered name for $\mathbf{K_x}$, in the general scientific literature. In the documentation of ARTS you find both terms. Also the term "the Jacobians" is used, which shall be interpreted as "the weighting functions". These names refer normally to $\mathbf{K_x}$, the partial derivatives with respect to the variables to be retrieved, forming the state vector $\mathbf{x}$. However, in the context of retrieval characterisation, the same matrix for the remaining model parameters is of equally high interest, denoted as $\mathbf{K_b}$. In the same manner, the terms inversion and retrieval are used interchangeably. "Weighting function" is below shortened as WF.

The main task of the user is to select which quantities that shall be retrieved, and to define the associated retrieval grids. These aspects must be considered for successful inversions, but are out of scope for this document. Beside for the most simple retrievals, it is further important to understand how the different WFs are calculated. A practical point is the calculation speed, primarily determined if perturbations or analytical expressions are used (Sec. 11.1). The derivation of the WFs involves some approximations due to theoretical and practical considerations. Such approximations can be accepted, if of low or moderate size, but will result in a slower convergence (the inversion will require more iter-

History

110826	First complete version by Patrick Eriksson.

ations). Due to these later aspects, and to meet the needs of more experienced users, this section is relatively detailed and contains a (high?) number of equations.

This section is restricted to WFs for clear-sky conditions, i.e. to be applied outside the cloudbox. As the basic radiative transfer expressions, the complete Stokes vector is considered, to correctly handle polarisation signatures created by the surface. So far none of the scattering methods provide WFs.

Sections 11.1 - 11.2 contain information of general character, while the available quantities are discussed in the remaining sections (Section 11.3 and forward).

11.1 Introduction

There are two main approaches for calculating the WFs, by analytical expressions and by perturbations. We start with the conceptually simplest one, but also the more inefficient approach.

11.1.1 Perturbations

The most straightforward method to determine a WF is by perturbing the model parameter of concern. For example, the WF for the state variable p can always be calculated as

$$\mathbf{K}_{\mathbf{x}}^{p} = \frac{\mathcal{F}(\mathbf{x}+\Delta x\mathbf{e}^{p},\mathbf{b}) - \mathcal{F}(\mathbf{x},\mathbf{b})}{\Delta x} \tag{11.2}$$

where $(\mathbf{x},\mathbf{b})$ is the linearization state, $\mathbf{e}^{p}$ is a vector of zeros except for the p:th component that is unity, and Δx is a small disturbance (but sufficiently large to avoid numerical instabilities).

However, it is normally not needed to make a recalculation using the total forward model as the variables are either part of the atmospheric or the sensor state, but not both. In addition, in many cases it is possibly to find short-cuts. For example, the perturbed state can be approximated by an interpolation of existing data (such as for a perturbed zenith angle). Such short-cuts are discussed separately for each retrieval quantity.

11.1.2 Analytical expressions

For most atmospheric variables, such as species abundance, it is possible to derive an analytical expression for the WFs. This is advantageous because they result in faster and more accurate calculations. Such expressions are derived below. Some of the terms involved are calculated as a perturbation. This is partly a consequence of the flexibility of ARTS. The high-level radiative transfer methods (e.g. iyEmissionStandardClearsky) do not know how the low-level methods are defining all quantities, and a fixed analytical expression can not be used (see e.g. Sec. 11.4). So, in practise, the calculations are “semi-analytical”.

To understand the analytical expressions, it is important to remember that ARTS covers the sensor by a response matrix:

$$\mathbf{y} = \mathbf{H}\mathbf{i}. \tag{11.3}$$

This equation covers a single measurement block (cf. Eq. 3.16).

11.1.3 Workspace variables and methods

As a workspace variable, the WF matrix is denoted as jacobian. Auxiliary information is provided by jacobian_quantities and jacobian_indices. The actual calculations are made as part of yCalc.

The retrieval quantities are defined separately, before calling yCalc. This process is started by calling jacobianInit. The retrieval quantities are then introduced through workspace methods named as jacobianAdd*Something*. For example, for atmospheric temperature the method is jacobianAddTemperature. The order in which these “add methods” is called does not matter.

Finalise the definition of retrieval quantities by calling jacobianClose. To disable the calculation of WFs, skip all above, and just use jacobianOff. The methods named jacobianCalc*Something* shall never be used directly. Neither needs the user to consider jacobian_agenda.

The input to the “add methods” differs. In some cases you can select between the analytical and perturbation options. For all perturbation calculations you must specify the size of the perturbation. For atmospheric gases you can use different units. For atmospheric fields, and some other quantities, you must define the retrieval grid(s) to use.

11.2 Basis functions

A forward model must use a discrete representation: it describes each quantity with one or several variables. This is unproblematic for quantities that are of discrete nature (including scalar variables). However, for atmospheric fields and other continuous model quantities, the discrete representation inside the forward model requires consideration. To avoid inconsistencies between model input and output it is important that the mapping from the discrete variables to the “continuous view” of the quantity is well defined, and applied consistently through the forward model . This mapping is given by the basis functions. Similar arguments and nomenclature are found in *Read et al.* [2006].

The basis functions are discussed explicitly in few places in this user guide, but it shall be noted that all interpolations imply an underlying set of basis functions. On the other hand, an understanding of both the derivation and the obtained WFs require direct consideration of the basis functions. ARTS operates with two types of basis functions.

11.2.1 Basis functions for piece-wise linear quantities

To treat an one-dimensional quantity to be piece-wise linear, or to say that a linear interpolation shall be applied, are identical definitions. The basis functions matching this definition have triangular shape, sometimes denoted as “tenth functions”. Such functions are exemplified in Fig. 11.1, see also *Buehler et al.* [2005].

In most cases, the quantity is considered to be undefined outside the end points of the grid. Hence, the basis function for a grid end point is then just “half a tenth”. The exception to this rule is retrieval grids of piece-wise linear variables. To avoid that retrieval grids must cover the complete atmosphere, end point values are assumed to be valid to the end of the atmosphere (or data range of concern). That is, the basis functions for end points of retrieval grids follow the tenth shape inside the grid range, and have a constant value

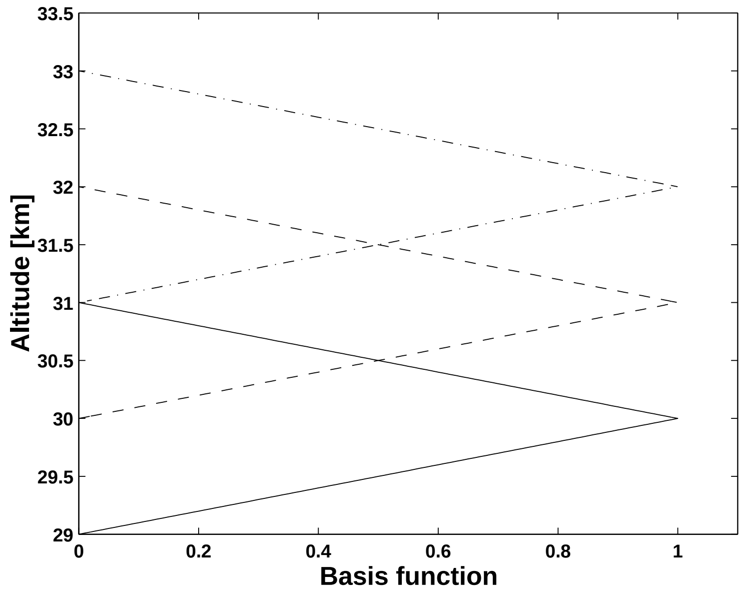

Figure 11.1: Examples on 1d basis functions for a vertical grid with a 1 km spacing: —— 30 km, – – – 31 km and – · – 32 km.

of 1 outside. In terms of interpolation, this matches to allow extrapolation, the applying a "nearest" interpolation for positions outside the covered range (the end values are valid all the way to ±infinity).

The basis functions are defined likewise for higher dimensions, but the tenth functions are then 2D or 3D "tenths".

11.2.2 Polynomial basis functions

Some retrieval quantities are expressed using a polynomial basis. Sensor zenith angle pointing off-set is one such quantity. The off-set is then treated to have a polynomial variation as a function of time. If the offset is assumed to be constant in time, a zero order polynomial shall be selected. If there is also a linear drift with time, use a first order polynomial, etc.

For these basis functions, the explanatory variable (time in the example above) is normalised to cover the range [-1,1], here denoted as z, and the continuous representation (f) of the variable of concern can be written as

$$f(z) = x_0 + x_1(z - b_1) + x_2(z^2 - b_2) + x_3(z^3 - b_3) + x_4(z^3 - b_4) + \dots \tag{11.4}$$

where $x_0, x_1, \dots$ are the coefficients to be retrieved (elements of $\mathbf{x}$). The interpretation of a retrieval is simplified if the average of f equals x_0, and the scalars $b_1, b_2, \dots$ are selected, schematically, as

$$0 = \int_{-1}^{1} (z^n - b_n)\,\mathrm{d}z, \quad n > 0. \tag{11.5}$$

According to this expression, b_n is zero for odd n. However, z is in practise a discrete variable (z_i, not necessarily symmetric around 0), and b_n is taken as the average of z_i^n: all

b_n can be non-zero. The normalisation of z is not only made for interpretation reasons, it can be required for pure numerical reasons, such as when z represent frequency (in Hz).

In practise, the basis functions are vectors, denoted below as $\mathbf{z}_i$. Element j of $\mathbf{z}_i$ is

$$\mathbf{z}_i(j) = z(j)^i - b_i. \tag{11.6}$$

11.3 Absorption species

11.3.1 Common practicalities

To obtain WFs for absorption species, use jacobianAddAbsSpecies. The method handles one species at the time. The calculations can either be done in in "analytical" or "perturbation" manner.

For gases, the WFs can be provided for several units of the gas abundance:

vmr Volume mixing ratio (a value between 0 and 1). The WF divided by 10^6 corresponds to that 1 ppm of the gas is added to the atmospheric volume of concern.

nd Number density. The WF corresponds here to that one molecule is added.

rel Relative/fractional change. In a perfectly linear case, the WF corresponds here to that the gas amount is doubled.

logrel This option returns the same WFs as "rel", but is included to flag that the natural logarithm of the "rel" unit is retrieved.

For the "rel" and "logrel" options it is important to note that ARTS calculate the WFs with respect to the given state, ARTS does not know anything about the actual reference state for which the "rel" unit is valid (where normally the a priori state is selected). For iterative inversions, a rescaling of the WFs provided by ARTS is likely needed, to make the WFs valid with respect to the (original) reference state. For the assumption made inside ARTS, the WFs for "rel" and "logrel" are identical.

A second main consideration is to select the retrieval grids. For analytical calculations there are no other selections to be made.

11.3.2 Perturbation calculations

For these pure numerical calculations, also the size of the perturbation must be specified (Δx in Eq. 11.2). The perturbation shall given following the unit selected. The same value is applied for all WFs (which can cause practical problems ...).

11.3.3 Analytical expressions

The final radiance obtained through Eq. 3.8 can be expressed as

$$I = I' + e^{-\tau'}\left[\bar{B}_i(1 - e^{-\tau_i}) + I_i e^{-\tau_i}\right] \tag{11.7}$$

where τ' is the optical thickness between the sensor and point $i + 1$ and I' is all emission generated along the same distance.

Let x be one element of $\mathbf{x}$, representing the amount of some gas at some point in the atmosphere. The contribution to the weighting function for x, originating from the radiative transfer step covered by Eq. 11.7, is calculated by applying the chain rule:

$$\frac{\partial \mathbf{y}}{\partial x} = \frac{\partial \mathbf{y}}{\partial \mathbf{i}} \frac{\partial \mathbf{i}}{\partial I} \frac{\partial I}{\partial \tau_i} \left[\frac{\partial \tau_i}{\partial k_i} \frac{\partial k_i}{\partial x_i} \frac{\partial x_i}{\partial x} + \frac{\partial \tau_i}{\partial k_{i+1}} \frac{\partial k_{i+1}}{\partial x_{i+1}} \frac{\partial x_{i+1}}{\partial x} \right], \tag{11.8}$$

where x_i and x_{i+1} is the amount of the gas at point i and $i+1$, respectively. Some of these terms can be expressed explicitly:

$$\begin{aligned} \frac{\partial \mathbf{y}}{\partial \mathbf{i}} &= \mathbf{H}, \qquad \text{(cf. Eq. 11.3)} && (11.9)\\ \frac{\partial I}{\partial \tau_i} &= e^{-\tau'} (\bar{B}_i - I_i)\, e^{-\tau_i}, && (11.10)\\ \frac{\partial \tau_i}{\partial k_i} = \frac{\partial \tau_i}{\partial k_{i+1}} &= \Delta s_i/2, \qquad \text{(cf. Eq. 3.10)} && (11.11)\\ \frac{\partial k_i}{\partial x_i} &= \sigma_i, && (11.12) \end{aligned}$$

where σ_i is the absorption cross-section for the species (at point i and in a unit matching the unit of x). The term $\partial \mathbf{i}/\partial I$ gives the position in $\mathbf{i}$ where the I of concern is stored. For clarity and efficiency reasons, the following nomenclature is used in the source code:

$$\begin{aligned} X &= \frac{\Delta s_i}{2} e^{-\tau'} e^{-\tau_i}, && (11.13)\\ Y &= \frac{\partial I}{\partial \tau_i} \frac{\partial \tau_i}{\partial k_i} \qquad (= X(\bar{B}_i - I_i))\,. && (11.14) \end{aligned}$$

The term $\partial x_i/\partial x$ appears due to the fact that x_i and x are placed at different positions, and the representation of the atmospheric fields must be considered here. In practise, the term is calculated as the value of the basis function for x at the location of x_i (further discussed in *Buehler et al.* [2005]). This is a slight approximation with respect to the goal of fully incorporating the piece-wise linear representation in the weighting functions [*Buehler et al.*, 2005]. A low value of Δs_i decreases the degree of approximation.

It can be noted that $\partial x_i/\partial x$ is normally non-zero for more than one element of $\mathbf{x}$. The exception is if the positions of x_i and x are identical. Reversely, the weighting function for element x can have contributions from several propagation path points (x_i), as well as from several radiance spectra. The expressions above are implemented in iyEmissionStandard-Clearsky.

For higher Stokes components, or if emission is ignored, Eq. 11.7 is simplified to

$$I = e^{-\tau'} I_i e^{-\tau_i}. \tag{11.15}$$

This affects only one of the chain rule terms:

$$\frac{\partial I}{\partial \tau_i} = -e^{-\tau'} I_i e^{-\tau_i}, \tag{11.16}$$

giving

$$\frac{\partial I}{\partial \tau_i} \frac{\partial \tau_i}{\partial k_i} = -X I_i. \tag{11.17}$$

This later form is used for Stokes element 2 to 4 in iyEmissionStandardClearsky and for all components in iyBeerLambertStandardClearsky.

In practise, all Stokes components are treated in parallel, to simply incorporate surface scattering (and in the future possibly also cloud scattering). Transmission $e^{-\tau'}$ must then be expressed as a matrix, $\mathbf{T}'$. For scattered down-welling radiation ($\mathbf{I}_i^d$) the effective transmission matrix is

$$\mathbf{T}' = \mathbf{T}_2\mathbf{R}_i\mathbf{T}_1, \tag{11.18}$$

where $\mathbf{T}_2$ is the transmission between the surface and the sensor, $\mathbf{R}_i$ is defined in Eq. 3.6 and $\mathbf{T}_1$ is the transmission between the point $i + 1$ and the surface.

Limitations

A constraint for the analytical expressions above is that the effect of the variable must only be local. Main examples on non-local effects should occur through hydrostatic equilibrium applies and refraction. Significant impact of a gas through these mechanisms should only be found for water vapour in the (lower?) troposphere.

11.4 Atmospheric temperatures

11.4.1 Common practicalities

To obtain WFs for absorption species, use jacobianAddTemperature. The calculations can either be done in "analytical" or "perturbation" manner. Retrieval grids must be specified.

A special consideration for temperature is hydrostatic equilibrium. If effects origination through hydrostatic equilibrium shall be included in the WFs, or not, is selected by an argument denoted as `hse`. A full account of hydrostatic equilibrium is possible for perturbation calculations, while the analytical approach only treats the local effect (see further below).

11.4.2 Perturbation calculations

The size of the perturbation must be selected (in K).

Complete radiative transfer calculations are done after perturbing the temperature field. Hence, all possible effects are included, such as changed propagation paths through the impact of temperature on the refractive index. Please, note that hydrostatic equilibrium comes in during the perturbation. If `hse` is set to "on", also z_field is recalculated as part of the temperature perturbation (Section 2.4). If set to "off", there is no change of z_field. That is, you must make an active choice regarding hydrostatic equilibrium, while others effects are included automatically.

11.4.3 Analytical expressions

As for gases, only local effects are considered by the analytical expressions. A changed temperature is not just affecting the absorption, but also the Planck function:

$$\frac{\partial \mathbf{y}}{\partial T} = \frac{\partial \mathbf{y}}{\partial \mathbf{i}}\frac{\partial \mathbf{i}}{\partial I}\left[\frac{\partial I}{\partial \tau_i}\frac{\partial \tau_i}{\partial T} + \frac{\partial I}{\partial \bar{B}_i}\frac{\partial \bar{B}_i}{\partial T}\right]. \tag{11.19}$$

The partial derivative of τ_i must now also consider the term Δs_i due to the possible constrain of hydrostatic equilibrium:

$$\begin{aligned}\frac{\partial \tau_i}{\partial T} &= \frac{\partial \tau_i}{\partial k_i}\frac{\partial k_i}{\partial T_i}\frac{\partial T_i}{\partial T} + \frac{\partial \tau_i}{\partial k_{i+1}}\frac{\partial k_{i+1}}{\partial T_{i+1}}\frac{\partial T_{i+1}}{\partial T} + \\ &\quad + \frac{\partial \tau_i}{\partial \Delta s_i}\left[\frac{\partial \Delta s_i}{\partial T_i}\frac{\partial T_i}{\partial T} + \frac{\partial \Delta s_i}{\partial T_{i+1}}\frac{\partial T_{i+1}}{\partial T}\right]. \qquad (11.20)\end{aligned}$$

Several of these terms are already derived in Section 11.3 (where $\partial T_i/\partial T$ and $\partial x_i/\partial x$ are identical terms). The temperature derivative of the absorption ($\partial k_i/\partial T_i$) must be calculated numerically. This derivative depends on if gas species concentrations are given as volume mixing ratio (VMR) or number density. As ARTS uses VMR, the term above will be calculated with the constrain of fixed VMR. A consequence of this is that it is not possible to mix retrieval of temperature and number densities.

Eq. 3.10 gives

$$\frac{\partial \tau_i}{\partial \Delta s_i} = \frac{k_i + k_{i+1}}{2}. \qquad (11.21)$$

The path length (Δs_i), for a given pressure, is linearly proportional to the temperature, and if $\bar{T}$ is the average temperature along the path step:

$$\frac{\partial \Delta s_i}{\partial \bar{T}} = \frac{\Delta s_i}{\bar{T}} \qquad (11.22)$$

Following the other variables, we set $\bar{T} = (T_i + T_{i+1})/2$, and

$$\frac{\partial \Delta s_i}{\partial T_i} = \frac{\Delta s_i}{2T_i}. \qquad (11.23)$$

Going back to Equation 11.19, the terms involving $\bar{B}_i$ are

$$\frac{\partial I}{\partial \bar{B}_i} = e^{-\tau'}\left(1 - e^{-\tau_i}\right), \qquad (11.24)$$

$$\frac{\partial \bar{B}_i}{\partial T} = \frac{1}{2}\left[\frac{\partial B}{\partial T_i}\frac{\partial T_i}{\partial T} + \frac{\partial B}{\partial T_{i+1}}\frac{\partial T_{i+1}}{\partial T}\right]. \qquad (11.25)$$

The term $\partial B(T)/\partial T_i$ can be expressed analytically [*Eriksson et al.*, 2002], but as B is provided by an agenda this term must anyhow be determined in a pure numerical manner; it is likely the derivative of the Planck function, but if the Rayleigh-Jeans approximation is for some reason selected ($B = T$) the term is just 1.

For higher Stokes components, or if emission is ignored:

$$\frac{\partial I}{\partial \bar{B}_i} = 0 \quad \text{and} \qquad (11.26)$$

$$\frac{\partial I}{\partial \tau_i} = -e^{-\tau'} I_i e^{-\tau_i}. \qquad (11.27)$$

Hydrostatic equilibrium and limitations

A changed temperature has non-local effects, originating from refraction and hydrostatic equilibrium. The expressions above ignore totally refraction effects.

If `hse` is set to "off", the term $\partial\Delta s_i/\partial T_i$ is set to zero. That is, the path length through the layer is not affected by a temperature change. With `hse` set to "on", the complete expressions above are used.

If this treatment of hydrostatic equilibrium is sufficient or not depends on the observation geometry. It is not sufficient for e.g. limb sounding, where changes even at altitudes below the tangent point will have an influence as the geometrical altitudes of all higher layers is changed through hydrostatic equilibrium. This coupling between temperature changes at altitudes below the tangent point originates from the geometrical calculations, mapping the sensor viewing angles to a propagation path through the atmosphere. If the sensor's viewing direction instead would have been specified as a tangent pressure, there would not be such a coupling. However, this effect vanishes for ground-based observations at zenith and satellite measurements at nadir, giving a full account of hydrostatic equilibrium even with the analytical expressions. In practise it should be possible to apply the expressions outside zenith and nadir, as long as the observations are of "up" or "down-ward" type. The same applies to measurements from inside the atmosphere, (e.g. aircraft ones), if the reference pressure for hydrostatic equilibrium (p_hse) is matched to the pressure of the observation point.

11.5 Sensor pointing

The term "sensor pointing" pointing refers to deviations between nominal and actual viewing direction of the sensor. So far, only deviations in zenith angle can be considered. The workspace method to initiate such WFs is jacobianAddPointingZa.

The pointing deviation is treated as a time varying variable, then having a polynomial variation. hence, the basis functions described in Section 11.2.2 are applied. The time is taken from sensor_time. If the pointing error is assumed to be constant with time, the polynomial order to select is 0, and so on. As a special case, the polynomial order -1 signifies here that the pointing off-set is totally uncorrelated between the spectra (sometime called "pointing jitter").

The WFs can be calculated in two manners:

recalc If this option is selected, radiative transfer calculations are performed for a shift of sensor_los (perturbation size selected by `dza`). Only a "one-sided" perturbation is applied.

interp The WFs are derived from existing data, by an interpolation of existing data. This achieved by interpolating pencil beam data to a shifted zenith angle grid. This will involve some extrapolation of the data, and this aspect should be considered when selecting mblock_za_grid. The average of a positive and negative shift. The shift to apply (`dza`) should be smaller than the minimum spacing of mblock_za_grid for accurate results. As interpolation is a relative fast operation and a "double-sided" disturbance is used, this option should in general be preferred.

11.6 Sensor frequencies

This class of WFs treats deviations between nominal and actually recorded frequencies. Such differences can originate in several ways, but the exact origin can normally be ignored and the effect can be modelled as the backend (spectrometer, filter bank ...) channels are shifted from their nominal position. The workspace method to define these WFs is jacobianAddFreqShiftAndStretch. The WFs can so far only be determined by applying an interpolation of existing monochromatic data ("interp"), then shifted `df` from the nominal values.

The method operates with two terms, "shift" and "stretch". This follows standard nomenclature. A "shift" is an off-set that is of the same size for all backend channels. That is, if only a shift is assumed, the nominal distances between the channels are assumed to be valid. The shift term is always included. The "stretch" term considers the distance between the channels. For a backend with all channels equally spaced, a stretch signifies that the spacing deviates from the nominal value (but all channels still equally spaced). More generally, a stretch means that the deviation from the nominal channel position increases linearly from the middle point of the backend. In terms of the basis functions Section 11.2.2, shift and stretch correspond to polynomial order 0 and 1.

11.7 Polynomial baseline fit

A "baseline" is microwave jargon for a disturbance of the spectrum that is not covered by the common sensor characteristics. The most common case is that the local oscillator signal leaks into the measurement by reflections occurring inside the sensor, causing a pattern in the spectrum of standing-wave type. Such effects are difficult to model in a physical manner, and a more general fitting procedure must be applied. A common option is then to model the baseline as a polynomial, of a specified order. That is, assuming a measurement giving a single spectrum, the measured spectrum $\mathbf{y}$ is modelled as

$$\mathbf{y} = \mathbf{y}' + \sum_{i=0}^{n} x_i \mathbf{z}_i, \tag{11.28}$$

where $\mathbf{y}'$ is the "baseline free" spectrum.

WFs for such baseline models are obtained by jacobianAddPolyfit. For single spectra measurements, the only consideration is the polynomial order to use. For measurements where several spectra are appended to form the measurement vector, the default option is that baseline can vary between all spectra. In some cases, it could be assumed that the baseline is common between data for different polarisations, viewing directions or measurement blocks, and flags can be set to mimic such assumptions.

Chapter 12

Sensor characteristics

A sensor model is needed because a practical instrument gives consistently spectra deviating from the hypothetical monochromatic pencil beam spectra provided by the atmospheric part of the forward model. For a radio (heterodyne) instrument, the most influential sensor parts are the antenna, the mixer, the sideband filter and the spectrometer. The response (as a function of frequency, zenith angle ...) of such instrument parts is here denoted as the sensor characteristics.

The treatment of sensor variables is introduced in Section 3.7. As described in that section, the position and line-of-sight (sensor_pos and sensor_los) are treated separately, while remaining sensor characteristics are summarised by a "response matrix" denoted as sensor_response. This matrix is applied following Equation 3.16. The purpose of this section is to describe how data on sensor characteristics are inputted to obtain a sensor_response that matches your particular instrument.

The implementation follows closely *Eriksson et al.* [2006]. That article provides also a more careful description of the approach of applying a response matrix, and the equations used to convert different type of characteristics to response values. As a user of ARTS, the main practical issue is to understand the different file formats used for the different sensor parts. For the moment, this is only described mainly through the on-line documentation.

12.1 General

In principle, a sensor must always be specified. However, if this shall be a hypothetical sensor just providing the monochromatic pencil beam data coming out of the atmospheric radiative transfer calculations, use sensorOff (sensor_response is in this case just an identity matrix).

For other cases, the definition of the sensor characteristics is initiated by calling sensor_responseInit. The natural order to call the main functions for the different sensor parts should be to follow the radiation through the instrument. That is, the antenna should normally be the first part to consider. If the order can be changed depends on the conditions. For example, for a double side-band receiver the antenna must be considered before the

History

110826 A simple version by Patrick Eriksson.

mixer, if the antenna response differs between the two bands. If the same antenna response can be assumed for both bands, the same result is obtained even if the mixer is introduced before the antenna.

Each response is defined for some grid. All responses are assumed to be zero outside the range covered by the grid, even if the end values deviate from zero. A positive aspect of this definition is that it is possible to define true “rectangular” responses. This is achieved by setting the end points of the grid where the response drops to zero.

The sensor parts are normally associated with some loss of power, and sensor contains also some amplification device. In general, it is not needed to consider these aspects, as such effects are cancelled out by the calibration process. The sensor should then be modelled as having no losses, which is ensured by setting sensor_norm to 1. The different responses are then normalised in an appropriate manner. With sensor_norm set to 0, all normalisation issues must be handled when defining the response files.

Part I

Bibliography and Appendices

Bibliography

Anderson, G. P., S. A. Clough, F. X. Kneizys, J. H. Chetwynd, and E. P. Shettle, AFGL atmospheric constituent profiles (0–120 km), *Tech. Rep. TR-86-0110*, AFGL, 1986.

Balluch, M., and D. Lary, Refraction and atmospheric photochemistry, *Journal of Geophysical Research*, *102*, 8845–8854, 1997.

Buehler, S. A., P. Eriksson, T. Kuhn, A. von Engeln, and C. Verdes, ARTS, the Atmospheric Radiative Transfer Simulator, *Journal of Quantitative Spectroscopy and Radiative Transfer*, *91*, 65–93, 2005.

Buehler, S. A., A. von Engeln, E. Brocard, V. O. John, T. Kuhn, and P. Eriksson, Recent developments in the line-by-line modeling of outgoing longwave radiation, *Journal of Quantitative Spectroscopy and Radiative Transfer*, *98*, 446–457, 2006.

Buehler, S. A., V. O. John, A. Kottayil, M. Milz, and P. Eriksson, Efficient radiative transfer simulations for a broadband infrared radiometer - Combining a weighted mean of representative frequencies approach with frequency selection by simulated annealing, *Journal of Quantitative Spectroscopy and Radiative Transfer*, *111*, 602–615, 2010.

Buehler, S. A., P. Eriksson, and O. Lemke, Absorption lookup tables in the radiative transfer model arts, *Journal of Quantitative Spectroscopy and Radiative Transfer*, *112*, 1559–1567, 2011.

Czekala, H., Microwave radiative transfer calculations with multiple scattering by non-spherical hydrometeors, Ph.D. thesis, Rheinische Friedrich-Wilhems-Universität Bonn, Auf dem Hügel 20, 53121 Bonn, 1999.

Davis, C., C. Emde, and R. Harwood, A 3D polarized reversed monte carlo radiative transfer model for mm and sub-mm passive remote sensing in cloudy atmospheres, *IEEE Transactions on Geoscience and Remote Sensing*, *43*, 1096–1101, 2005.

Emde, C., A polarized discrete ordinate scatterig model for radiative transfer simulations in spherical atmospheres with thermal source, Ph.D. thesis, University of Bremen, 2005.

Emde, C., and T. R. Sreerekha, Development of a RT model for frequencies between 200 and 1000 GHz, WP1.2 Model Review, *Tech. rep.*, ESTEC Contract No AO/1-4320/03/NL/FF, 2004.

Emde, C., S. A. Buehler, C. Davis, P. Eriksson, T. R. Sreerekha, and C. Teichmann, A polarized discrete ordinate scattering model for simulations of limb and nadir longwave measurements in 1D/3D spherical atmospheres, *Journal of Geophysical Research*, *109*, 2004.

Eriksson, P., Analysis and comparison of two linear regularization methods for passive atmospheric observations, *Journal of Geophysical Research*, *105(D14)*, 18157–18167, 2000.

Eriksson, P., F. Merino, D. Murtagh, P. Baron, P. Ricaud, and J. de La Noë, Studies for the Odin sub-millimetre radiometer: 1. Radiative transfer and instrument simulation, *Canadian Journal of Physics*, *80*, 321–340, 2002.

Eriksson, P., M. Ekström, S. Bühler, and C. Melzheimer, Efficient forward modelling by matrix representation of sensor responses, *International Journal of Remote Sensing*, *27*, 1793–1808, 2006.

Eriksson, P., S. A. Buehler, C. P. Davis, C. Emde, and O. Lemke, ARTS, the atmospheric radiative transfer simulator, version 2, *Journal of Quantitative Spectroscopy and Radiative Transfer*, *112*, 1551–1558, 2011.

Evans, K. F., S. J. Walter, A. J. Heymsfield, and M. N. Deeter, Modeling of submillimeter passive remote sensing of cirrus clouds, *J. Appl. Met.*, *37*, 184–205, 1998.

John, V. O., S. A. Buehler, A. von Engeln, P. Eriksson, T. Kuhn, E. Brocard, and G. Koenig-Langlo, Understanding the variability of clear-sky outgoing long-wave radiation based on ship-based temperature and water vapor measurements, *Quarterly Journal of the Royal Meteorological Society*, *132*, 2675–2691, 2006.

Kyle, T., *Atmospheric transmission, emission and scaterring*, Pergamon Press, 1991.

McFarquhar, G. M., and A. J. Heymsfield, Parametritation of tropical cirrus ice crystal size distributions and implications for radiative transfer: Results from CEPEX, *Journal of the Atmospheric Sciences*, *54*, 2187–2200, 1997.

Mishchenko, M. I., Calculation of the amplitude matrix for a nonspherical particle in a fixed orientation, *Appl. Opt.*, pp. 1026–1031, 2000.

Mishchenko, M. I., and L. D. Travis, Capabilities and limitations of a current fortran implementation of the t-matrix method for randomly oriented rotationally symmetric scatterers, *J. Quant. Spectrosc. Radiat. Transfer*, *60*, 309–324, 1998.

Mishchenko, M. I., L. D. Travis, and A. A. Lacis, *Scattering, Absorption and Emission of Light by Small Particles*, Cambridge University Press, 2002, ISBN 0-521-78252.

Montenbruck, O., and E. Gill, *Satellite orbits: Models, methods and applications*, Springer Verlag, 2000.

Read, W. G., Z. Shippony, M. J. Schwartz, N. J. Livesey, and W. V. Snyder, The clear-cky unpolarized forward model for the EOS Aura Microwave Limb Sounder (MLS), *IEEE Transactions on Geoscience and Remote Sensing*, *44*, 1367–1379, 2006.

Rodgers, C., *Inverse methods for atmospheric sounding: Theory and practise*, 1st ed., World Scientific Publishing, 2000.

Rodgers, C. D., Characterization and error analysis of profiles retrieved from remote sounding measurements, *Journal of Geophysical Research*, *95*, 5587–5595, 1990.

Rosenkranz, P. W., Absorption of microwaves by atmospheric gases, in *Atmospheric remote sensing by microwave radiometry*, edited by M. A. Janssen, pp. 37–90, John Wiley & Sons, Inc., 1993, `ftp://mesa.mit.edu/phil/lbl_rt`.

Rothman, L. S., et al., The HITRAN molecular spectroscopic database and HAWKS (HITRAN atmospheric workstation): 1996 edition, *Journal of Quantitative Spectroscopy and Radiative Transfer*, *60*, 665–710, 1998.

Part II

Index

Index