ARTS User Guide

edited by

Patrick Eriksson and Stefan Buehler

November 9, 2023
ARTS Version 2.5.12 (git: e212f7cd-dirty)

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 https://www.radiativetransfer.org/contact/.

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)uni-hamburg.de`**.**

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 https://www.radiativetransfer.org/docs/.

Contributing authors

Author/email	Main contribution(s)
Stefan Buehler[a] sbuehler (at) uni-hamburg.de	Editor, Sections 1, 6 and 17.
Claudia Emde[c] claudia.emde (at) dlr.de	Sections 8 and 18.
Patrick Eriksson[b] patrick.eriksson (at) chalmers.se	Editor, Sections 3, 4, 5, 7, 9, 10, 11, 12, 16, 13, 14, 15, and 20.
Oliver Lemke[a] olemke (at) core-dump.info	Latex fixes.
Simon Pfreundschuh[b]	Section **??**.

The present address is given for active contributors, while for others the address to the institute where the work was performed is given:

[a] Meteorological Institute, University of Hamburg, Bundesstr. 55, 20146 Hamburg, Germany.

[b] Department of Space, Earth and Environment, Chalmers University of Technology, SE-41296 Gothenburg, Sweden.

[c] Institute of Environmental Physics, University of Bremen, P.O. Box 33044, 28334 Bremen, Germany.

Contents

Part I

Overview

Chapter 1

Introduction

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). However, ARTS can be used for basic studies of lonwave radiation fluxes, as for example in *Buehler et al.* [2006] or *John et al.* [2006]. More lately, some support for handling radio link calculations have been added.

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.

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 concentrations etc.) can be assumed to only vary in the vertical dimension (1D), to

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.

have no longitude variation (2D) or vary in all three spatial dimensions (3D).

The surface is by default assumed to be spherical. For 2D and 3D, a complete reference ellipsoid is used and the surface can have arbitrary shape.

Polarisation is fully described by using the Stokes formalism.

Scattering can be considered in several manners. Extinction from scatterers can be included in transmission type calculations. For radiance calculations of thermal emission (in contrast to solar radiation) there are two modules at hand to take the scattering into account: DOIT (Chapter 18) and MC (Chapter 19). The scattering particles are for efficiency reasons confined to a region of the atmosphere denoted as the cloudbox.

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; they are explained in more depth further below). The easiest way to access this documentation is on the web page https://atmtools.github.io/arts-docs-2.6/workspace.html. 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.
FIXME: Control file structure changed, update this section!

1.2.5 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.

1.2.6 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
```

```
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 Environment variables

Environment variables can be used to control the behaviour of ARTS:

- `ARTS_DATA_PATH` List of search paths for data files. The `-D` commandline option takes precedence over this variable.
- `ARTS_INCLUDE_PATH` List of search paths for include files. The paths will also be searched for data files. The `-I` commandline option takes precedence over this variable.
- `ARTS_HEADLESS` If set, ARTS will not display any graphical interface elements. Mostly useful for testing.

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 https://atmtools.github.io/arts-docs-2.6/stubs/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`.)

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.

```
*----------------------------------------------------------*
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 https://atmtools.github.io/arts-docs-2.6/stubs/methods/WriteXML.

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.

Alternatively, workspace methods can be called with named arguments. All omitted

```
*----------------------------------------------------------------*
Workspace variable = propmat_clearsky_agenda
------------------------------------------------------------------

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 *propmat_clearsky*, the
atmospheric state has to be specified by *rtp_pressure*,
*rtp_temperature*, *rtp_mag*, and *rtp_vmr*.

The methods inside this agenda may require a lot of additional
input variables, such as *f_grid*, *species*, etc.
------------------------------------------------------------------
Group  = Agenda
Output = propmat_clearsky
Input  = f_grid, rtp_doppler, rtp_mag, rtp_pressure,
         rtp_temperature, rtp_vmr
*----------------------------------------------------------------*
```

```
AgendaSet(
propmat_clearsky_agenda)
{
  propmat_clearskyInit
  propmat_clearskyAddXsecAgenda
  propmat_clearskyAddZeeman
}
```

```
AgendaSet(
propmat_clearsky_agenda)
{
  Ignore(rtp_mag)
  propmat_clearskyInit
  propmat_clearskyAddFromLookup
}
```

Figure 1.4: Top: Built-in documentation for variable propmat_clearsky_agenda, obtained by command 'arts -d propmat_clearsky_agenda', or on page https://atmtools.github.io/arts-docs-2.6/stubs/pyarts.workspace.Workspace.propmat_clearsky_agenda.html. 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.

arguments will be set to their default values. `WriteXML( in=v )` is equivalent to calling `WriteXML( output_file_format, v )`. Note that named arguments can not be mixed with positional arguments.

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 method calls. It can be executed, which means that the method calls it contains are executed one after another.

Figure 1.4 gives an example, for the agenda propmat_clearsky_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 propmat_clearsky in its list of output workspace variables, at least one workspace method which generates propmat_clearsky 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 `controlfiles` 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. The file `agendas.arts` is also useful, because it predefines agendas with settings suitable for many applications. **FIXME: Revise this section after general.arts has been changed.**

1.5 Test controlfiles

The directory `controlfiles` 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 build directory, 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 GitHub 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.)

FIXME: The tests are now structured differently. Update the text.

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

Importing and exporting data

Sorry, so far just a few words about supported data formats.

FIXME: Extend this chapter.

2.1 Data formats

2.1.1 XML files

XML is the default file format for exchanging data with ARTS. Two flavors are supported: Plain text and binary. In the plain format all data is stored in the XML file. For binary, the structure of the data is stored in the XML file and the data itself in binary format in a separate file.

2.1.2 NetCDF files

NetCDF input and output is supported for a subset of the data types available in ARTS.

2.1.3 Gridded fields

Naming convention for grids

```
% Grid names for GriddedField variables.
% Keep the description short and on one line.
% Sort alphabetically.

Complex         Complex number "grid" (i.e. exactly 2 elements: real and imaginary)
Frequency       Frequency dimension for spectral dependent fields.
Latitude        Latitude dimension for atmospheric fields (z,T,VMRs,winds,Ne,B,pnd)
Longitude       Longitude dimension for atmospheric fields (z,T,VMRs,winds,Ne,B,pnd
Pressure        Pressure dimension for atmospheric fields (z,T,VMRs,winds,Ne,B,pnd)
```

History

? ?.

```
% Data field names
% mere suggestions and not tested anywhere in ARTS
Temperature     Temperature field
VMR             Volume mixing ratio field
Altitude        Altitude/height field
pnd_field       Particle number density field
```

Chapter 3

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 (Sec. 3.1). The main vertical coordinate is pressure and atmospheric quantities are defined as a function of pressure (Sec. 3.3), but the effective vertical coordinate from a geometrical perspective (such as the determination of propagation paths) is the radius (Sec. 3.2). Pressures and radii are linked by specifying the geometrical altitudes (z_field).

3.1 Altitude coordinates

Pressure The main altitude coordinate is pressure. This is most clearly manifested by the fact that the vertical atmospheric grid consists of equal-pressure levels. The vertical grid is accordingly 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.

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.

Geometrical altitude The term geometrical altitude signifies here the difference in radius between a point and the reference ellipsoid (Sec. 11.1) along the vector to the centre of the coordinate system (Equation 11.6). This is consistent with the usage of geocentric latitudes (see below). Hence, the altitude is not measured along the local zenith direction.

History

130219 Revised by Patrick Eriksson.

020315 First version by Patrick Eriksson.

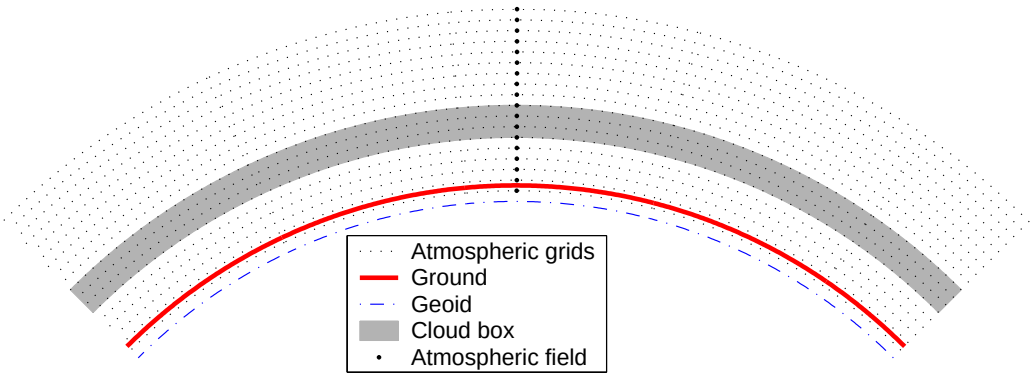


Figure 3.1: Schematic of a 1D atmosphere. The atmosphere is here spherically symmetric. This means that the radius of the ellipsoid, the surface and all the pressure levels are constant around the globe. The fields are specified by a value for each pressure level. The extension of the cloud box is either from the surface up to a pressure level, or between two pressure levels (which is the case shown in the figure). The figure indicates also that the surface must be above the lowermost pressure level. ("Geoid" in the legend should be "Ellipsoid".)

3.2 Atmospheric dimensionality

The structure of the modelled atmosphere can be selected to have different degree of complexity, the atmospheric dimensionality. There exist three options, 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.

1D A 1D atmosphere can be described as being spherically symmetric. 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. The latitude and longitude of the sensor are normally of no concern, but when required the sensor is considered to be placed at latitude and longitude zero ($[\alpha, \beta] = [0, 0]$). The sensor is assumed to by directed towards the North pole, corresponding to an azimuth angle of 0°. A 1D atmosphere is shown in Figure 3.1.

2D In contrast to the 1D and 3D cases, a 2D atmosphere is only strictly defined inside a plane. More in detail, this case be seen as observations restricted to the plane where the longitude equals 0° or 180°. A polar system, consisting of a radial and an angular coordinate, is applied. The angular coordinate is denoted as latitude, and matches the 3D latitude in the range $[-90°, +90°]$, but for 2D there is no lower or upper limit for the latitude coordinate. The 2D case is most likely used for satellite measurements where the atmosphere is observed inside the orbit plane. The 2D "latitude" can then

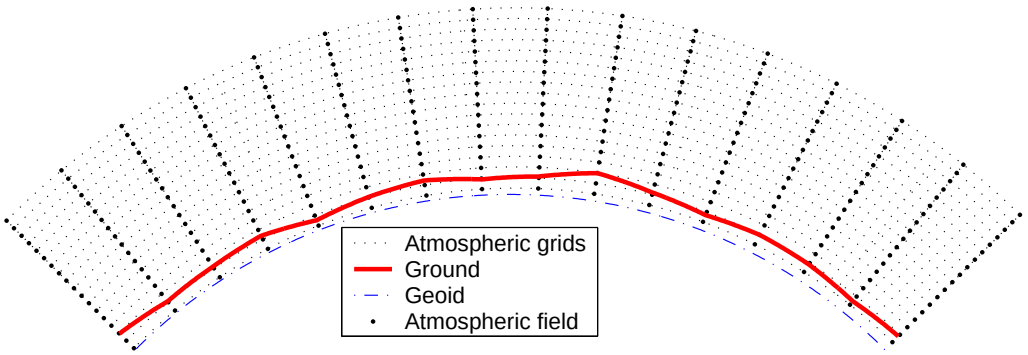


Figure 3.2: Schematic of a 2D atmosphere. The radii (for the surface and the pressure levels) vary here linear between the latitude grid points. The atmospheric fields vary linearly along the pressure levels and the latitude grid points (that is, along the dotted lines). Inside the grid cells, the fields have a bi-linear variation. No cloud box is included in this figure. ("Geoid" in the legend should be "Ellipsoid".)

be taken as the angular distance along the satellite track. A 2D-latitude of e.g. 100° will then correspond to a 3D-latitude of 80°. The atmosphere is normally treated to be undefined outside the considered plane, but some scattering calculations may treat the surrounding atmosphere in an simplified manner. A 2D atmosphere is shown in Figure 3.2.

3D In this, the most general, case, the atmospheric fields vary in all three spatial coordinates, as in a true atmosphere (Figures 3.3). 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. The valid range for latitudes is $[-90°, +90°]$, 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 360° then the whole globe is considered to be covered. The user must ensure that the atmospheric fields for β and $\beta + 360°$ are equal. If a point of propagation path is found to be outside the range of the longitude 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.

3.3 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 3.2. The

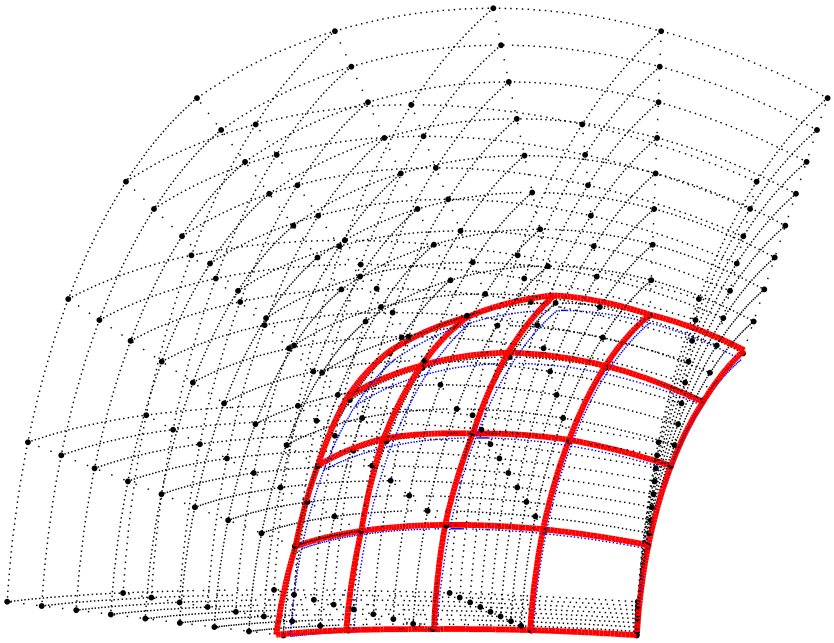


Figure 3.3: Schematic of a 3D atmosphere. Plotting symbols as in Figure 3.2. Radii and fields are here defined to vary linearly along the latitude and longitude grid points. This means that the radius of a pressure level has a bi-linear variation inside the area limited by two latitude and longitude grid values, while the atmospheric fields have a tri-linear variation inside the grid cells.

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 (Sec. 11.1.1). This is in accordance to the definition of geometric altitudes (Sec. 3.1). For 3D, a longitude grid (lon_grid) must also be specified. Valid ranges for latitude and longitude values are given in Section 3.2. 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) must be set to be empty.

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 9.2. The volume covered by the grids is denoted as the model atmosphere.

The basic atmospheric quantities are represented by their values at each crossing of the involved grids (indicated by thick dots in Figure 3.2), or for 1D at each pressure level (thick dots in Figure 3.1). 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 content and units of vmr_field are discussed in Section 4.2.2.

All the fields are assumed to be piece-wise linear functions vertically (with pressure altitude as the vertical coordinate), 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 Section 16.2, where the concept of basis functions is introduced. In short, the basis functions give the mapping from the set of discrete values to the continuous representation of the quantity.

3.4 Geo-location of 1D and 2D

For 1D and 2D atmospheres, lat_grid and lon_grid do not contain true geographical positions (they are either empty or lat_grid contains transformed data). However, some operations require that the positions is known, and this is handled by lat_true and lon_true. See the built-in documentation for further information on how to specify these variables.

3.5 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. The method considers that gravitation varies with latitude and altitude, and lat_true and lon_true must be set for 1D and 2D.

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.

3.6 The reference ellipsoid and the surface

Geometrical altitudes are specified as the vertical distance to the reference ellipsoid (refellipsoid), discussed further in Section 11.1. 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.

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.

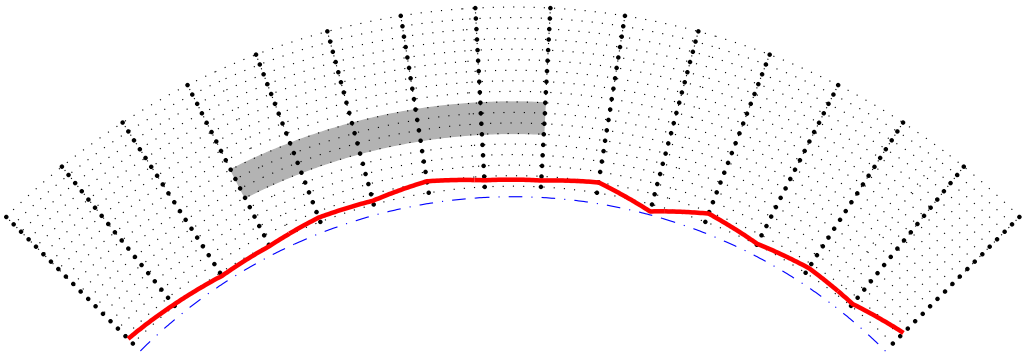


Figure 3.4: A latitudinal, or longitudinal, cross section of a 3D atmosphere. Plotting symbols as in Figure 3.1. Radii and fields inside the cross section match the definitions for 2D. The vertical extension of the cloud box is defined identical for 1D and 3D. The horizontal extension of the cloud box is between two latitude and longitude grid positions, where only one of the dimensions is visible in this figure.

3.7 The cloud box

In order to save computational time, calculations involving scattering are limited to a special atmospheric domain. This atmospheric region is denoted as the cloud box. The distribution of scattering matter inside the cloud box is specified by pnd_field, see further Section 4.2.2.

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 (Fig. 3.4). 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.

FIXME: add info on available cloudbox setting methods. particularly manual and automatic methods and which fields the latter look at

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.

3.8 Wind vector fields

The atmospheric fields discussed above are scalar quantities, while some atmospheric variables can be seen as vector fields. However, in ARTS input vector fields are broken down into the zonal, meridional and vertical components and are given as three scalar fields. This division into scalar values is used to allow that one or several of the components easily can be set to zero, which is done by setting the corresponding workspace variable to be empty. Following the standard naming scheme for winds, the components are denoted as

u The zonal component. A positive value signifies an Eastward direction.

v The meridional component. A positive value signifies a Northward direction.

w The vertical component. A positive value signifies an upward direction.

The workspace variables to describe the wind vector field are wind_u_field, wind_v_field and wind_w_field. To clarify the definition of the vector components above, the winds components are defined as follows

v_u A positive wind is defined as air moving from west to east, i.e. towards higher longitudes.

v_v A positive wind is defined as air moving from south to north, i.e. towards higher latitudes.

v_w A positive wind is defined as air moving upwards, i.e. towards higher altitudes.

As described above, one, two or all of these variables can be set to be empty, if the corresponding wind component is zero.

Winds affect the radiative transfer by inducing Doppler shifts, see further Chapter 13. Note that v_u causes no Doppler shift for 1D and 2D atmospheric setups. Considering the sensor viewing conventions in the 1D atmosphere case as laid out in Sec. 3.2, positive v_v correspond to tail winds, negative to head winds.

3.9 Magnetic field vector fields

To consider Faraday rotation (Sec. 14) and Zeeman splitting (Sec. 6.5.6), the magnetic field must be specified. The same strategy of specifying the vector field by three scalar components is applied as for the winds field (see above).

The three component fields are mag_u_field, mag_v_field and mag_w_field. All three components must be specified (but can be set to zero for a part of, or the complete, atmosphere). However, some component can be irrelevant for the calculations. For example, the u-component has no influence on Faraday rotation for 1D and 2D cases. (The internal representation of the magnetic field at a specific point is handled by rtp_mag. For this variable the three components are stored together, and thus the local magnetic field is represented as a vector.)

Chapter 4

Radiative transfer basics

This chapter introduces some basic radiative transfer nomenclature and equations. The radiative transfer equation is here presented in general terms, while special cases and solutions are discussed in later parts of the document.

4.1 Stokes dimensionality

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

$$\mathbf{s} = \begin{bmatrix} I \\ Q \\ U \\ V \end{bmatrix}, \tag{4.1}$$

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

$$I = I_v + I_h = I_{+45^\circ} + I_{-45^\circ} = I_{lhc} + I_{rhc}, \tag{4.2}$$

$$Q = I_v - I_h, \tag{4.3}$$

$$U = I_{+45^\circ} - I_{-45^\circ}, \tag{4.4}$$

$$V = I_{lhc} - I_{rhc}, \tag{4.5}$$

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 5.

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 particle

History
130218 First version by Patrick Eriksson.

and surface scattering have to be considered it is sufficient to only include the first Stokes components in the simulations (ie. stokes_dim set to 1). In this case, to include higher order Stokes components results only in slower calculations. Simulations where stokes_dim is two or higher are denoted as vector radiative transfer, while scalar radiative transfer refers to the case when only the first Stokes component is considered.

4.2 The radiative transfer equation

The radiative transfer problem can only be expressed in a general manner as a differential equation. One version for vector radiative transfer is

$$\frac{\mathrm{d}\mathbf{s}(\nu,\mathbf{r},\hat{\mathbf{n}})}{\mathrm{d}l} = -\mathbf{K}(\nu,\mathbf{r},\hat{\mathbf{n}})\mathbf{s}(\nu,\mathbf{r},\hat{\mathbf{n}}) + \mathbf{j}_e(\nu,\mathbf{r},\hat{\mathbf{n}}) + \mathbf{j}_s(\nu,\mathbf{r},\hat{\mathbf{n}}), \tag{4.6}$$

where ν is frequency, $\mathbf{r}$ represents the atmospheric position, $\hat{\mathbf{n}}$ is the propagation direction (at $\mathbf{r}$), l is distance along $\hat{\mathbf{n}}$, $\mathbf{K}$ is the propagation matrix, $\mathbf{j}_e$ represents the emission at the point, and $\mathbf{j}_s$ covers the scattering from other directions into the propagation direction.

4.2.1 Propagation effects

Three mechanisms contribute to the elements of the propagation matrix: absorption, scattering and magneto-optical effects. Absorption and scattering can together be denoted as extinction, referring to that these two mechanisms result in a decrease of the intensity (I). A common name for $\mathbf{K}$ is also the extinction matrix. The extinction processes also affect the Q, U and V elements of the Stokes vector. If the degree of polarisation (p)

$$p = \frac{\sqrt{Q^2+U^2+V^2}}{I} \tag{4.7}$$

is kept constant or not depend on symmetry properties of the attenuating media. For example, absorption of atmospheric gases is not changing p as long as the molecules have no preferred orientation, which is a valid assumption beside when there is a significant Zeeman effect. Non-polarising absorption corresponds to that the propgation matrix can be written as $\alpha\mathbf{1}$, where α is the absorption coefficient and $\mathbf{1}$ is the identity matrix.

There are also examples on effects that change the polarisation state of the Stokes vector without affecting I. These effects are caused by an interaction with the magnetic field, and are thus denoted magneto-optical. Ionospheric Faraday rotation can (approximately) be seen as a pure magneto-optical effect, while the Zeeman effect cause both non-isotropic absorption and has magneto-optical aspects.

As Equation 4.6 indicates, there are two source mechanisms that can act to increase the intensity: emission and scattering.

4.2.2 Absorbing species and scattering particles

The complexity of the radiative transfer is largely dependent on whether scattering must be considered or not. For this reason, ARTS operates with two classes of atmospheric matter:

Absorbing species This class covers atmospheric matter for which scattering can be neglected. The set of "species" to consider is described by the workspace variable abs_species, and the associated atmospheric fields are gathered into vmr_field. As the

name indicates, this later variable is mainly containing volume mixing ratio (VMR) data, but as this unit is not applicable in all cases also other units are accepted. That is, the unit for the fields varies, for gases it is VMR, while particle mass concentrations are given in kg/m^3 and electron density in m^{-3}.

Particles This second class treats all matter causing significant scattering (and likely also adding to the absorption). The amount of scattering matter is given as particle number density fields (m^{-3}), denoted as pnd_field. The pnd_field is provided per scattering element (see Chapter 8 for definition). The corresponding optical properties of the particles are given by scat_data containing one set of single scattering properties for each scattering element. Particles can be grouped into scattering species, each characterized by a mass density (kg/m^3) or flux ($kg/(m^2 s)$ field that is converted into particle number density fields before solving the radiative transfer equation.

Atmospheric quantities are not hard-coded to belong to any of these matter classes, as the practical division depends on the conditions of the simulations. The general rule is that for the shorter the wavelengths, a higher number of atmospheric constituents must be treated as "particles". For thermal infrared and microwave calculations, molecules and electrons (and likely also aerosols) can be treated as absorbing species. It can also be possible to place some hydrometeors in this class. For example, non-precipitating liquid clouds can be treated as purely absorbing in the microwave region. Related to the division between absorbing species and particles is:

Clear-sky In ARTS, the term "clear-sky" refers to the case when the influence of "particles" can be ignored, and only "absorbing species" are of relevance. Hence, a clear-sky calculation can involve e.g. cloud water droplets, but on the condition that the wavelength is such that scattering can be neglected.

The propagation effects of absorbing species and of (scattering) particles are kept separated. The total propagation matrix is

$$\mathbf{K} = \mathbf{A}_a + \mathbf{K}_p, \tag{4.8}$$

where a and p refers to absorbing species and particles, respectively, and the symbol $\mathbf{A}$ is used for the first term to remind about that the only extinction process covered by this matrix is absorption (but including magneto-optical effects). As workspace variable, $\mathbf{A}_a$ is denoted as propmat_clearsky, obtained by the propmat_clearsky_agenda agenda. $\mathbf{K}_p$ is obtained internally from scat_data.

4.2.3 Emission and absorption vectors

One of the general assumptions in ARTS is that local thermodynamic equilibrium (LTE) can be assumed. For the moment there exists no method in ARTS to handle deviations from LTE, but this can be changed in the future. On the condition of LTE, the emission vector ($\mathbf{j}_e$) in Equation 4.6 can be written as

$$\mathbf{j}_e = B\mathbf{a}, \tag{4.9}$$

where B is the Planck function (a scalar value), describing blackbody radiation. For non-LTE, the emission vector is instead

$$\mathbf{j}_e = B\mathbf{a}_s, \tag{4.10}$$

where the source term $\mathbf{a}_s$ contains both LTE and non-LTE effects. We code non-LTE not as above but with the concept of "relative additional" non-LTE effects. This additional non-LTE effect is named nlte_source and it has the form

$$\mathbf{j}_n = B\left(\mathbf{a}_s \oslash \mathbf{a} - 1\right). \tag{4.11}$$

(Note that the operator means that we are using element-wise division. Below, the dot means element-wise multiplication.) The idea here is that $\mathbf{j}_e$ in Equation 4.13 is never directly used but instead

$$\mathbf{j}_e = \mathbf{a} \odot \left(B + \mathbf{j}_n\right). \tag{4.12}$$

The quantity $\mathbf{a}$ is denoted as the absorption vector. For clear-sky calculations the absorption vector is given by $\mathbf{A}_a$, as in this case the emission vector can be calculated as

$$\mathbf{j}_e = \mathbf{A}_a\mathbf{b}, \tag{4.13}$$

where $\mathbf{b}$ can be seen as the emission source vector, defined as

$$\mathbf{b} = [B, 0, 0, 0]^T. \tag{4.14}$$

Hence, as only the first element of $\mathbf{b}$ is non-zero, the absorption vector is in this case equal to the first column of $\mathbf{A}_a$. The absorption vector can not be extracted from $\mathbf{K}_p$ as this propagation matrix covers also scattering and scat_data must also contain such data.

In summary, the total absorption vector in ARTS is obtained as

$$\mathbf{a} = \mathbf{a}_a + \mathbf{a}_p \tag{4.15}$$

where $\mathbf{a}_a$ is the first column of $\mathbf{A}_a$ and $\mathbf{a}_p$ is the absorption vector due to particles.

4.2.4 Main cases

The two equations below are discussed thoroughly in 6 of *ARTS Theory*. This including that for some conditions also the "n2-law of radiance" must be considered to obtain completely exact results (see also Section 9.5).

The equations below treat a single frequency and a single direction, at a time, and can be said to describe monochromatic pencil beam radiative transfer. For simplicity, the frequency and direction are left out from many of the equations in this user guide.

Clear-sky radiative transfer

If we for the moment assume that scattering can be totally neglected, then Equation 4.6 can be simplified to

$$\frac{\mathrm{d}\mathbf{s}(\nu, \mathbf{r}, \hat{\mathbf{n}})}{\mathrm{d}l} = \mathbf{A}_a(\nu, \mathbf{r}, \hat{\mathbf{n}})\left[\mathbf{b}(\nu, \mathbf{r}, \hat{\mathbf{n}}) - \mathbf{s}(\nu, \mathbf{r}, \hat{\mathbf{n}})\right] \quad \left(= -\mathbf{A}_a\mathbf{s} + B\mathbf{a}_a\right). \tag{4.16}$$

Cases where Equation 4.16 is valid, are in ARTS denoted as clear-sky radiative transfer (implying LTE if nothing else is stated). The discussion of such radiative transfer calculations is continued in Section 9.

Radiative transfer with scattering

Some additional conditions are required to put scattering into the picture. If scattering is of incoherent and elastic nature, the extension of Equation 4.16 is

$$\begin{aligned}\frac{\mathrm{d}\mathbf{s}(\nu,\mathbf{r},\hat{\mathbf{n}})}{\mathrm{d}l} &= -\mathbf{K}(\nu,\mathbf{r},\hat{\mathbf{n}})\mathbf{s}(\nu,\mathbf{r},\hat{\mathbf{n}}) + B\mathbf{a}(\nu,\mathbf{r},\hat{\mathbf{n}}) + \\ &\quad + \int_{4\pi} \mathbf{Z}(\nu,\mathbf{r},\hat{\mathbf{n}},\hat{\mathbf{n}}')\mathbf{s}(\nu,\mathbf{r},\hat{\mathbf{n}}')\,\mathrm{d}\hat{\mathbf{n}}',\end{aligned} \tag{4.17}$$

where $\mathbf{Z}$ is the scattering (or phase) matrix. ARTS includes several modules to handle scattering, introduced in the last part of this document.

Chapter 5

Complete calculations

This chapter outlines how complete radiative transfer simulations are performed. ARTS is not only performing atmospheric radiative transfer, also sensor characteristics can be considered. As a consequence, the topics of this chapter include the distinction between monochromatic pencil beam and "full" calculations, and how the sensor is introduced.

5.1 Overview

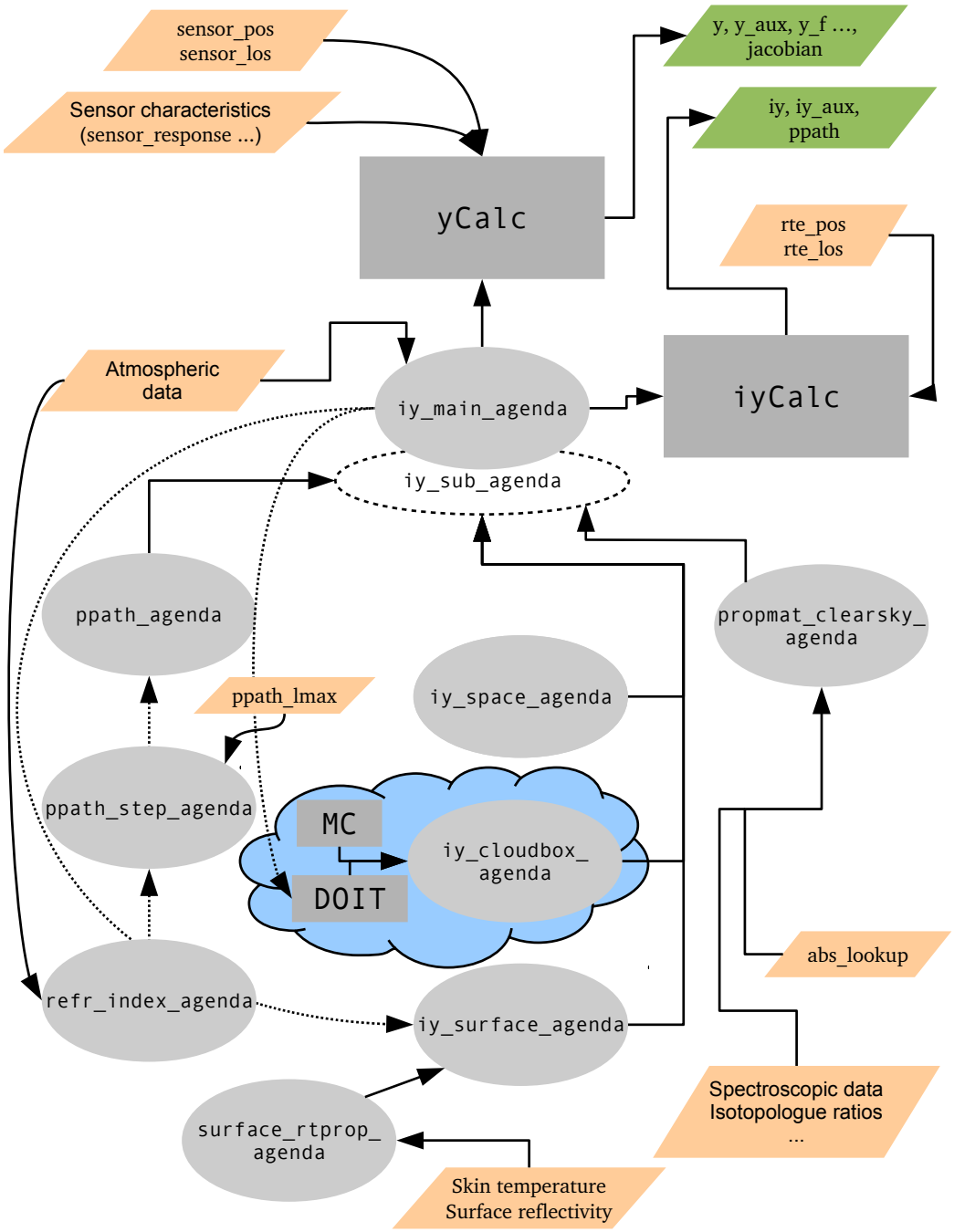
An attempt to illustrate a "standard" ARTS calculation is found in Figure 5.1. The figure shows that most calculation tasks are handled by an agenda. For example, ppath_agenda has the task of determining the propagation path for the given observation geometry. In principle, the agenda could solve the task by loading data from a file, but most likely it will use some of the dedicated workspace methods. These workspace methods are targeted towards different observation types, for example, radio link calculations require special consideration. Anyhow, the main message here is that by using the agenda concept, a very high degree of flexibility can be achieved and new features can be added fairly simply. On the other hand, the concept require that the user actually apply methods that make sense for the agenda. The code of ARTS performs some consistency checks of the agenda output, but this can only catch some types of mistakes.

Complete radiative transfer calculations are normally performed by yCalc. This method incorporates sensor responses and has the variable y as main output. The letter y here refers to the measurement vector, $\mathbf{y}$, found in the formalism of *Rodgers* [1990, 2000] (see also Sec. 1.3 of *ARTS Theory*). The vector can hold anything from a single value, to a high number of spectra appended. The spectra in the last case can correspond to a limb sounding sequence (hence measured for different zenith angles), or even be measured by different sensors. In any case, the data in $\mathbf{y}$ contain likely significant impact of different parts of the sensor used, such as the angular weighting by the antenna pattern.

On the other hand, atmospheric radiative transfer is performed for monochromatic frequencies, along pencil beam directions. The outcome of one such calculation is the Stokes vector for each frequency considered, and as workspace variable this quantity is denoted as iy. Please, note the distinction to y, that can contain information from a number of monochromatic pencil beam calculations, as shown in Section 5.3.

History

130219 First version by Patrick Eriksson.



(For caption see top of the next page.)

Figure 5.1: A flowchart of a radiative transfer calculation (on previous page), using yCalc or iyCalc. The figure assumes that iy_main_agenda holds iyEmissionStandard, that represents the most complex case. Some important input data are shown in orange, and main output are shown in green. The two main methods are plotted as grey boxes, while agendas are shown as ovals. For connections between methods and agendas the arrows show the direction of output (calculated) data. Every call of an agenda involves also some input data (settings), but this aspect has been ignored for clarity reasons. Data entering an agenda along a common line indicates mutually independent options. For example, absorption can be calculated "on-the-fly" from basic spectroscopic data or be extracted from a pre-calculated look-up table (see bottom-right corner). The dotted lines indicate that some methods and agendas can make further calls of iy_main_agenda.

Further, the term "iy" is not restricted to the final outcome of atmospheric radiative transfer, it is also used to indicate that quantities and operations are of monochromatic pencil beam character. For example, the agenda returning the radiation entering the atmosphere from space is named as iy_space_agenda to indicate that the agenda output is directly associated with the calculation of iy. If only a single iy is of concern, the radiative transfer can be performed by iyCalc, having a smaller set of input arguments than yCalc.

The output is not restricted to y or iy, also some auxiliary data can be extracted as described in Section 9.8. In addition, yCalc can also provide weighting functions (Chapter 16).

5.2 Compulsory sensor and data reduction variables

The instrument real or hypothetical, 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 and mblock_dlos. These variables are presented separately below.

5.2.1 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 geometrical altitude, latitude and longitude. Accordingly, row i of sensor_pos for a 3D case is (z_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 9.2). 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.

One or several spectra can be calculated for each position as described in Section 5.3. The corresponding workspace variable for single pencil beam calculations is rte_pos, that is an input argument to e.g. iy_main_agenda and iyCalc.

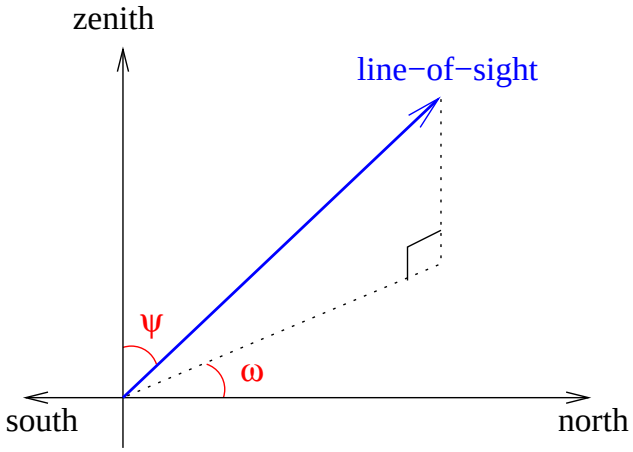


Figure 5.2: Definition of zenith angle, ψ, and azimuth angle, ω, for a line-of-sight. The figure shows a line-of-sight with a negative azimuth angle.

5.2.2 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 9.2). The zenith and azimuth 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 5.2). 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 11.1.1). 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. The correspondance to rte_pos is rte_los.

5.2.3 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 of 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 and mblock_dlos 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.

See further Chapter 12, where inclusion of sensor characteristics is described in .

5.3 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. A measurement block can be treated as a measurement cycle that is repeated, an integer number of times, to form the measurement 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 altitude 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.

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

$$\mathbf{i}_b = \begin{bmatrix} \begin{bmatrix} \nu_1 \\ \vdots \\ \nu_n \end{bmatrix} \\ \vdots \\ \begin{bmatrix} \nu_1 \\ \vdots \\ \nu_n \end{bmatrix} \end{bmatrix} \tag{5.1}$$

where ν_i is element i of f_grid and n the length of the same vector. The order of the angles inside mblock_dlos is followed when looping the pencil beam directions

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

$$\mathbf{y} = \begin{bmatrix} \mathbf{H}_b\mathbf{i}_{b,1} \\ \mathbf{H}_b\mathbf{i}_{b,2} \\ \vdots \\ \mathbf{H}_b\mathbf{i}_{b,n} \end{bmatrix} \tag{5.2}$$

where 1 indicates the first sensor position etc.

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 $\mathbf{H}_b$ 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 $\mathbf{H}_b$ 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.

Part II

Atmospheric properties

Chapter 6

Gas absorption

6.1 Introduction

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

This chapter discusses different practical aspects of absorption in ARTS. Sections 6.2 and 6.3 introduce the main physical quantities and the main agendas, respectively. Section 6.4 explains how absorption is handled inside radiative transfer calculations. Section 6.5 discusses how absorption is actually calculated, and how the calculation is set up. Finally, Section 6.6 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.

6.2 Key physical quantities

The scalar gas absorption coefficient α has units of 1/m. You can think of it as being defined by the Lambert-Beer law

$$I_1 = I_0 e^{-\alpha l}, \tag{6.1}$$

History	
2013-06-21	New intro and general revision. Added CIA part. — Stefan Buehler
2012-08-28	Updated to propmat_clearsky — by Richard Larsson.
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. — Stefan Buehler
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 6.1: 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
α	m^{-1}		Scalar gas absorption coefficient
I	$\frac{\mathrm{W}}{\mathrm{m}^2\ \mathrm{Hz}\ \mathrm{sr}}$	iy	Intensity
l	m		Path length element
n_i	molec/m^{-3}		Number density of species i
$\kappa_{i,j}$	m^5/molec^2	abs_cia_data	Binary absorption cross-section of absorbing species pair (i,j)
$\mathbf{K}$	m^{-1} (4×4)	propmat_clearsky	Clear-sky propagation matrix

where I is intensity and l is the distance through a homogeneous medium with absorption coefficient α.

Absorption is additive, so the total absorption is the sum of the partial absorptions of all absorbers. For an individual absorber, we can define another important quantity, the absorption cross-section κ_i, as

$$\kappa_i = \frac{\alpha_i}{n_i}, \tag{6.2}$$

where subscript i denotes the absorber and n_i is the partial number density of that absorber. Absorption cross-sections depend less strongly on pressure than absorption coefficients, and are therefore more suitable for storing in a lookup table.

Some processes create polarized absorption, which is described by a 4×4 matrix, in ARTS called propagation matrix (propmat_clearsky, already introduced in Section 4.2). This variable is used by the ARTS RT functions to describe absorption. In the absence of polarizing effects it is simply equal to $\alpha\mathbf{1}_4$, where $\mathbf{1}_4$ is the 4×4 identity matrix.

Both, gases and particles in the atmosphere absorb, and the total absorption is the sum of these two contributions. This chapter only deals with absorption of non-scattering matter, i.e., gas absorption in the first place (but can also include absorption by grey-body particles and polarization changes by electrons).

6.3 Agendas

There is one key agenda related to absorption in ARTS: propmat_clearsky_agenda is called by RT methods when they need absorption (more precisely the clear-sky propagation matrix, as defined above).

6.4 Gas absorption in radiative transfer simulations

The interface between the RT part of ARTS and the absorption part of ARTS is the agenda propmat_clearsky_agenda (see Figure 6.1). RT functions execute this agenda whenever they need local absorption matrices. In a typical ARTS run, the agenda will be executed many times over, for different points in the atmosphere. See the built-in documentation for the exact input and output arguments of the agenda. The idea is that input arguments are the local

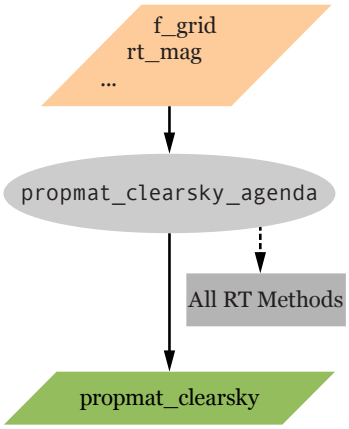


Figure 6.1: An outside view of propmat_clearsky_agenda.

atmospheric conditions (temperature, pressure, trace gas volume mixing ratios, magnetic field, etc.).

The output of the agenda is a single variable, propmat_clearsky, a tensor with dimensions of absorption species, frequency, Stokes dimension and Stokes dimension (Stokes dimension of one thus emulates scalar absorption). The physical quantity corresponding to this variable is the clear-sky propagation matrix $\mathbf{A}_a$ as defined in Equation 4.8. It describes all non-scattering extinction effects, that is, absorption and related polarization effects.

The agenda can contain a number of different workspace methods that in some way or another compute propmat_clearsky. See the built-in documentation of the individual methods to learn more. File `agendas.arts`, one of the standard include-controlfiles, predefines some typical alternatives how propmat_clearsky_agenda can be set for different purposes.

The method propmat_clearsky_agendaAuto constructs a propmat_clearsky_agenda that contains all required absorption methods. The list of methods is determined automatically from the species present in abs_species. This agenda can then be used for RT calculation with one-the-fly absorption or the preparation of an absorption lookup table. The method propmat_clearsky_agendaAuto can also constructs a propmat_clearsky_agenda that uses a lookup table in RT calculations if given the correct flag (see method description).

6.5 Calculating gas absorption

This section deals with calculating gas absorption matrices in ARTS. This can typically occur in three different contexts: as on-the-fly absorption matrix calculation within the radiative transfer calculation (see above), when preparing a gas absorption lookup table (see Section 6.6), or when the user is only interested in the absorption itself (see Section 6.7).

In all these cases, the same agenda is used to actually calculate absorption: propmat_clearsky_agenda.

6.5.1 Absorption species

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

An absorption species in ARTS is an abstract entity that has a partial absorption matrix 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 abs_speciesSet.

The absorption species tags specify the different considered absorbers, which can be gaseous species but also free electrons and (grey-body) particles. For gaseous species, they also describe the model that should be used to calculate the absorption for each of the species. There are four types of tags, those for explicit line-by-line calculations, those for continua and complete absorption models, those for empirical cross section model, and a special Zeeman effect tag. 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. An example of the third is `"CFC11-XFIT"`, which identifies the empirical absorption cross section model of CFC11. An example of the fourth is `"O2-Z"`, which identifies that special Zeeman routines should be used. 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 isotopologue 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 isotopologue ratio (the isotopologue ratio can be changed instead; see Section 6.5.10). As an example, to make a line-by-line calculation for all ozone isotopologues, you could represent them in different ways by abs_speciesSet.

```
a) abs_speciesSet(species=["O3"])
b) abs_speciesSet(species=
                  ["O3-666, O3-668, O3-686, O3-667, O3-676"])
c) abs_speciesSet(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 all 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 abs_speciesSet gives a detailed explanation of the tag syntax and some examples.

Particularly note, that order of the species list matters as absorption line data is assigned to species in their order within the abs_species list and no line record is assigned to more than one species. It is furthermore important to note that there is no 'intelligence' in ARTS

that checks that the chosen tag combinations make sense, so the user should know what s/he is doing, or follow one of the many examples in the ARTS `controlfiles` directory.

6.5.2 Explicit line-by-line calculations

For absorption species with explicit line-by-line calculation the calculation involves the steps summarized in Table 6.2, which contains the steps that are common to all the three contexts in which explicit line-by-line calculations can occur as well as the steps that are specific to each of those cases. 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.

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. Further input data and parameters required (not only) for line-by-line calculations is described in Section 6.5.10.

6.5.3 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 6.2 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 abs_speciesSet. Currently available models are listed in Table 6.3.

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.

The core method to calculate continua and complete absorption models is propmat_clearskyAddPredefined. Users normally do not have to call this method explicitly, since it is used implicitly by higher level methods, such as propmat_clearsky_fieldCalc or the RT methods via propmat_clearsky_agenda, which is easiest set using propmat_clearsky_agendaAuto.

6.5.4 Collision-induced absorption

Collisions of centro-symmetric molecules, e.g., O_2, N_2, H_2, CO_2, and CH_4, possessing no permanent electric dipole create a transient dipole, which causes so-called collision-induced absorption (CIA). Absorption strength of CIA is characterized by its dependency on the molecular density of both molecular species involved in the collision.

Recently, the well-known HITRAN spectral line catalogue has started to offer also tabulated binary absorption cross-sections for CIA. This is described in detail in *Richard et al.* [2012], and also in the documentation that comes with the data themselves.

#	Step	Variables and Methods
1	Read spectral line data (the order of the first two steps does not matter).	Variable: abs_lines. Methods: ReadArrayOfARTSCAT (legacy catalog format), ReadARTSCAT (legacy catalog format), ReadHITRAN, ReadJPL, (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.
2	Split line data for different absorption species.	Variable: abs_lines_per_species. Methods: abs_lines_per_speciesCreateFromLines.
3	Optimize line data. (optional)	Variable: abs_lines_per_species. Methods: Add mirror lines for VVW line shape with abs_linesMirroring (see *ARTS Theory*, Chapter 2).
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.		
4a	Calculate absorption on-the-fly.	Agenda: propmat_clearsky_agenda. Variable: propmat_clearsky, which is initialized in propmat_clearskyInit. Methods: propmat_clearskyAddCIA computes collision-induced absorption, propmat_clearskyAddLines compute line-by-line absorption, propmat_clearskyAddZeeman (see Sec. 6.5.6), propmat_clearskyAddFaraday (see Sec. 6.5.8), propmat_clearskyAddParticles (see Sec. 6.5.9). propmat_clearskyAddPredefined computes legacy continua or full absorption models, Alternative: propmat_clearskyAddFromLookup (extract absorption from pre-calculated lookup table, see Sec. 6.6). Note that the lookup table cannot contain absorption for Zeeman tagged species, Faraday rotation, and particles due to their directional dependencies. Recommendation: propmat_clearsky_agendaAuto can be used to setup all propmat_clearsky_agenda calculations
4b	Calculate absorption lookup table.	Variable: abs_lookup. Methods: abs_lookupCalc. Alternative: Load lookup table from file with ReadXML, it then has to be adapted to the current calculation (and checked) with abs_lookupAdapt.
4c	Calculate absorption only (no RT).	Variable: propmat_clearsky_field. Methods, high level: propmat_clearsky_fieldCalc. Methods, low level: See 4a.

Table 6.2: 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-SelfContCKD222
	H2O-ForeignContCKD222
	H2O-SelfContCKD242
	H2O-ForeignContCKD242
	H2O-SelfContCKD24
	H2O-ForeignContCKD24
	H2O-SelfContCKDMT100
	H2O-ForeignContCKDMT100
	H2O-SelfContCKDMT252
	H2O-ForeignContCKDMT252
	H2O-SelfContCKDMT320
	H2O-ForeignContCKDMT320
	H2O-SelfContCKDMT350
	H2O-ForeignContCKDMT350
	H2O-ForeignContATM01
Complete water vapor models	H2O-CP98
	H2O-MPM87
	H2O-MPM89
	H2O-MPM93
	H2O-PWR98
Carbon dioxide continua	CO2-CKD241
	CO2-CKDMT100
	CO2-CKDMT252
	CO2-SelfContPWR93
	CO2-ForeignContPWR93
	CO2-SelfContHo66
	CO2-ForeignContHo66
Oxygen continua	O2-CIAfunCKDMT100
	O2-v0v0CKDMT100
	O2-v1v0CKDMT100
	O2-visCKDMT252
	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
	O2-TRE05
Nitrogen continua	N2-SelfContMPM93
	N2-SelfContPWR93
	N2-SelfContStandardType
	N2-SelfContBorysow
	N2-CIArotCKDMT100
	N2-CIAfunCKDMT100
	N2-CIArotCKDMT252
	N2-CIAfunCKDMT252
	N2-DryContATM01
Condensate absorption models	liquidcloud-MPM93
	icecloud-MPM93
	rain-MPM93

Table 6.3: 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.

Binary absorption cross-sections $\kappa_{i,j}$ have to be multiplied with the number densities of both involved molecular species to yield absorption coefficients:

$$\alpha_{i,j} = \kappa_{i,j}\, n_i\, n_j, \tag{6.3}$$

where i and j denote the two different absorbing species. As a consequence, $\kappa_{i,j}$ has units of $\mathrm{m}^5/\mathrm{molec}^2$ in ARTS (the original HITRAN units are different).

Using CIA in ARTS is easy. First of all, include one or more CIA tags in your absorption species list (abs_species). All valid tags are listed in Table 6.4. Secondly, read in WSV abs_cia_data, which contains the tabulated binary absorption cross-sections, from a file. This will usually be the file `hitran_cia2012_adapted.xml.gz`, which is included in the `arts-xml-data`, but the original HITRAN data files can also be read. Finally, use WSM propmat_clearskyAddCIA in propmat_clearsky_agenda to add the CIA absorption. For usage examples, look in directory `controlfiles/artscomponents/cia` that is part of the ARTS distribution.

Figure 6.2 shows all CIA continua that are currently available in ARTS (left) and separately the ones that are relevant for Earth's atmosphere (right). The valid frequency and temperature ranges for these data, as available in ARTS, are listed in Table 6.4. Outside the covered frequency ranges, the binary absorption cross-sections are set to zero, while exceeding the valid temperature range will produce `NaN` values and eventually trigger a runtime error.

To make the HITRAN data work in ARTS, some modifications were necessary, specifically:

N_2-N_2: The two high-frequency datasets were merged into one.

O_2-O_2: Three apparently separate datasets that really belong together were merged. UV/Vis dataset were removed.

CO_2-CO_2: Caveat: This is only the self continuum of CO2. The CO2-air continuum has strong features above 250 cm^{-1} that are present in CKD_MT (also available in ARTS as one of the continuum and full absorption models), but are missing here. Furthermore, *Richard et al.* [2012] point out that for molecules with more than two atoms further mechanisms affecting CIA exist, which are not covered by the simple models used. They hence state that "these data should be used very carefully". Conclusion: No changes, but use with care.

Also note that no data exists at frequencies below 30 GHz (1 cm^{-1}) though some significant absorption is still present at the limiting frequency. For those low frequencies, the CO2-SelfContPWR93 continuum (see Table 6.3) can be used as an alternative (we estimated that to be valid at least up to about 100 GHz, but deviating significantly above 500 GHz).

O2-N2, O2-CO2: These UV/Vis-only datasets were removed.

6.5.5 Absorption cross section model

The empirical absorption cross sections are calculated in the method propmat_clearskyAddXsecFit. If this method is included in the propmat_clearsky_agenda, then species with the tag "`-XFIT`" will be calculated as empirical cross section model

Table 6.4: Absorption species tags, frequency ranges, and temperature ranges for HITRAN CIA data as implemented in ARTS. (These data contain some modifications from the original HITRAN data, which are described in the text.)

CIA tag	Spectral range [cm^{-1}]	Temp range [K]	No. of datasets
N2-CIA-N2-0	0.02 – 554.00	40.00 – 400.00	10
N2-CIA-N2-1	1850.00 – 3000.09	228.20 – 362.50	10
N2-CIA-H2-0	0.02 – 1886.00	40.00 – 400.00	10
N2-CIA-CH4-0	0.02 – 1379.00	40.00 – 400.00	10
H2-CIA-H2-0	20.00 – 10000.00	200.00 – 3000.00	113
H2-CIA-He-0	20.00 – 20000.00	200.00 – 9900.00	334
H2-CIA-CH4-0	0.02 – 1946.00	40.00 – 400.00	10
H2-CIA-H-0	100.00 – 10000.00	1000.00 – 2500.00	4
He-CIA-H-0	50.00 – 11000.00	1500.00 – 10000.00	10
O2-CIA-O2-0	1150.00 – 1950.00	193.40 – 353.40	15
CO2-CIA-CO2-0	1.00 – 250.00	200.00 – 800.00	7
CH4-CIA-CH4-0	0.02 – 990.00	40.00 – 400.00	10
CH4-CIA-Ar-0	1.00 – 697.00	70.00 – 296.00	5

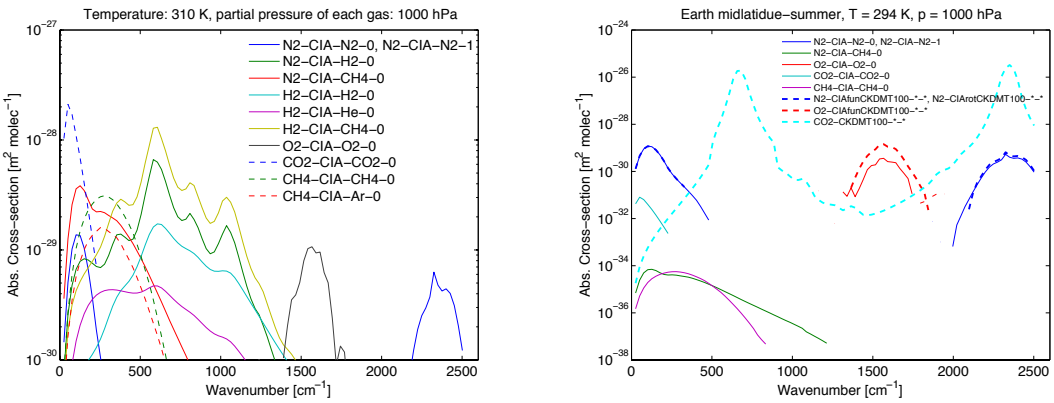


Figure 6.2: Left: All HITRAN CIA continua that are implemented in ARTS (each gas here has a partial pressure of 1000 hPa). Right: Only the ones that are relevant for Earth (for Earth surface conditions).

species. The absorption cross section model is based on HITRAN cross-section data, which are described in detail in *Gordon et al.* [2017]. For the details on the empirical cross section model, see *ARTS Theory*, Section 2.4.

6.5.6 Zeeman calculations

The Zeeman effect is calculated in the method propmat_clearskyAddZeeman. If this method is included in the propmat_clearsky_agenda, then species with the tag "`-Z`" will be calculated as Zeeman species. Note that the order within the tag string is important: the Zeeman tag must directly follow the molecular species tag. That is, `O2-Z-66` will be counted as Zeeman splitting on the $O^{16}O^{16}$ molecule.

The physics and internal workings of the Zeeman calculations follow the scheme presented in *Larsson et al.* [2014]. However, *Larsson et al.* [2014] only presents the correct solution for ground-state $^3\Sigma$ (e.g., molecular oxygen in millimeter). For other states, more appropriate expressions should be used. Also note that there is one additional error in *Larsson et al.* [2014], where the table representing the relative line strengths is incorrectly normalized to 2 rather than 1 for $\Delta J = 0$.

In order to calculate Zeeman splitting, additional line parameters are necessary. These are attached to the line-by-line. There are many ways to compute these. An update from *Larsson et al.* [2014] for O_2 is available in *Larsson et al.* [2019], and precise calculations for ClO, OH and NO is available in *Larsson and Lankhaar* [2020].

Beside the additional line parameters, it is also necessary to input the magnetic field into the model. This can be done either by calling MagFieldsCalcIGRF (earth only model), or by manually supplying raw magnetic field measurements on a gridded field and use the combination of MagRawRead and one of MagFieldsCalc, MagFieldsCalcExpand1D, or MagFieldsFromAltitudeRawCalc. Since altitude and not pressure is the natural coordinate for magnetic fields, it is required that z_field and/or the refellipsoid are defined. Note that it is possible but absolutely not recommended to use pressure as your altitude coordinate.

6.5.7 Internal line-mixing

Line mixing is provided together with the line-by-line data and is then computed naturally during line-by-line calculations. If no such data is provided, line mixing is not computed.

You can deactivate line mixing calculations manually using one of the methods that sets the line mixing limit. As an example, abs_linesLinemixingLimit sets the line mixing limit for all lines in abs_lines. Only at pressures higher than this line mixing limit will the line mixing calculations be available, so setting this much higher than the pressure will ensure that no line mixing is computed.

6.5.8 Faraday rotation

Faraday rotation is a change of polarization state of radiation in interaction with (free) electrons in presence of a static magnetic field. For further details on theory and usage in ARTS see Section 14. Here we only give a short summary how to setup the calculation of Faraday contribution to the absorption (or better: propagation) matrix propmat_clearsky.

First, to include Faraday rotation effects propmat_clearskyAddFaraday must be included in the propmat_clearsky_agenda. Second, a species tag "`free_electrons`" needs to be contained in abs_species. Correspondingly, a field of electron densities is required in vmr_field.

For usage examples, check `controlfiles/artscomponents/faraday` that is part of the ARTS distribution.

6.5.9 Absorbing particles

As pointed out before, this chapter deals with absorption by non-scattering matter. In first place this refers to gases, while particles (aerosols, clouds, precipitation) are considered to (also) scatter radiation and are handled differently (see Chapters 8, 18, and 19). However, when particles are small compared to the wavelength of the radiation they act as broadband grey-body absorbers and can be treated similarly to continuum absorption by gases.

This is reflected in ARTS providing a few continuum models for condensed matter (see Tab. 6.3), which essentially are particles, too. It is tedious, though, to implement those kind of particle continua for a wide range of different base materials as become of interest when being interested in other than the Earth's atmosphere.

In the ARTS scattering modules, particles are represented by single scattering property data (scat_data) and particle concentrations (particle number densitity fields pnd_field). The single scattering data originate from scattering theory programs (e.g., Mie theory, T-matrix model, Discrete Dipole Approximation) and their preparation typically requires significant efforts. Comprehensive data for hydrometeors in the Earth atmosphere, but also clouds, dust and the like for other planets is available from the `arts-xml-data` package. It is appealing to apply this data in non-scattering calculations (e.g. at low frequencies, where the scattering contribution is negligible) in a consistent manner. The ARTS method for that is propmat_clearskyAddParticles and its application is described in the following.

To consider grey-body particle absorption, the user has to include propmat_clearskyAddParticles in the propmat_clearsky_agenda. Furthermore, for each scattering element (see Section 8.1 for how a scattering element is defined) 1) a `"particles"` tag needs to be added to abs_species, 2) the corresponding concentration field has to be added to vmr_field, and 3) its single scattering data have to be added to scat_data. This can be done each-by-each using ReadXML and Append methods, but a dedicated method ScatElementsToabs_speciesAdd is available performing these three steps for one scattering element at once. ScatElementsToabs_speciesAdd adds the raw number density field to vmr_field_raw, i.e., the raw concentration fields can be converted to internal atmospheric grids together with the gas concentration fields using, e.g., AtmFieldsCalc. Single scattering data of all individual scattering elements is added to one and the same scattering species, specifically to the last one of these in the scat_data array. Note that ScatElementsToabs_speciesAdd is essentially doing the same as ScatElementsPndAndScatAdd, but for non-scattering instead for scattering-in-cloudbox cases (where in non-scattering setups the concentration data is stored together with gas concentrations in vmr_field, while for scattering setups it is stored separately in pnd_field), and that for the single scattering data and concentration fields the exact same data can be applied.

Beside being able to re-use particle data from scattering cases, this method is also advantageous compared to the particles-as-continuum-models implementations as it allows for directional dependent absorption and for polarization effects that occur, e.g., with non-spherical particles.

In the default case, absorption by particles is applied both in the extinction and emission terms of the radiative transfer equation, i.e. both right hand terms in Equation 4.16. However, using a flag, propmat_clearskyAddParticles can apply total particle extinction instead. It shall be noted, that while this applies the correct extinction, it also creates an unphysical

emission term. Hence, this option shall only be applied when the source term is negligible, as e.g. for occultation measurements.

Be aware that both propmat_clearskyAddParticles and the scattering methods use scat_data to store the particle single scattering data. Hence, it is straight-forward that these methods can not be applied simultaneously. In one ARTS run, all particles are handled either as scattering entities (when using the scattering modules) or as grey-body absorbers (when applying propmat_clearskyAddParticles). Trying to use both in parallel results in a runtime error.

For a setup example check `TestAbsParticle.arts` in `controlfiles/artscomponents/absorption/`. See the built-in documentation of the individual methods for further information.

6.5.10 Further input data and parameters for calculating gas absorption

Spectral line data

Important input to the line-by-line calculations is the spectral line data, usually provided by spectroscopic catalogues. ARTS has its own format for the spectral line data, but is also capable of handling data from other catalogues like HITRAN (both pre- and post-2004 formats) and JPL (see Table 6.2, step 1). Section **??** of *ARTS Theory* contains more information on the internal format of the spectral line data. It also contains theoretical background for the calculation itself.

Isotopologue ratios

Isotopologue ratios (mostly) from HITRAN and valid for Earth atmosphere are stored in ARTS source code. These data are necessary for working with HITRAN data (as HITRAN line strengths are weighted with isotopologue abundance). However, it is convenient for the user to be able to change isotopologue ratio values, e.g., when modeling absorption in other planets' atmospheres.

The WSV isotopologue_ratios holds the isotopologue ratios applied in the absorption calculation. They have to be set by the user. It is possible to apply the ARTS built-in values mentioned above using isotopologue_ratiosInitFromBuiltin. Alternatively, they can be read from file using ReadXML. For easy manipulation, the user might initialize isotopologue_ratios from built-in data, write the isotopologue_ratios structure to file using WriteXML, modify the data accordingly, and read in the manipulated file. Files with isotopologue ratios for a couple of planetary atmospheres are provided with the `arts-xml-data` package.

It shall be noted, that only isotopologue ratios of the species used in the absorption calculation need to be given. Reading in from file resets the full list of isotopologue ratios (i.e., the values for all absorption species known to ARTS) with species not given in the input data set to `NaN`.

Partition functions

Partition functions are compiled into ARTS from our distributed partition functions by default. Users also have the option to select their own partition functions via CMake build options. It is possible to check the partition functions that have been used by storing all of

the data to a directory using WriteBuiltinPartitionFunctionsXML (note that many files will likely be generated, so it is advised to write to an empty directory).

It is also possible to directly check the computed partition function values using the executable `partfun` that is generated as part of the standard build.

For more information on theoretical background as well as the source and implementation in ARTS see Section 2.1.3 of *ARTS Theory*.

6.6 The gas absorption lookup table

6.6.1 Introduction

Calculating gas absorption matrix 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 matrix depends not directly on position, but on the atmospheric state variables:

- Pressure
- Temperature
- Concentrations of absorbing matter (i.e., gases, absorbing particles, free electrons)
- Magnetic field

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. Due to the nature of the Zeeman and Faraday effects (also particle absorption), particularly due to their directional dependence, those are not implemented in the lookup table. Thus, we can ignore the magnetic field.

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].

6.6.2 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:

$$\frac{dI}{dl} = -I \sum_i \kappa_i n_i = -I \sum_i \alpha_i = -I \alpha_{\text{total}} \tag{6.4}$$

where the meaning of the symbols is defined in Table 6.1.

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.

[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.

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 memory (and computation time). 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 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.

6.6.3 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_lookupCalc. ARTS also includes some methods that automatically set input parameters for abs_lookupCalc, 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] as well as the built-in documentation 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 checks that all used species (apart from Zeeman, Faraday, and particle species) are present in the table, reduces the table to the used species, and sorts the table species data 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 matrices with the method propmat_clearskyAddFromLookup. This will extract *absorption matrices*, 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 propmat_clearskyAddFromLookup method is meant to be used inside the agenda propmat_clearsky_agenda, which is applied in several places where absorption matrices are needed, both inside the scattering box and outside.

6.6.4 Format of the lookup table

Usually the user does not need to bother with it, as ARTS provides methods to create, read and write, and extract data from the lookup table. However, sometimes one desires to analyze, e.g., the absorption cross-section data calculated and stored in the lookup table. Therefore we give a short description of the format of the absorption lookup table here. More detailed information can be found in the source code, where the `GasAbsLookup` class is implemented – specifically in `gas_abs_lookup.h`.

The absorption lookup table is a compound type variable comprising of (in this order; variable type of each entry shown in parantheses)

- **species:** an array of the species tags the lookup table is valid for (ArrayOfArrayOfSpeciesTag)
- **nonlinear_species:** an array indicating the species that require non-linear treatment (ArrayOfIndex)
- **f_grid:** the frequency grid (Vector)
- **p_grid:** the pressure grid (Vector)

- **vmrs_ref:** the reference profiles of volume mixing ratios (VMRs) for all species associated with the pressure (Matrix; dimension: [number of species, number of pressure levels])
- **t_ref:** the reference temperature profile associated with the pressure grid (Vector)
- **t_pert:** the temperature perturbations (Vector)
- **nls_pert:** the VMR perturbations of the non-linear species in terms of fractional units of the reference VMRs (Vector)
- **xsec:** the absorption cross-sections (Tensor4; dimension: [number of temperature perturbations, number of species (and non-linear species perturbations), number of frequencies, number of pressure levels])

6.7 Stand-alone gas absorption calculation

Within the RT calculations, gas absorption is calculated or extracted locally, i.e., for a specific point in the atmosphere or in other words for a specific set of pressure, temperature, and trace gas VMR. However, sometimes it is of interest to explicitly calculate and output absorption, e.g., for testing and validating modules of the absorption calculation, for model comparisons, for plotting and analyzing absorption coefficients, etc. Table 6.2, step 4c lists high- and low-level workspace methods for this purpose. In particular, the method propmat_clearsky_fieldCalc provides the absorption matrices, i.e., polarized absorption coefficients, per species tag group for an entire atmospheric scenario and the complete frequency grid.

Chapter 7

Refractive index

FIXME: Write a proper introduction. Comment on that this is not the complex refractive index, also covering absorption. ...

Refractive index (here restricted to the real part of the refractive index, which is basically a complex quantity with the imaginary part expressing absorption) describes several effects of matter on propagation of electromagnetic waves. This particularly includes changes of the propagation speed of electromagnetic waves, which leads to a delay of the signal as well as a change of the propagation direction, a bending of the propagation path. The latter is commonly called refraction.

Several components in the atmosphere contribute to refraction, hence to the refractive index: the gas mixture("air"), solid and liquid constituents (clouds, precipitation, aerosols), and electrons. ARTS includes mechanisms for deriving the contributions from gases and electrons with the available methods described in the Sections below. ARTS does not consider refraction by solid and liquid particle as their refractive index is not known to ARTS (when scattering is considered, ARTS gets their single scattering properties as input, see Chapter 8). However, the contribution of solid and liquid constituents to the refractive index is commonly neglected in radiative transfer models and is expected to have only small effects.

Refractivity (N) describes the deviation of the refractive index of a medium $n$from the vacuum refractive index ($n_{\rm vacuum} = 1$): $N = n - 1$. Contributions of the different components to refractivity are additive. Therefore, all ARTS methods that provide refractive index, calculate refractivity and sum it up with the input refractive index.

Within ARTS, refractive index is required for calculations of refracted propagation paths and related parameters (e.g., deriving viewing angle for a given tangent altitude). **FIXME: for more?**

Whenever refractive index is required, e.g., at each point along a propagation path, it is evaluated according to the mechanism specified by refr_index_air_agenda. refr_index_air_agenda provides both the monochromatic refractive index refr_index_air, in the following denoted as n, as well as the group refractive index refr_index_air_group, denoted as n_g. refr_index_air differs from refr_index_air_group in case of dispersion, which e.g. leads to diverging propagation paths at different frequencies.

FIXME: Describe where each variable is used. Expand ...

History

120918 Started (Patrick Eriksson).

7.1 Gases

For calculating the contribution to the refractive index from atmospheric gases, the following workspace methods are currently available in ARTS: refr_index_airMicrowavesGeneral, refr_index_airMicrowavesEarth and refr_index_airInfraredEarth. All of them are non-dispersive, i.e., monochromatic and group refractive index are identical. They are supposed to be applied as alternatives, not in addition to each other.

refr_index_airMicrowavesGeneral provides refractivity due to different gas mixtures as occuring in planetary atmospheres and is valid in the microwave spectral region. It uses the methodology introduced by *Newell and Baird* [1965] for calculating refractivity of the gas mixture at actual pressure and temperature conditions based on the refractivity of the individual gases at reference conditions. Reference refractivities from *Newell and Baird* [1965] are available for N_2, O_2, CO_2, H_2, and He. Additionally, reference refractivity for H_2O has been derived from H_2O contribution as described by refr_index_airMicrowavesEarth (see below) for a reference temperature of T_0=273.15 K[1]. Any mixture of these gases can be taken into account. The missing contribution from further gases is roughly accounted for by normalising the calculated refractivity from the six reference gases to a volume mixing ratio of 1. More details on the applied formulas are given in Section 4.1.1 of *ARTS Theory*.

refr_index_airMicrowavesEarth calculates the microwave "air" refractivity in the Earth's atmosphere taking into account refractivity of "dry air" and water vapour. All other gases are assumed to have a negligible contribution.

refr_index_airInfraredEarth derives the infrared "air" refractivity in the Earth's atmosphere considering only refractivity of "dry air".

7.2 Free electrons

Free electrons, as exist in the ionosphere, affect propagating radio waves in several ways. Free electrons will have an impact of the propagation speed of radio waves, hence a signal can be delayed and refracted. This section consideres only the refraction effect (neglecting influences of any magnetic field). For effects on polarisation state of the waves in presence of a static magnetic field, i.e., Faraday rotation, see Section 14.

refr_index_airFreeElectrons derives this contribution of free electrons to the refractive index. The method is only valid when the radiative transfer frequency is large enough (at least twice the plasma frequency). Information on theoretical background and details on the applied formulas are provided in Section 4.2 of *ARTS Theory*.

[1] Reducing the refr_index_airMicrowavesEarth approach to inverse temperature proportionality as applied by refr_index_airMicrowavesGeneral causes significant deviations from H_2O refractivity from refr_index_airMicrowavesEarth. However, they are smaller than when refraction by H_2O is neglected.

Chapter 8

Description of scattering media

8.1 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 *scattering elements*.

A scattering element is either a specific single particle or a particle ensemble, e.g., an ensemble following a certain size or shape distribution. The scattering element can represent particles that are completely randomly oriented, azimuthally randomly oriented or arbitrarily oriented. Each scattering element is characterized by its single scattering properties (SSP) and a field of particle number densities. For each grid point in the cloud box, the atmospheric volume that encloses all scattering particles, the single scattering properties of all scattering elements weighted by their respective particle number density at this location are summed up to derive the cloud ensemble optical properties.

The scat_data structure contains the single scattering properties ($\langle \mathbf{K}_i \rangle$, $\langle \mathbf{a}_i \rangle$, and $\langle \mathbf{Z}_i \rangle$) for each of the scattering elements. In scat_data, the SSP are stored in different coordinate systems, depending on the kind of particle. For instance, SSP of totally randomly oriented particles are stored in the so-called scattering frame in order to reduce memory requirements, while others use the particle frame. Section 8.2 describes in detail the `SingleScatteringData` class and section 8.3 presents options for preparing SSP. For details on the coordinate systems used also see *ARTS Theory* Section 6.2.6 (Note: Coordinate systems used by the SSP as well as the line-of-sight and radiation propagation directions in the scattering solvers and the general radiative transfer part are not fully consistent. However, as long as only totally or azimuthally randomly oriented particles and only up to two Stokes components are considered this is of no practical concern. **FIXME: add appendix on known ARTS issues with a section on this particular one**).

The number density field, pnd_field, contains the number densities of all scattering elements at all grid points within the cloudbox. pnd_field can be read in from externally prepared data files or be derived from mass density of flux fields provided to the model.

History	
050913	Created and written by Claudia Emde
161107	Extended regarding internal calculations of scat_data and pnd_fields by Jana Mendrok

Section 8.4 describes the available options.

The WSVs scat_data and pnd_field together exhaustively describe particle ensembles in ARTS. All ARTS scattering solvers consistently require and use these two WSVs to derive the bulk extinction, absorption, and phase matrices ($\mathbf{K}_p$, $\mathbf{A}_p$, and $\mathbf{Z}$).

8.2 Single scattering properties

8.2.1 Scattering data structure

The single scattering data is stored in a specific structure format, the the `SingleScatteringData` class[1]. The format allows space reduction due to symmetry for certain special cases, e.g. totally random or azimuthally random orientation. The class consists of the following fields (compare also Table 8.1):

- *String* `ptype`: An attribute, which flags type and format of the data stored. It essentially describes the type of particle regarding its symmetry properties (totally randomly oriented, azimuthally 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:

  `"totally_random"`
  `"azimuthally_random"`
  `"general"`

  A detailed description of the different types including their different data formats and corrdinate systems used is given in Section 8.2.2.

- *String* `description`: Here, the scattering element is characterized explicitly in free text form. For example, information on the size and shape of the particle or the respective distributions of a particle ensemble might be given. 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. `"totally_random"`: Scattering angle grid. Range: $0.0° \leq \text{za} \leq 180.0°$.
  2. `"azimuthally_random"`: Zenith angle grid. Range: $0.0° \leq \text{za} \leq 180.0°$. Symmetric with respect to 90.0° and explicitly including the 90.0° point.
  3. `"general"`: Zenith angle grid. Range: $0.0° \leq \text{za} \leq 180.0°$.

- *Vector* `aa_grid`: Azimuth angle grid.

  1. `"totally_random"`: Empty (not needed, since optical properties depend solely on scattering angle).

[1] Definition resides in `optproperties.h`.

Symbol	Type	Dimensions	Description
	enum		ptype specification
	String		short description of the scattering element
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	(ν, T, ψ, ω, ψ', ω', i)	phase matrix
$\langle \mathbf{K} \rangle$	Tensor5	(ν, T, ψ, ω, i)	extinction matrix
$\langle \mathbf{a} \rangle$	Tensor5	(ν, T, ψ, ω, i)	absorption vector

Table 8.1: Structure of single scattering data files

2. `"azimuthally_random"`: Range: 0.0° ≤ aa ≤ 180.0° (due to symmetry only half of the general grid is required).
3. `"general"`: Range: -180.0° ≤ aa ≤ 180.0°

The angular grids have to satisfy the following conditions:

- Range limits (0°, 180°, ...) must be grid points.
- Data at azimuth angle values of -180.0° and 180.0° must be equal.

- *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 ptype of the data. For most types not all matrix elements need to be stored (see description of ptypes in Section 8.2.2).

- *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 ptype of the data.

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

  The dimensions are:

  ```
  [frequency temperature za_inc aa_inc vector_element]
  ```

  The absorption vector is explicitly given. It could be calculated from extinction matrix and phase matrix. However, this would take significant computation time, as it requires an angular integration over the phase matrix and poses integration accuracy and grid representation issues, whereas the additional storage burden is comparably low.

8.2.2 Definition of ptypes

Ptype essentially classifies the scattering elements regarding their symmetry properties, which are largely governed by particle orientation. As indicated above, this affects the optimal choice of the coordinate system to represent the scattering element in. Possible ptypes in the model are:

"totally_random"

The ptype value "totally_random" refers to macroscopically isotropic and mirror-symmetric scattering media. It covers totally randomly oriented particles (with at least one plane of symmetry[2]) as well as spherical particles.

For this type of scattering media, the optical properties are stored in the scattering frame (see *ARTS Theory* Section 6.2.6, where the z-axis corresponds to the incident direction and the x-z-plane with the scattering plane. Using this frame, only the scattering angle is needed, the angle between incident and scattered direction. Furthermore, the number of matrix elements of both the phase matrix and the extinction matrix can be reduced (see *Mishchenko et al.* [2002], p.90). This representation, however, requires an ARTS-internal transformation of the phase matrix data from the particle frame representation (in the scattering frame) to the laboratory frame representation. These transformations are described in the Appendix of *Emde* [2005].

Only six elements of the phase matrix in the scattering frame, which is commonly called scattering matrix $\mathbf{F}$, are independent. The order of the stored matrix elements is: *F11*, *F12*, *F22*, *F33*, *F34*, *F44*. The size of `pha_mat_data` is

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

The extinction matrix is in this case diagonal and independent of direction and polarization. That means only one element per frequency and temperature needs to be stored. 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` is

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

"azimuthally_random"

The ptype value "azimuthally_random" refers to particles that exhibit a preferred orientation with respect to polar angle, but are oriented randomly regarding the azimuthal angle. Horizontally aligned particles are one prominent example, but this class is not limited to them. For these particles, one angular dimension is redundant and can be omitted in the data, if the coordinate system is oriented appropriately.

SSP here are stored in the laboratory frame. In this frame, the phase matrix, extinction matrix, and absorption vector become independent of the incident azimuth angle. Furthermore only half of the "general" angle ranges are required for the azimuth angle due to symmetry reasons. That is, `aa_grid` from 0° to 180°.

All 16 elements of the phase matrix are required. Their order is the same as in the general case. The size of `pha_mat_data` is

[2]Note: Even if totally randomly oriented, particles lacking a symmetry plane are not part of this category unless their mirrored counterparts are considered with equal weight.

`[N_f N_T N_za_sca N_aa_sca N_za_sca 1 16]`

For this ptype, the extinction matrix has only three independent elements, *Kjj*, *K12(=K21)*, and *K34(=-K43)*. The size of `ext_mat_data` is

`[N_f N_T N_za 1 3]`

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

`[N_f N_T N_za 1 2]`

"general"

NOTE: While basic definitions and infrastructure to handle particles of general type exist, not all methods are fully implemented for this type. Hence, calculations with arbitrary particles are currently **not possible**.

The ptype value "general" refers to arbitrarily shaped and oriented particles. For those, generally no symmetries exist, hence all 16 elements of the phase matrix have to be stored for the complete set of incident and scattered directions. The matrix elements are stored in the order *Z11*, *Z12*, *Z13*, *Z14*, *Z21*, *Z22*, ... The size of `pha_mat_data` is

`[N_f N_T N_za_sca N_aa_sca N_za_inc N_aa_inc 16]`

Seven extinction matrix elements are independent (cp. *Mishchenko et al.* [2002], p.55). The elements being equal for single particles are equal for a distribution, too, as total extinction is derived by summing up individual contribution. Extinction matrix generally only depends on the incident direction, hence 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]`

8.3 Generating single scattering properties

The single scattering properties in the above described format have to be available to ARTS when entering the scattering solver. The data can be prepared by external tools or from internally interfaced methods.

Generally, single scattering properties can be calculated for example by Mie [e.g. *Wiscombe*, 1980; *Mätzler*, 2002], T-Matrix [*Mishchenko and Travis*, 1998] or Discrete dipole approximation, DDA, [e.g *Yurkin and Hoekstra*, 2011] methods. While Mie methods can only provide data for macroscopically isotropic particles, T-matrix methods are applicable for certain anisotropic particles, too, and DDA is capable of handling completely general particles.

The user is free to use any tools to derive the single scattering properties, but has to ensure to store the data in the appropriate format. That is, totally_random particle data has to be provided in the scattering frame, while the other two types are given in the laboratory frame. Depending on the method used, averaging the single scattering properties of particles in fixed orientation over a range of orientations might required.

Within ARTS and its supporting software packages some methods to prepare single scattering data are available:

- The WSM scat_data_singleTmatrix provides an ARTS internal interface to the the T-matrix code by *Mishchenko et al.* [2002]. It allows to calculate single scattering

properties of individual scattering elements including orientation averaging and reference frame adjustion. It is currently only available for a limited number of particle shapes (see online documentation) and orientations (namely for totally randomly oriented and perfectly horizontally aligned particles, but not for generally azimuthally randomly oriented ones).

- The ATMLAB package, available at https://www.radiativetransfer.org/tools/, includes functions to generate single scattering properties for spherical particles (Mie-Theory) as well as an interface to ARTS' own scat_data_singleTmatrix WSM.
- The Python module PyARTS used to provide methods to generate single scattering properties for horizontally aligned as well as for randomly oriented particles in the ARTS data-file-format. NOTE: PyARTS is currently not maintained and updated along with ARTS development. Hence, functionality of specific methods is not guaranteed.

Since WSVs scat_data and pnd_field form a paired input to ARTS' scattering solvers, dedicated methods exist for loading externally prepared data consistently into these WSVs. WSMs ScatElementsPndAndScatAdd and ScatSpeciesPndAndScatAdd can be applied for loading externally prepared single scattering data and particle number density field raw data (into pnd_field_raw) simultaneously. In case, particle scattering should be neglected and particles considered as absorbing species only (see Section 6.5.9), WSM ScatElementsToabs_speciesAdd should be used to load the data into scat_data and vmr_field_raw. If particle number density fields are calculated internally from atmospheric cloud fields (see Section 8.4.2, WSM ScatSpeciesScatAndMetaRead can be applied.

8.4 Generating particle number density fields

8.4.1 Externally created particle number density fields

Externally created particle number density fields have to be provided to ARTS in the form of GriddedField3 data. Using the WSMs ScatElementsPndAndScatAdd or ScatSpeciesPndAndScatAdd, they are loaded into the WSV pnd_field_raw simultaneously as the corresponding single scattering properties are loaded into scat_data. WSM ScatElementsPndAndScatAdd reads number density fields for individual scattering elements, i.e., GriddedField3 format data, and appends the data as one array element to pnd_field_raw, whereas ScatSpeciesPndAndScatAdd expects a set of number density fields in the form of ArrayOfGriddedField3, which is appended to pnd_field_raw.

Before entering a scattering solver, pnd_field has to be calculated from the raw number density fields using pnd_fieldCalcFrompnd_field_raw. The data for all scattering elements is regridded to the common RT calculation grids of pressure, latitude, and longitude and stored together in Tensor4 format. Particle number density fields are restricted to the cloudbox, i.e., pnd_field is only derived for the RT grid points within the cloudbox region. That is, the cloudbox region has to be defined before applying pnd_fieldCalcFrompnd_field_raw. For WSMs to set the cloudbox see Section 3.7.

8.4.2 Internal calculation of particle number density fields

ARTS offers the possibility to internally derive the pnd_field making use of atmospheric hydrometeor fields as provided by numerical weather or general circulation models.

FIXME: write. consider/discuss:

- related atm fields
- scat_species, PSDs
- scat_meta

This mechanism requires additional information about the scattering elements compared to scat_data, particularly measures used in the characterization of the hydrometeor ensembles, like mass and other size parameters. This information, also called the scattering meta data, is stored in the WSV scat_meta, where the external structure, that is the scattering species and scattering element hierarchy as well as the number of these instances) has to be identical to scat_data.

8.4.3 Scattering meta data structure

The scattering meta data is stored in a specific structure format, the the `ScatteringMetaData` class[3]. The class consists of the following fields (compare also the built-in doumentation of scat_meta_single):

- *String* `description`: Similar to the `description` field in scat_data, this gives a free-form description of the scattering element, e.g. information deemed of interest by the user but not covered by other structure members. It is only for informative purpose, i.e., not used within ARTS for any classifications or calculations.
- *String* `source`: Free-form description of the source of the data, e.g., Mie, T-Matrix, or DDA calculation or a database or literature source. As `description` only for informative purpose.
- *String* `refr_index`: Free-form description of the underlying complex refractive index data, e.g., a literature source. As `description` only for informative purpose.
- *Numeric* `mass`: Mass m of the scattering element [Unit: kg].
- *Numeric* `diameter_max`: Maximum diameter D_{max} of the scattering element [Unit: m].

  The maximum diameter (or dimension) is defined by the circumferential sphere diameter of the element. Note that this parameter is only used by some size distributions; it does not have a proper meaning if the scattering element represents an ensemble of differently sized particles.
- *Numeric* `diameter_volume_equ`: Volume equivalent sphere diameter D_{veq} of the scattering element [Unit: m].

  The volume equivalent sphere diameter is the diameter of a sphere with the same volume. For nonspherical particles, volume refers to the volume of the particle-forming substance, not that of the circumferential sphere (which can be derived from diameter_max).

  If the particle consists of a mixture of materials, the substance encompasses the complete mixture, but excluding inclusions and pockets of the the material, the particle

[3] Definition resides in `optproperties.h`.

is embedded in, typically air. Note: the air-material mixture of 'soft' particles forms a new substance, though; i.e. the soft particle volume should include both the basic material and air of the homogeneous mixture. However, this interpretation might be inconsistent with certain microphysical models. In this case, the user should rather ensure consistency of the D_{veq} setting with microphysics.

FIXME: emntion/discuss relation to mass equivalent diameter? melted diameter?

- *Numeric* `diameter_area_equ_aerodynamical`: Aerodynamical area equivalent sphere diameter of the scattering element [Unit: m].

  The area equivalent sphere diameter is the diameter of a sphere with the same cross-sectional area. Here, area refers to the aerodynamically active area, i.e., the cross-sectional area perpendicular to the falling direction. Similarly to volume in the definition of `diameter_volume_equ`, for non-spherical and mixed-material particles, area refers to the area covered by the substance mixture of the particle. The parameter might be relevant for characterizing the particle shape (preferred orientation) and fall velocity. However, it is so far unused within ARTS.

8.5 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`.

8.5.1 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 can use the method cloudboxSetManuallyAltitude:

```
cloudboxSetManuallyAltitude( cloudbox_on, cloudbox_limits,
                             atmosphere_dim, z_field,
                             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 scattering element is added in this example
ScatElementsPndAndScatAdd( scat_data, pnd_field_raw,
                 atmosphere_dim, f_grid,
```

```
                    ["ssd_sphere_50um_macroscopically_isotropic.xml",
                     "ssd_cylinder_30um_horizontally_aligned.xml"],
                    ["pnd_sphere_50um_macroscopically_isotropic.xml",
                     "pnd_cylinder_30um_horizontally_aligned.xml"] )
```

In the workspace method ScatElementsPndAndScatAdd the single scattering properties for individual (one or more) scattering elements are read and the scattering element data is appended to the last defined scattering species. The generic input `scat_data_filenames` holds the list of filenames of the datafiles containing the single scattering data per scattering element (class `SingleScatteringData`) in xml-format. The generic input `pnd_field_files` holds the list of filenames of the corresponding particle number density fields in xml-format (class `GField3`). Adding multiple scattering element instances can be done at once as demonstrated in the above example, where two elements, a sphere and and a horizontally aligned cylindrical particle, are added.

Alternatively it is possible to use the method ScatSpeciesPndAndScatAdd. It is, e.g., convenient to generate a size distribution using several size bins, where each scattering element constitutes one size bin. ScatSpeciesPndAndScatAdd creates a new scattering species and adds all data of all covered scattering elements to this species. It requires as input an array of string including the filenames of the single scattering data files for all individual scattering elements and the name of the data file holding the complete variable pnd_field_raw, i.e., the particle number density fields for all scattering elements at once. 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_fieldCalcFrompnd_field_raw:

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

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 18 and the Monte Carlo module in Chapter 19.

Part III

Radiative transfer: clear-sky + general functionality

Chapter 9

Clear-sky radiative transfer

This section discusses variables and the approach used to handle the actual radiative transfer calculations. This includes how effects caused by the sensor and surface are incorporated. Measurements of thermal emission in absence of particle scattering are used as example, and the basic theory for such simulations is also covered. The first ARTS version was developed for emission measurements, and such observations remain the standard case in ARTS.

A basic assumption for this chapter is thus that there is no particle scattering. This is denoted as clear-sky calculations. Scattering is restricted to the "cloud box" (Sec. 3.7). In short, the more demanding calculations are restricted to a smaller domain of the model atmosphere, and the radiative transfer in that domain is mainly treated by dedicated workspace methods. For pure transmission measurements (where scattering into the line-of-sight is neglected), see Chapter 15. This chapter discusses only the direct radiative transfer, partial derivatives (i.e. the Jacobian or weighting functions) are discussed in Section 16.

Absorption by atmospheric gases does normally not depend on polarisation but exceptions exist, where Zeeman splitting is one example. Both polarised and unpolarised absorption is handled. Even if the gaseous absorption in itself is unpolarised, the expressions to apply must allow that polarisation signals from the surface and the cloud box are correctly propagated to the sensor.

For an introduction to a complete radiative transfer calculations, see Chapter 5. For example, the content of this chapter corresponds roughly to the flowchart displayed in Figure 5.1, outlining a standard radiative transfer emission calculation. In fact, this chapter can be seen as a direct continuation of Chapter 5.

9.1 Overall calculation procedure

The structure handling complete radiative transfer calculations is fixed, where the main workspace method is denoted as yCalc (Fig. 5.1). That is, most ARTS control files include

History	
130220	Revised after parts moved to a new chapter (Patrick Eriksson).
120831	Added flowchart and sections on polarised absorption, iyCalc, auxiliary data and dispersion (Patrick Eriksson).
110611	Extended and general revision (Patrick Eriksson).
050613	First complete version by Patrick Eriksson.

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

allocate memory for the matrix $\mathbf{y}$ (Equation 5.2)
allocate memory for the matrix $\mathbf{i}_b$ (Equation 5.1)
for all sensor positions **do**
  for all pencil beam directions of the block **do** (Section 5.3)
    call iy_main_agenda, giving $\mathbf{i}_y$ (Algorithm 2)
    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_main_agenda.

determine the propagation path by ppath_agenda (Section 9.2)
determine the radiation at the start of the propagation path (Section 9.3)
perform radiative transfer along the propagation path (Section 9.4)
unit conversion of $\mathbf{i}_y$ following iy_unit (Section 9.5)

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 5.3. This vector is only affected by atmospheric radiative transfer. As workspace variable denoted as iyb, but can be considered as a pure internal variable and should not be of concern for the user.

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

$\mathbf{H}_b$: 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.

Atmospheric radiative transfer is solved for each pencil beam direction (line-of-sight) separately. It is the task of iy_main_agenda (Algorithm 2) to perform a single clear sky radiative transfer calculation. This agenda, in its turn, makes us of other agendas, such as ppath_agenda. All methods developed for iy_main_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_main_agenda. The three following sections describes the main calculation steps of iy_main_agenda, in the order they are executed.

9.2 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 10.5. 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 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 9.9 and 10.1.

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 starts. Paths are tracked backwards until the top of the atmosphere or to 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 11.3). Example on propagation paths are shown in Figures 9.1 and 9.2.

Not all propagation paths are allowed for 2D and 3D. In short, 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.

Controlled by ppath_step_agenda, propagation paths can be calculated purely geometrically or considering refraction. When considering refraction, the refractive index is determined at each point along the path according to refr_index_air_agenda. Details about different methods applicable within refr_index_air_agenda are given in Chapter 7.

If nothing else is stated, it assumed that all frequency components share a single propagation path. Another way to express this assumption is that dispersion is neglected. See Section 9.7 for how to consider dispersion. In the non-dispersive case, the propagation path is valid for average of the first and last element in f_grid, as this is the frequency given to refr_index_air_agenda.

Propagation paths can be calculated separately by the method ppathCalc, but for standard calculations the propagation paths are calculated internally by yCalc. Methods and variables to control the path calculations are discussed in Section 10.1.

9.3 The radiative background

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

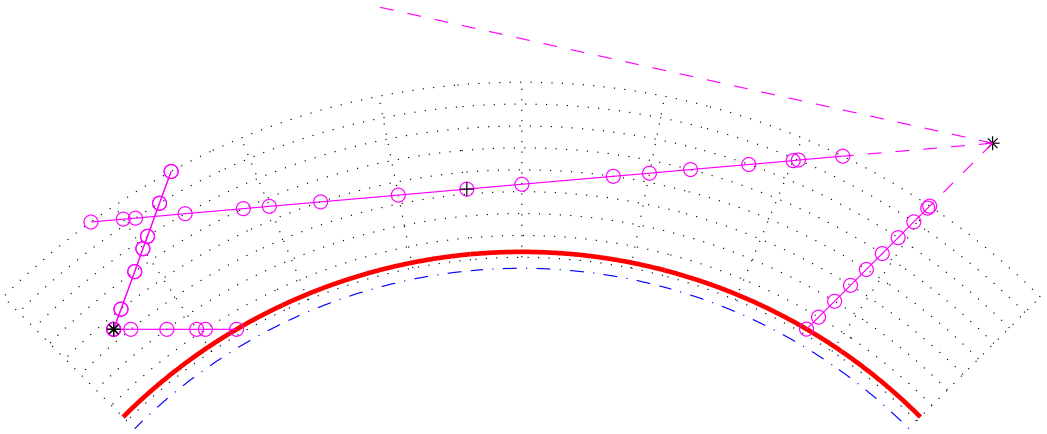


Figure 9.1: Examples on allowed propagation paths for a 2D atmosphere. The atmosphere is plotted as in Figure 3.2 beside that the points for the atmospheric fields are not emphasised. The position of the sensor is indicated by an asterisk ($*$), the points defining the paths are plotted as circles ($\circ$), 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 ($\oplus$); but note that these no longer are explicitly included as path point (in contrast to ARTS-2.0 and earlier). 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.

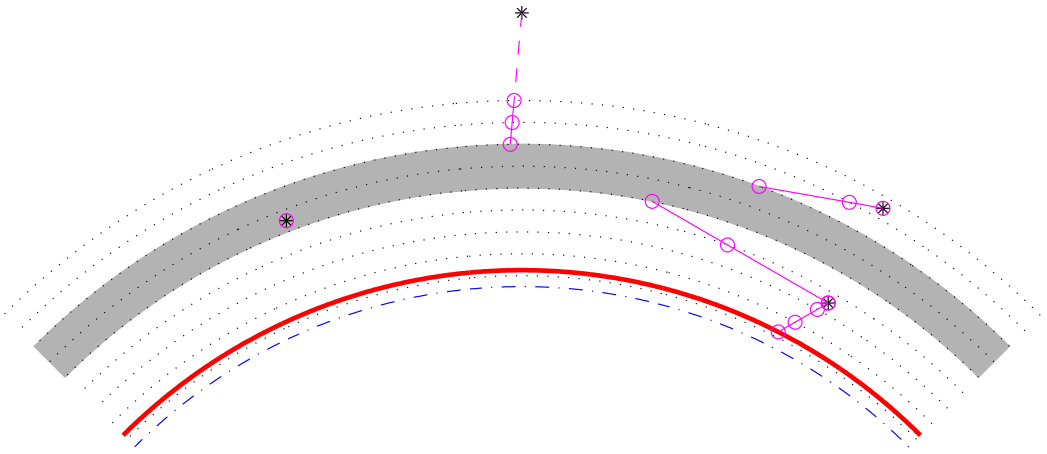


Figure 9.2: Examples on allowed propagation paths for a 1D atmosphere with an activated cloud box. Plotting symbols as in Figure 9.1. When the sensor is placed inside the cloud box, the path is defined with a single point, to know for which position and line-of-sight the intensity field of the cloud box shall be interpolated.

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. It is the task of iy_surface_agenda to return this up-welling radiation from the surface, see further Chapter 11.

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 could be determined by a new call of iy_main_agenda and the radiative background for that calculation is then space or the cloud box. The intensity field entering the cloud box is in some cases calculated by calls of iy_main_agenda (with cloud box deactivated) and the radiative background for these calculations 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.

9.4 Basic radiative transfer variables and expressions

This section describes how the core radiative transfer equation is solved practically in ARTS. As mentioned, in this chapter focus is put on emission measurements. The equation to solve is Equation 4.16:

$$\frac{\mathrm{d}\mathbf{s}}{\mathrm{d}l} = \mathbf{A}_a \left[\mathbf{b} - \mathbf{s}\right] = -\mathbf{A}_a\mathbf{s} + B\mathbf{a}_a,$$

where the involved quantities are defined and discussed in Section 4.2.

9.4.1 Unpolarised absorption

Let's start with the simpler case of non-polarised absorption (that is, the absorption is independent of polarisation state). For unpolarised absorption the matrix $\mathbf{A}_a$ is diagonal, with all diagonal elements equal, and only the first of the elements of $\mathbf{a}_a$) 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 6.62)

$$I_{i+1} = I_i e^{-\tau_i} + \left(\bar{B}_i + \bar{j_{n,i}}/\bar{\alpha}\right)\left(1 - e^{-\tau_i}\right), \tag{9.1}$$

with

$$\bar{B}_i = (B(T_i) + B(T_{i+1}))/2, \tag{9.2}$$
$$\bar{\alpha} = (\alpha_i + \alpha_{i+1})/2 \tag{9.3}$$
$$\bar{j_{n,i}} = (j_{n,i} + j_{n,i+1})/2 \tag{9.4}$$
$$\tau_i = \Delta l_i \bar{\alpha}, \tag{9.5}$$

where I_i, T_i and α_i are the radiance, temperature and absorption coefficient, respectively, at point i of the propagation path, and Δl_i is the distance along the path between point i and $i+1$. That is, $\bar{B}_i$ is an average of the Planck function at the path step end points, and the absorption is assumed to vary linearly between the two points. The start value of I is governed by the radiative background (Section 9.3).

A consequence of unpolarised absorption is that also the emission is unpolarised, and the emission term vanishes for higher Stokes elements. Accordingly, the expression for the second Stokes component is

$$Q_{i+1}(\nu) = Q_i(\nu) e^{-\tau_i}. \tag{9.6}$$

The third and forth Stokes component are handled likewise. The expressions above are implemented in the workspace method iyEmissionStandard, intended to be part of iy_main_agenda. The non-LTE term is as in Equation 4.11.

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

$$I = \sum_i t_{i+1} \left(\bar{B}_i + \bar{j_{n,i}}/\bar{\alpha}\right) (1 - e^{-\tau_i}), \tag{9.7}$$

where I is the final intensity and 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 (I_i must be known, Section 16). However, the calculation of weighting functions is simplified if $\mathbf{T}_i$ is at hand, and this quantity is also tracked by iyEmissionStandard.

9.4.2 Polarised absorption

The overall calculation procedure is the same with polarised absorption, the only difference is the radiative transfer expression applied. The calculations for the different Stokes components can here not be separated, and matrix-vector notation is required:

$$\mathbf{s}_{i+1} = e^{-\Delta l_i \bar{\mathbf{K}}_i} \mathbf{s}_i + (\mathbf{1} - e^{-\Delta l_i \bar{\mathbf{K}}_i}) \left(\bar{\mathbf{b}}_i + \bar{\mathbf{K}}_i^{-1} \bar{\mathbf{j}_{n,i}}\right), \tag{9.8}$$

where $\mathbf{1}$ is the identity matrix. The $\mathbf{K}$ and $\mathbf{b}$ at point i and $i+1$ are averaged (element-wise) to give $\bar{\mathbf{K}}$ and $\bar{\mathbf{b}}$, respectively, in line with Equation 9.3. The calculation of the transmission matrix,

$$\mathbf{T}_i = e^{-\Delta l_i \bar{\mathbf{K}}_i}, \tag{9.9}$$

involves a matrix exponential. This calculation step is handled for simpler cases with analytical expressions, while for more complex cases the Padé approximation (*ARTS Developer Guide*, Section 7.3) is applied. Only the first element of $\bar{\mathbf{b}}$ is non-zero, and only the first column of the matrix corresponding to the term $(\mathbf{1} - e^{-\Delta l_i \bar{\mathbf{K}}_i})$ is of interest.

9.4.3 Blackbody and cosmic background radiation

As mentioned, the term B is the Planck function. In ARTS the following version of the Planck function is used:

$$B(T) = \frac{2h\nu^3}{c^2(exp(h\nu/k_bT) - 1)} \tag{9.10}$$

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.) The expression in Equation 9.10 deviates from the exact definition (see Eq. 6.16 in *ARTS Theory*) as it includes c instead of the local propagation speed (v). The reason for this is the n^2-law of radiance, discussed in the section below.

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).

9.5 Output unit and the n^2-law

First of all, it should be noticed that ARTS does not enforce any 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. 9.10). Conversion to other units is selected by the iy_unit workspace variable. The standard manner is to apply the unit conversion as part of the calculations performed inside yCalc. See the built-in documentation of the workspace method you have selected for iy_main_agenda for comments on practical aspects and available output units. The most extensive support for conversion to other units is provided by iyEmissionStandard, while other methods have no support at all (ie. they ignore iy_unit). It is also possible to change the unit as a post-processing step by yApplyUnit (or iyApplyUnit), but some restrictions apply and there are no automatic checks if the input data have correct unit. Further considerations and expressions for the unit conversion are discussed in the ARTS-2 journal paper [*Eriksson et al.*, 2011, Sec. 5.7].

The n^2-law of radiance is introduced in Section 6.5 of *ARTS Theory*. As shown in that section, the main impact of the law is handled by consistently using the vacuum speed in the definition of the Planck radiation law, as done inside ARTS (Eq. 9.10). This suffices if the sensor is placed in space (where the refractive index is 1), or if you use brightness temperatures. Remaining cases are also handled exactly if iyEmissionStandard is used. For those remaining cases, radiance data shall be scaled with the refractive index squared at the observation position. For Earth, the maximum value of this factor is about 0.1 %, and can anyhow normally be neglected.

In summary, there is normally no need for you as an user to consider the n^2-law. The exception is if you extract radiance data for a point inside an atmosphere, and the refractive index deviates significantly from 1 at this point.

9.6 Single pencil beam calculations

The text above assumes that yCalc is used. This method can always be used, but yCalc is not mandatory if the simulations only deal with monochromatic data for a single line-of-sight. In this case, it could be more handy to use iyCalc, which basically is a direct call of iy_main_agenda. A reason for selecting iyCalc is that a larger set of auxiliary quantities can be extracted (Sec. 9.8).

On the input side, the main difference when using iyCalc is that the observation position and line-of-sight are specified by rte_pos and rte_los (instead of sensor_pos and sensor_los). The calculated radiances are returned as the matrix iy (instead of the vector y). No automatic unit conversion is made inside iyCalc. This is instead handled separately by iyApplyUnit.

9.7 Dispersion

The clear-sky radiative transfer methods handle all frequencies in f_grid in parallel, for efficiency reasons. One consequence of this feature is that only a single propagation path is calculated, that is assumed to be common for all frequencies. With other words, dispersion is not considered. This is in general an acceptable simplification, but exceptions exist where one example is radiative transfer through the ionosphere at frequencies approaching the "plasma frequency".

When dispersion is expected to give a significant impact on the results, ARTS offers a general solution. Dispersion can be handled by setting iy_main_agenda as:

```
AgendaSet( iy_main_agenda ){
  iyLoopFrequencies
}
```

The radiative transfer method you put in iy_main_agenda for non-dispersive calculations are now moved to iy_loop_freqs_agenda. For example, if iyEmissionStandard is the method of your choice:

```
AgendaSet( iy_loop_freqs_agenda ){
  iyEmissionStandard
}
```

The approach is simple, iyLoopFrequencies calls iy_loop_freqs_agenda for each single frequency in f_grid and appends the output. With some details, iyLoopFrequencies performs a loop over the f_grid, creates an internal f_grid of length 1 holding the frequency of concern and calls iy_loop_freqs_agenda with this length-1 frequency grid. This has the result that a propagation path is calculated for each frequency component.

Some more steps are required to correctly include dispersion. A basic demand is that ppath_agenda considers refraction. Further, refr_index_air_agenda must provide a dispersive refractive index. Most methods aimed for refr_index_air_agenda give a refractive index that does not varies with frequency. An example on the opposite is refr_index_airFreeElectrons. If a method with dispersive refractive index is used for non-dispersive calculations, it receives the mean of the first and last element in f_grid (as already commented above).

A limitation of iyLoopFrequencies is that it can not be combined with auxiliary data of along-the-path character (Sec. 9.8)

9.8 Auxiliary data

The core output of the radiative calculations is y (iy if iyCalc is used, jacobians discussed in Sec. 16), but different auxiliary data can be extracted. First of all, yCalc outputs automatically y_f, y_pol, y_pos and y_los. These data give information about the frequency, polarisation, sensor position and sensor bore-sight, respectively, corresponding to each value in y. The content of the variables are governed by the sensor settings and the order calculated radiances are stored (discussed in Sec. 5.3).

A more general mechanism for extracting auxiliary data is controlled by the iy_aux_vars workspace variable. This mechanism is most useful together with iyCalc, and for the moment we assume that this method is used (limitations for yCalc are discussed below). The quantities that can be extracted differ, see the built-in documentation for the options for each workspace method of concern, e.g.:

```
arts -d iyEmissionStandard
```

The options for this particular method (iyEmissionStandard) can be divided into different groups (more variables will/can be added):

Atmosphere, along-the-path The pressure, temperature and volume mixing rations along the propagation path.

Attenuation, along-the-path Total and species specific absorption coefficients along the propagation path.

Radiative properties, along-the-path The radiance at each propagation path point.

Overall radiative properties The total (clear-sky) optical depth along the path and flag giving the radiative background.

"Along-the-path" means that data are provided for each point of the propagation path. The path is described by ppath, that is also returned by iy_main_agenda. The ppath variable contains the information needed to geo-position, for example, "along-the path temperatures".

Example on setting of iy_aux_vars (again valid for iyEmissionStandard):

```
ArrayOfStringSet( iy_aux_vars,
    [ "Temperature",
      "VMR, species 0",
      "Absorption, summed",
      "Absorption, species 0",
      "Absorption, species 2",
      "iy",
      "Optical depth" ] )
```

The data are outputted in a single variable, iy_aux. This variable is an array of Tensor4. All dimensions are used when storing e.g. the propagation matrix along the path (for all frequencies of f_grid). For other types of quantities, one or several dimensions are set to have length 1. See the built-in documentation for further details, such as the order of the data dimensions.

Storage of quantities of "along-the-path" type assumes that there exists a common propagation path. This is necessarily not the case for calculations by yCalc. This is the case as a calculation considering an antenna response includes radiative transfer along several propagation paths. The points of these paths do not end up on common altitude grid, neither

are at a fixed distance from the sensor. In fact, the number of points of the paths will likely differ. For this reason, yCalc will issue an error if you in iy_aux_vars include a quantity of "along-the-path" character.

The same applies to dispersion calculations (here the propagation path differs already between the frequencies), and also iyLoopFrequencies gives also an error if "along-the-path" auxiliary data are selected.

9.9 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 require 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 piece-wise (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 9.1 is used, the underlying 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. See also Section 10.3.

Chapter 10

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 9.2 and this section provides further details. For a general usage of ARTS, it should suffice to read Section 10.1. 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. The actual equations applied are found in Chapter 7 of *ARTS Theory*.

10.1 Practical usage

The overall calculation approach for finding the propagation path is specified by ppath_agenda. The standard choice for this agenda is ppathStepByStep, applying ppath_step_agenda repeatedly in order to trace the path backwards, starting at the sensor. This set-up is assumed throughout this chapter. A slighltly different selection of workspace methods is required for radio link calculations, see further Section **??**.

The exact ray tracing algorithm to be applied for the calculation of propagation path is selected through ppath_step_agenda (see further Section 9.2). 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 9.9. This variable sets the maximum distance between points of the 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_stepRefractionBasic. 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. At each ray tracing step, the refractive index is evaluated according to the specification of refr_index_air_agenda. Several methods to determine refractive index are available (see Chapter 7).

History

120202 Revised and parts moved to *ARTS Theory* (Patrick Eriksson).

030310 First complete version written by Patrick Eriksson.

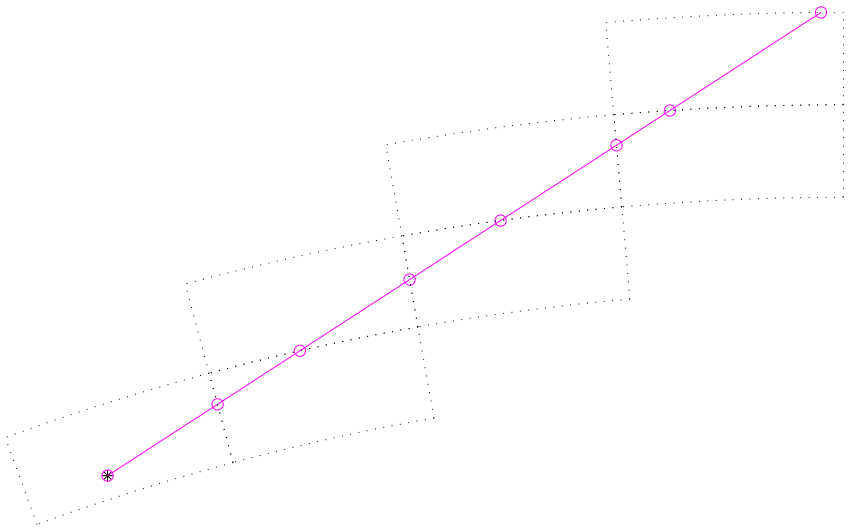


Figure 10.1: Tracking of propagation paths. For legend, see Figure 10.2. The figure tries to visualize how the calculations of propagation paths are performed from one grid cell to next. In this example, the calculations start directly at the sensor position $(*)$ as it placed inside the model atmosphere. 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.

10.2 Calculation approach

The propagation paths are calculated in steps, as outlined in Section 9.2. 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 9.2 of *ARTS Theory*.

A full propagation path is stored in the workspace variable ppath, that is of the type Ppath (see Section 10.5). 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 10.1). 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. 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.

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

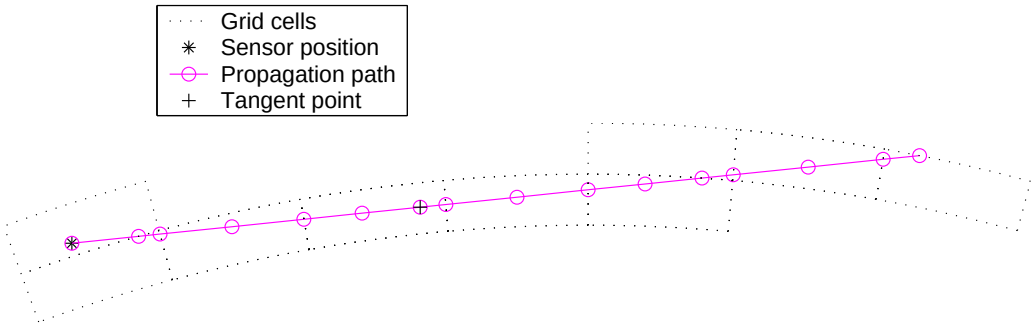


Figure 10.2: As Figure 10.1, but with a length criterion for the distance between the points defining the path. Note: Tangent points are no longer included automatically.

of the atmosphere (determined by geometrical calculations starting at the sensor).

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 cloud box or below the surface level, and check if the last point of the path has been reached.

10.3 Spacing of additional path points

The strategy when considering ppath_lmax differs somewhat between the workspace methods. For pure geometrical calculations, the points are spaced evenly inside the grid box. That is, the points are separated with the same distance ($\leq$ ppath_lmax).

When refraction is considered, the ray tracing moves forward in steps following ppath_lraytrace. When another step of this size would result in a distance $>$ ppath_lmax, the present point is added to $\leq$ ppath_step. A consequence of this is that additional points are likely not evenly spaced. The distance between most points will be ppath_lraytrace times an integer value, but the distance between the last additional point and the grid border can be any number ($\leq$ ppath_lmax). If ppath_step is set to be negative, no additional points are included, as for geometrical calculations.

As points are always included in the propagation paths at the crossings of the atmospheric grids, making these grids finer 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 step length (ppath_lmax) shall be applied for this purpose.

10.4 Tangent points

The term "tangent point" refers to the point of a limb sounding path with the lowest altitude. For 1D cases this definition is clear, but for 2D and 3D calculations there are complications. First of all, there are two different possible definitions: the point having the lowest radius (ie. distance to the planets centre) or the point with the lowest altitude (ie. vertical distance to the reference ellipsoid). The later is the more important with respect to optical thickness

of the path, but the point of the highest pressure would be an even more relevant definition in this context. Another complication is that with refraction there can in principle exist more than one tangent point.

Up to ARTS-2.0 minimum-radius tangent points were added as extra points to the propagation paths (a reminiscent from ARTS-1), but this feature has now been removed following the discussion in the paragraph above. However, many internal functions make use of the concept of tangent point (as the minimum-radius one). The altitude-based tangent point for a propagation path can be determined with the method TangentPointExtract.

10.5 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 as follows:

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 through the atmosphere. Allowed values are ≥ 1. 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` ≤ 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. It is defined as the product: $rn\sin(\psi)$ (see Chapter 7.4.1 of *ARTS Theory*), at the position of the sensor. This is a true constant for the path just for 1D atmospheres, but can be useful also in other cases. For example, it equals the impact parameter normally used to define limb radio occultations. This field is initiated -1, to indicate that the constant is not yet set.

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

start_pos [Vector] The practical start position of the propagation path. This vector equals in general the last row of `pos`. The exception is radio link calculations where the transmitter is placed above the model atmosphere, where this field gives the position of the transmitter.

start_los [Vector] Line-of-sight at start point of propagation path. Set and used in the same way as `start_pos`.

start_lstep [Numeric] The distance between `start_pos` and the last position in `pos`. This value is zero, except for a transmitter placed above the top-of-the-atmosphere. Hence, this length corresponds to propgation if free space (n=1).

end_pos [Vector] The end position of the propagation path. If the point is placed inside the atmosphere, this field is redundant as it is equal to the first row of `pos`, but identifies the sensor position for observations from space.

end_los [Vector] The line-of-sight at the end point of the propagation path. Provides additional information if the sensor is placed above the top-of-the-atmosphere, and gives then the observation direction of the sensor.

end_lstep [Numeric] The distance between `end_pos` and the first position in `pos`. This value is non-zero just if the sensor is placed above the top-of-the-atmosphere. Hence, this length corresponds to propgation if free space (n=1).

pos [Matrix] The position of the propagation path points inside the atmosphere. 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 5.2.1). 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 3.2). 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 `start_pos` if it is inside the atmosphere).

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 5.2.2. 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.

r [Vector] The radius for each path position. The length of this vector is accordingly `np`. This is a help variable for plotting and similar purposes.

lstep [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.

nreal [Vector] The real part of the refractive index at each path position. This index corresponds to the phase velocity.

ngroup [Vector] The group index of refraction. This index corresponds to the group velocity.

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.

10.6 Further reading

The implementation, calculation approaches and the numerical expressions used are discussed in Chapter 7 of *ARTS Theory*.

Chapter 11

Reference ellipsoid and surface properties

11.1 The reference ellipsoid

The ellipsoid is an imaginary surface used as a reference when specifying the surface altitude and the altitude of pressure levels. As the name indicates, this reference surface is defined as an ellipsoid. The reference ellipsoid should normally be set to some global geodetic datum, such as WGS-84. Inside ARTS, the ellipsoid is represented as a vector denoted as refellipsoid. This vector must have length two, where the two elements are equatorial radius (r_e) and eccentricity (e), respectively.

11.1.1 Ellipsoid models

A geodetic datum 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 11.1. The definition of the ellipsoid used in ARTS is based on r_e and the eccentricity, e:

$$e = \sqrt{1 - r_e^2/r_p^2}. \tag{11.1}$$

Workspace methods to set the reference ellipsoid for a particular planet include refellipsoidEarth, refellipsoidMars and refellipsoidMars.

The radius of the ellipsoid for a given (geocentric) latitude, α, is

$$r_\odot(\alpha) = \sqrt{\frac{r_e^2 r_p^2}{r_e^2 \sin^2\alpha + r_p^2 \cos^2\alpha}} = \frac{r_p}{\sqrt{\sin^2\alpha + (1-e^2)\cos^2\alpha}} \tag{11.2}$$

The radius given by Equation 11.2 can be directly applied for 3D atmospheres. For 2D cases, the ellipsoid data must be adopted. The assumption inside ARTS is that the 2D plane goes through the North and South poles. The polar radius to apply should then match

History

120224 Extended and revised (Patrick Eriksson).

050613 First version finished by Patrick Eriksson.

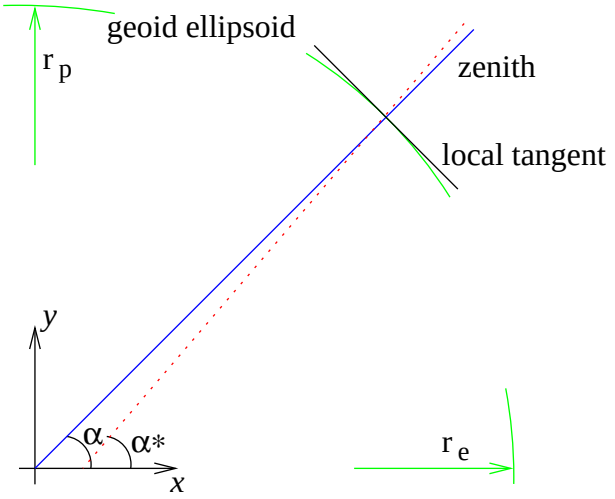


Figure 11.1: Definition of the ellipsoid radii, r_e and r_p, geocentric latitude, α, and geodetic latitude, α^*. The dotted line is the normal to the local tangent of the ellipsoid. The zenith and nadir directions, and geometrical altitudes, are here defined to follow the solid line.

the real ellipsoid radius at the highest latitude inside the satellite orbit plane. The method refellipsoidOrbitPlane performs this operation.

Further, for 1D cases the reference ellipsoid 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 with refellipsoidForAzimuth, that sets r_e to the local radius of curvature of the ellipsoid and e to zero. The curvature radius differs from the local radius except at the equator and an east-west direction. For example, at 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 11.2). The curvature radius, r_c, of an ellipsoid is [*Rodgers*, 2000]

$$r_c = \frac{1}{r_{ns}^{-1}\cos^2\alpha + r_{ew}^{-1}\sin^2\alpha} \tag{11.3}$$

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

$$r_{ns} = r_e^2 r_p^2 (r_e^2\cos^2\omega + r_p^2\sin^2\omega)^{-\frac{3}{2}} \tag{11.4}$$

$$r_{ew} = r_e^2 (r_e^2\cos^2\omega + r_p^2\sin^2\omega)^{-\frac{1}{2}} \tag{11.5}$$

The azimuth angle, ω, is defined in Section 5.2.2. The latitude and azimuth angle to apply in Equations 11.3–11.5 shall rather be valid for a middle point of the propagation paths (such as some tangent point), instead of the sensor position.

11.1.2 Geocentric and geodetic latitudes

The fact that Earth 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 11.1. It could be mentioned that a geocentric latitude does not depend on the ellipsoid used, while the geodetic latitudes change if another reference ellipsoid is selected. For Earth, the largest difference between geocentric and geodetic latitude is found at mid-latitudes, where it reaches 12 arc-minutes. There are yet no methods in ARTS for conversion of data between geodetic and geocentric latitudes.

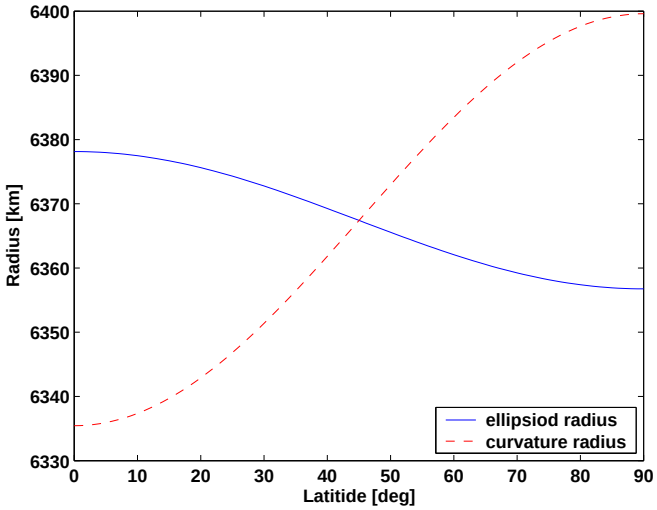


Figure 11.2: The ellipsoid radius ($r_\odot$) and curvature radius (r_c) for the WGS-84 reference ellipsoid. The curvature radii are valid for the north-south direction.

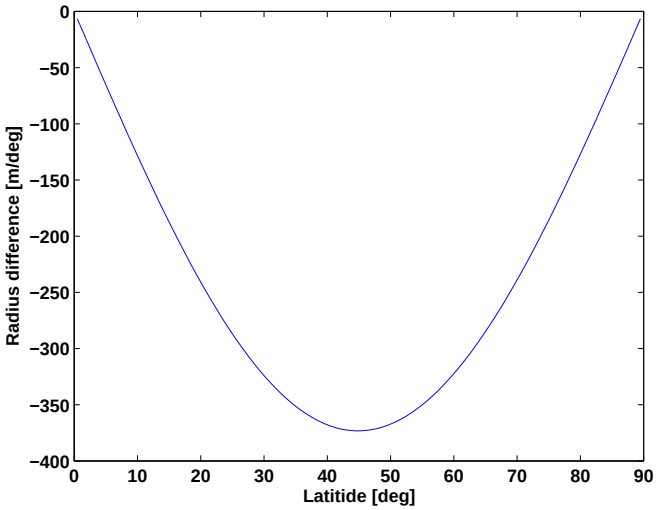


Figure 11.3: The change of the WGS-84 ellipsoid radius for 1° latitude differences.

11.2 Surface altitude

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

$$r_s = r_\odot + z_s \tag{11.6}$$

As also mentioned in Section 3.6, 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. For 1D, the surface constitutes a sphere by definition (as the ellipsoid), while for 2D and 3D any shape is allowed and a rough model of the surface topography can be made.

11.3 Surface radiative properties

If there is an interception of the propagation path by the surface, emission and scattering by the surface must be considered. This is the task of iy_surface_agenda. The standard method for this agenda is iySurfaceRtpropAgenda (in fact, the only option implemented so far) and the rest of this section outlines how this workspace method works. The upwelling radiation from the surface can be written as (Figure 11.4)

$$I_a^u = I_e + \sum_l \mathbf{R}_l I_l^d \tag{11.7}$$

where I indicates the Stokes vector for one frequency, I_a^u is the total upward travelling intensity from the surface along the propagation path, I_e is the emission from the surface, 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 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.

Some special cases are discussed below. Section 6.9 of *ARTS Theory* provides the theoretical background.

11.3.1 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.

11.3.2 Specular reflections

Several methods to incorporate a flat surface exist, including surfaceFlatRefractiveIndex and surfaceFlatScalarReflectivity. 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` Equations 6.79-6.82 in *ARTS Theory* give the values of surface_rmatrix and surface_emission.

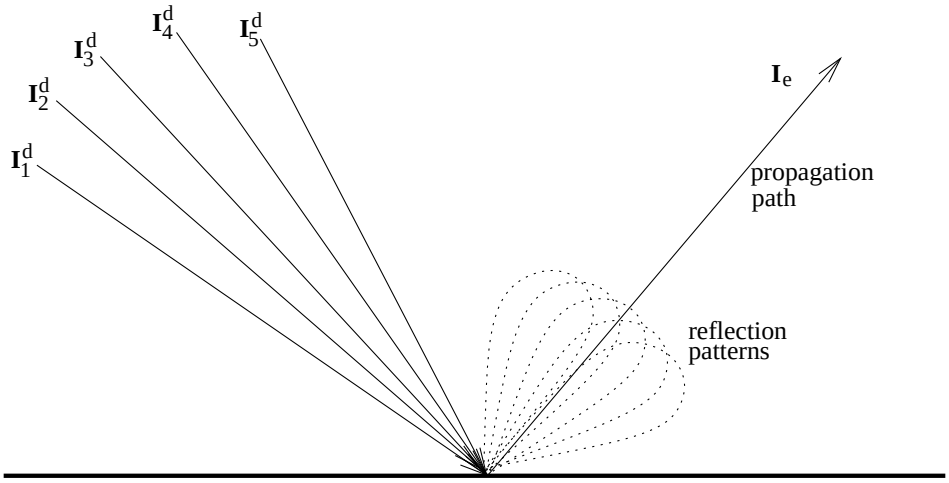


Figure 11.4: Schematic of Equation 11.7.

11.3.3 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.

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:

$$\mathbf{R} = \begin{bmatrix} w & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}. \tag{11.8}$$

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 6.83 of *ARTS Theory*)

$$w = \int_{\theta_a}^{\theta_b} \int_{\phi_a}^{\phi_b} \cos(\theta) f(\theta, \phi, \theta_1, \phi_1) \sin(\theta) \, \mathrm{d}\phi \, \mathrm{d}\theta, \tag{11.9}$$

that gives

$$w = \frac{r_d}{2} \left[\cos(2\theta_a) - \cos(2\theta_b) \right], \tag{11.10}$$

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

The emission (surface_emission) becomes:

$$\mathbf{b} = \begin{bmatrix} (1 - r_d) \, B \\ 0 \\ 0 \\ 0 \end{bmatrix}. \tag{11.11}$$

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 5.2. 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 5.2. 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 mixer, if the antenna response differs between the two bands. If the same antenna response

History
110826 A simple version by Patrick Eriksson.

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.

12.2 Some comments

Some text removed from other chapters, to be intgrated into this chapter:

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_dlos. For example, pencil beam zenith angle i is calculated as

$$\psi_i = \psi_0 + \Delta\psi_i \tag{12.1}$$

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

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_dlos will not be centred around zero which is the case when the "true" line-of-sight is used.

Chapter 13

Doppler effects and winds

The default assumption in ARTS can be expressed as the atmosphere is assumed to be static and the observation platform is static during each measurement, while any relative movement between the atmosphere and the sensor will cause Doppler effects. ARTS handles three sources to Doppler shifts: winds, rotation of the planet and sensor velocity.

13.1 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 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 shorthand for saying that the wind is zero throughout the atmosphere (Sec. **??**). Otherwise, the size of the variable is required to match the atmospheric grids.

No further input is required, a Doppler shift is added as soon as any of the winds is non-zero (exceptions discussed in Sec. 13.4). For clarity, even though a setting of wind_u_field always is demanded, this wind component has no effect on Doppler shifts for 1D and 2D calculations, as the wind moves at an angle of 90° from the observation plane (Sec. **??**).

13.2 Planet rotation

ARTS applies an Earth-Centred, Earth-Fixed, (ECEF) coordinate system. This implies that a ground-based receiver follows the planet's rotation. On the other hand, if e.g. the sensor is placed on another planet, or is in a transit orbit, the rotation of the observed planet will cause Doppler effects. This is treated in ARTS by a short-cut, the rotational movement can be translated to an imaginary wind by the method wind_u_fieldIncludePlanetRotation.

History	
130223	Restructured to include effects beside winds (PE).
121218	First version by Patrick Eriksson.

This pseudo-wind, v'_u, is calculated as

$$v'_u = \frac{2\pi \cos(\alpha)(r+z)}{t_p}, \tag{13.1}$$

where α is the latitude, r is the local planet radius, z is the altitude and t_p is the planet's rotational period. This term is added to the true zonal wind speed, v_u.

13.3 Sensor velocity

The feature is not yet fully implemented, but a rudimentary inclusion of the sensor's velocity can be made by the rte_alonglos_v workspace variable. The variable shall be set to the sensor velocity component along the viewing direction. For the moment, this velocity is assumed to be the same for all pencil beam calculations.

13.4 Limitations

For efficiency reasons, when extracting particle extinction (absorption and scattering) the mean Doppler shift along the propagation path is applied (but the shift varies with frequency). This allows to at least include larger shifts caused by e.g. satellite velocity. The smaller shifts (due to winds) are here of less importance as the particle extinction is quite smooth with frequency.

The above is valid for transmission type calculations. Doppler shifts are so far totally neglected inside the DOIT and MC scattering modules.

13.5 Equations

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

$$v = \sqrt{v_u^2 + v_v^2 + v_w^2}. \tag{13.2}$$

The zenith angle of the wind direction is

$$\psi_v = \arccos(v_w/v), \tag{13.3}$$

and the azimuth angle is

$$\omega_v = \arctan(v_u/v_v). \qquad \text{(implemented by the atan2 function)} \tag{13.4}$$

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

$$\cos\gamma = \cos\psi_v \cos\psi_l + \sin\psi_v \sin\psi_l \cos(\omega_v - \omega_l), \tag{13.5}$$

where ψ_l and ω_l are the angles of the line-of-sight.

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

$$\Delta\nu = \frac{-v\nu_0 \cos(\gamma)}{c}, \tag{13.6}$$

where ν_0 is the rest frequency and c is the speed of light. The core part of these calculations is implemented in the general internal function `dotprod_with_los`. The negative sign is caused by the fact the "line-of-sight" is the observation direction, not the direction of the EM waves.

Chapter 14

Faraday rotation

The polarisation state of an electromagnetic wave propagating through a plasma with a static magnetic field will be changed, normally denoted as Faraday rotation. The effect is present for both passive and active signals, but Faraday rotation is proportional to ν^{-2} and can in general be neglected for emission measurements due to the relatively high frequencies applied. For Earth, the effect can in general be neglected above $\sim$5 GHz. Hence, Faraday rotation is a special consideration for radio/microwave radiative transfer. A brief theoretical description is found below in Section 14.2.

14.1 Practical usage

The first step is to ensure that the magnetic field and field of free electrons are non-zero. See Section **??** for how to introduce a magnetic field. Free electrons are treated as an "absorbing species", and hence are part of vmr_field (Sec. 4.2.2). A further constrain is that the Stokes dimensionality (stokes_dim) is set to 3 or 4. This can be understood by Equation 14.4, Faraday rotation affects Stokes elements 2 and 3.

Faraday rotation is treated by propmat_clearsky_agenda. For the moment, there exists a single workspace method for the task and that is propmat_clearskyAddFaraday. That is, this method must be included in propmat_clearsky_agenda if you want to include Faraday rotation in your calculations.

14.2 Theory

A wave propagating through the ionosphere will force free electrons to move in curved paths. If the incident wave is circularly polarised, the motion of the electrons will be circular. The refractive index will then not be a single constant, but depending on polarisation (i.e. anisotropic). More precisely, left and right hand polarised waves will propagate with different speeds. Moreover, as a plane polarised wave can be thought of as a linear superposition of a left and a right hand polarised wave with equal amplitudes, but different phase, the plane of polarisation will then rotate as the wave is propagating through the media. This is denoted as Faraday rotation.

History

121217 Written (PE), partly based on text written originally by Bengt Rydberg.

Birefringance is an associated mechanism, but it is not yet treated by ARTS. This later effect originates also on the fact that right- and left-hand circular polarisation have different refractive index. This can result in that the two polarisations obtain different propagation paths. For frequencies close to the "plasma frequency" the birefringance can be a strong effect, but for higher frequencies it should be secondary to Faraday rotation. Expressed roughly, a difference in optical path for the two circular polarisation of a quarter of a wavelength changes the polarisation state strongly by Faraday rotation, while additional effects coming from a difference in propagation path (birefringance) should be negligible.

According to *Rybicki and Lightman* [1979], using Gaussian (cgs) units, the angle of rotation (ϑ_F) of a plane polarised wave can be described as

$$\vartheta_F = \frac{e^3}{2\pi c^2 m^2 \nu^2} \int_0^d n_e(s)\mathbf{B_{geo}}(s) \cdot \mathrm{d}\mathbf{s},$$

which converted to SI units becomes [*Kraus*, 1966][1]

$$\vartheta_F = \frac{e^3}{8\pi^2 c\epsilon_0 m^2 \nu^2} \int_0^d n_e(s)\mathbf{B_{geo}}(s) \cdot \mathrm{d}\mathbf{s} \approx \frac{23648}{\nu^2} \int_0^d n_e(s)\mathbf{B_{geo}}(s) \cdot \mathrm{d}\mathbf{s}, \quad (14.1)$$

where e is the charge of an electron, ϵ_0 is the permittivity of vaccum, m is the electron mass, $n_e(s)$ is the density of electrons at point s, $\mathbf{B_{geo}}(s)$ is the geomagnetic field at point s, and $\cdot$ denotes the dot (scalar) product. Accordingly, the Faraday rotation is proportional to the part of the magnetic field along the propagation path, the field normal to the path gives no effect.

The change in rotation angle along the propagation path, r is then given by

$$r = \frac{\mathrm{d}\vartheta_F}{\mathrm{d}s} = \frac{e^3}{8\pi^2 c\epsilon_0 m^2 \nu^2} n_e(s)\mathbf{B_{geo}}(s) \cdot \hat{\mathbf{s}}. \quad (14.2)$$

An "magneto-optical" effect (Sec. 4.2.1) of this type is mapped to a propagation matrix as

$$\mathbf{K} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 2r & 0 \\ 0 & -2r & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}. \quad (14.3)$$

If no other effects are present, the effect on the Stokes vector for some part of the propgation path can be expressed as a Mueller rotation matrix [*Goldstein*, 2003; *Meissner and Wentz*, 2006]:

$$\begin{bmatrix} I_F \\ Q_F \\ U_F \\ V_F \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(2\vartheta_F) & -\sin(2\vartheta_F) & 0 \\ 0 & \sin(2\vartheta_F) & \cos(2\vartheta_F) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} I \\ Q \\ U \\ V \end{bmatrix}. \quad (14.4)$$

[1] The two versions of this equation are also found in Wikipedia (under *Faraday effect*), but expressed using λ and an error for the numerical factor (2012-12-17). The value $2.365 \cdot 10^4$, derived from the fundamental constants, is confirmed by *Wright et al.* [2003].

Chapter 15

Transmission calculations

The term "transmission calculations" refers here to situations when the emission from the atmosphere and surface can be neglected. These calculations can be divided into two main types. The first one is when just the transmission of the atmosphere is diagnosed (Sec. 15.1). The observation geometry is then given exactly as for emission simulations, by a position and a line-of-sight.

The second main type is radio link budgets (Sec. **??**). For this case, the propagation path is determined solely by the position of the transmitter and the receiver. That is, the user does not need to set a line-of-sight of the sensor (receiver), it is determined by the transmitter position. A characterisation of a radio link normally involves several attenuation terms not encountered for passive measurements, such as free space loss and defocusing. Faraday rotation (Sec. 14) is an additional physical mechanism that is of special concern for active microwave devices (it can normally be neglected at the higher frequencies used for passive observations).

15.1 Pure transmission calculations

This section discusses the iyTransmissionStandard workspace method, that is relevant if you want to calculate the transmission through the atmosphere for a given position and line-of-sight. The set-up is largely the same as for simulations involving emission, such as that the observation geometry is defined by sensor_pos and sensor_los (or rte_pos and rte_los if iyCalc is used). The first main difference is that iy_transmitter replaces iy_space_agenda and iy_surface_agenda. The second main difference is that handling of cloud scattering is built-in and the need to define iy_cloudbox_agenda vanishes. These differences appear inside iy_main_agenda.

As for emission measurements (Sec. 9.1, Algorithm 2) the first main operation is to determine the propagation path through the atmosphere, but this is here done without considering the cloud-box (it is simple deactivated during this step). The possible "radiative backgrounds" are accordingly the surface or space, i.e. where the propagation path starts.

The next step is to set iy_transmitter. It should be noted that the same variable is use independently if the radiative background is space or the surface. It is up to the user to decide if these two cases shall be distinguished in some manner (no workspace method for this task

History

130206 Written (PE), partly based on text written originally by Bengt Rydberg.

exists yet). For these calculations, the standard choice is to set iy_transmitter with MatrixUnitIntensity. If this workspace method is used, the output of iyTransmissionStandard shows you the fraction of unpolarised radiation that is transmitted through the atmosphere, and the polarisation state of the transmitted part.

The radiative transfer expression applied is (cf. Eq. 9.8)

$$\mathbf{s}_{i+1} = e^{-\bar{\mathbf{K}}s}\mathbf{s}_i \tag{15.1}$$

where the extinction matrix is determined in the same manner as for emission cases (Sec. 9.3). In situations where the matrix $\mathbf{K}$ is diagonal, the scalar form shown in Eq. 9.6 is used.

The method determines automatically if any part of the propagation path is inside the cloud-box (if active). If this is the case, particle extinction is included in $\mathbf{K}$, following the same path step averaging as applied for pure absorbing species. For this method, scattering is purely a loss mechanism, there is no gain by scattering into the line-of-sight

A related concern is the treatment of the surface. In standard usage of the method, there is no signal reflected by the surface, and the radiative transfer calculations are started at the surface.

See the built-in documentation of iyTransmissionStandard for a full list of possible auxiliary quantities. These data include quantities that make it possible to determine the transmission for different parts of the propagation path. For example, the state of iy at each point of the propagation path can be extracted, and also the transmission matrix (Eq. 9.9) for each path step is accessible.

Chapter 16

Clear-sky Jacobians

Inversions of both OEM and Tikhonov type require that the Jacobian can be provided by the forward model [see e.g. *Eriksson*, 2000]. A retrieval characterisation following *Rodgers* [1990, 2000] raises the same demand. A column of the Jacobian, $\mathbf{K_x}$, is defined as

$$\frac{\partial \mathbf{y}}{\partial x_p}, \tag{16.1}$$

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

The quantity in Eq. 16.1 is in the atmospheric sounding community frequently denoted as a "weighting function", and accordingly $\mathbf{K_x}$ is called the weighting function matrix. In the documentation of ARTS both terms (Jacobian and weighting functions) are used. 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.

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 Jacobian is calculated. A practical point is the calculation speed, primarily determined if perturbations or analytical expressions are used (Sec. 16.1). The derivation of the different Jacobians 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 iterations). 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 Jacobians for clear-sky conditions, i.e. to be applied outside the cloudbox. So far none of the scattering methods provide Jacobians. Sections 16.1 - 16.2 contain information of general character, while the available quantities are discussed in the remaining sections (Section 16.4 and forward).

History

110826 First complete version by Patrick Eriksson.

16.1 Introduction

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

16.1.1 Perturbations

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

$$\mathbf{K}_{\mathbf{x}}^{p} = \frac{\mathcal{F}(\mathbf{x} + \Delta x \mathbf{e}^{p}, \mathbf{b}) - \mathcal{F}(\mathbf{x}, \mathbf{b})}{\Delta x} \tag{16.2}$$

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.

16.1.2 Analytical expressions

For most atmospheric variables, such as species abundance, it is possible to derive an analytical expression for the Jacobians. 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 because some underlying functions of ARTS are best-fit methods that are internally interpolated to the correct grid anyways. The analytical calculations are introduced in Sec. 16.3.

16.1.3 Workspace variables and methods

As a workspace variable, the complete Jacobian is denoted as jacobian. Auxiliary information is provided by jacobian_quantities. 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. It does not matter in which order these "add methods" are called.

The definition of retrieval quantities is finalised by calling jacobianClose. To disable the calculation of the Jacobian, 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.

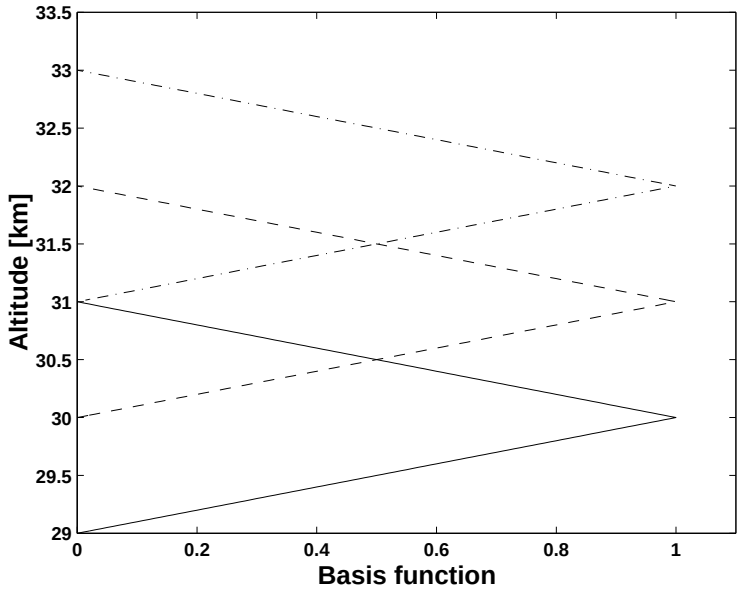


Figure 16.1: Examples on 1d basis functions for a vertical grid with a 1 km spacing: —— 30 km, – – – 31 km and – · – 32 km.

16.2 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 Jacobians require direct consideration of the basis functions. ARTS operates with two types of basis functions.

16.2.1 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 "tent functions". Such functions are exemplified in Fig. 16.1, 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 tent". 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 tent shape inside the grid range, and have a constant value 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 tent functions are then 2D or 3D "tents".

16.2.2 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

$$f(z) = x_0 + x_1(z - b_1) + x_2(z^2 - b_2) + x_3(z^3 - b_3) + x_4(z^3 - b_4) + \dots \quad (16.3)$$

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_{p0}, and the scalars $b_1, b_2, \dots$ are selected, schematically, as

$$0 = \int_{-1}^{1} (z^n - b_n)\, \mathrm{d}z, \quad n > 0. \quad (16.4)$$

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

$$\mathbf{z}_i(j) = z(j)^i - b_i. \quad (16.5)$$

16.3 Atmospheric variables, common expressions

The analytically-oriented calculation procedure to obtain the Jacobian for atmospheric quantities is here outlined. The expressions are based on the chain rule, and can be applied for absorption constituents, atmospheric temperatures and winds. It is important to notice that only local effects are considered, and the expressions have limitations, as discussed below in Sec. 16.3.7.

16.3.1 Matrix derivatives

Matrix exponents are calculated using the Padé approximation with scaling & squaring. Likewise, matrix exponent derivatives are calculated using the same method. The implementation in ARTS follows *Brančík* [2006]. As a brief reminder, following the same notation as *Brančík* [2006], but with altered input/output to match the rest of this text, the transmission matrix is

$$\mathbf{T}_i(x_i) = \exp\left(-\mathbf{K}_i(x_i)r\right) \approx \mathbf{D}_{pq}^{-1}(x_i)\mathbf{N}_{pq}(x_i), \quad (16.6)$$

with

$$\mathbf{N}_{pq}(x_i) = \sum_{j=0}^{p} \frac{(p+q-j)!}{(p+q)!j!\,(p-j)} \left(-\mathbf{K}_i(x_i)r\right)^j , \tag{16.7}$$

and

$$\mathbf{D}_{pq}(x_i) = \sum_{j=0}^{q} \frac{(p+q-j)!}{(p+q)!j!\,(p-j)} \left(\mathbf{K}_i(x_i)r\right)^j . \tag{16.8}$$

This means that the matrix derivation of the transmission of the Padé approximation method is

$$\frac{\partial \mathbf{T}_i}{\partial x_i} \approx \frac{\partial \mathbf{D}_{pq}^{-1}}{\partial x_i}\mathbf{N}_{pq} + \mathbf{D}_{pq}^{-1}\frac{\partial \mathbf{N}_{pq}}{\partial x_i}, \tag{16.9}$$

where the input arguments have been dropped (and will not appear again in this subsection) to make the equation(s) easier to read. To be explicit, the above can be rewritten as

$$\frac{\partial \mathbf{T}_i}{\partial x_i} \approx \mathbf{D}_{pq}^{-1}\left(\frac{\partial \mathbf{N}_{pq}}{\partial x_i} - \frac{\partial \mathbf{D}_{pq}}{\partial x_i}\mathbf{T}_i\right) \tag{16.10}$$

in calculations. Finally, the internal partial derivatives are just

$$\frac{\partial \mathbf{N}_{pq}}{\partial x_i} = \sum_{j=0}^{p} \frac{(p+q-j)!}{(p+q)!j!\,(p-j)} \left(-\frac{\partial \mathbf{K}_i}{\partial x_i}r\right)^j , \tag{16.11}$$

$$\frac{\partial \mathbf{D}_{pq}}{\partial x_i} = \sum_{j=0}^{q} \frac{(p+q-j)!}{(p+q)!j!\,(p-j)} \left(\frac{\partial \mathbf{K}_i}{\partial x_i}r\right)^j . \tag{16.12}$$

Important to note in ARTS is that we average layer properties from properties of the surrounding path points simply by averaging the property at the two path points. That is

$$\mathbf{K}_i = \frac{\mathbf{K}_{i-1/2} + \mathbf{K}_{i+1/2}}{2}, \tag{16.13}$$

where the half denotes the propagation matrix at the two levels. So for every layer, we need the partial derivation of the layer for both the influence by the level above and the level below. Thus,

$$\frac{\partial \mathbf{K}_i}{\partial x_i} = \frac{1}{2}\left(\frac{\partial \mathbf{K}_{i-1/2}}{\partial x_i} + \frac{\partial \mathbf{K}_{i+1/2}}{\partial x_i}\right) \tag{16.14}$$

These calculations are done in parallel to save memory and/or time. In fact, only Equations 16.11 and 16.12 are recalculated for each analytical Jacobian entity that has been requested by the user.

16.3.2 Analytical expression for partial derivation of the propagation matrix

All lines and contributions to the propagation matrix are summed for a total propagation. This means that

$$\mathbf{K}_i = \sum_j \mathbf{K}_j; \quad \frac{\partial \mathbf{K}_i}{\partial x_i} = \sum_j \frac{\partial \mathbf{K}_j}{\partial x_i}, \tag{16.15}$$

for all j lines, continua contributions, collision induced absorption contributions, etc.. This means that to calculate each individual $\partial \mathbf{K}_j/\partial x_i$ is enough to get $\partial \mathbf{K}/\partial x_i$. The method of calculating $\partial \mathbf{K}_j/\partial x_i$ depends on how the absorption is calculated.

Parameterized absorption models

Several methods in ARTS are best-fit or parameterized models. If $\mathbf{K}_j$ is from one of these functions, then a perturbation approach is used in a low-level function to get $\partial\mathbf{K}_j/\partial x_i$. The user sets a small perturbation that is then used internally. This way is used when continua, collision induced absorption and lookup table is used since these are only available in parameterized form in ARTS.

Line-by-line absorption

Expanding $\mathbf{K}_j$ for line-by-line calculations, it can roughly be written

$$\mathbf{K}_j = S_j F_j \mathbf{\Phi}_j, \tag{16.16}$$

where S_j is the scaled line strength, F_j is the shape of the line, and $\mathbf{\Phi}_j$ is the polarization matrix. This rough expression is useful for defining the partial derivatives since we can write

$$\frac{\partial \mathbf{K}_j}{\partial x_i} = \frac{\partial S_j}{\partial x_i} F_j \mathbf{\Phi}_j + S_j \frac{\partial F_j}{\partial x_i} \mathbf{\Phi}_j + S_j F_j \frac{\partial \mathbf{\Phi}_j}{\partial x_i}. \tag{16.17}$$

An important thing to remember about the equation above is that everything in Equation 16.16 is already known (to the level of certainty in the model). Thus, Equation 16.17 can, when it is convenient, be written as

$$\frac{\partial \mathbf{K}_j}{\partial x_i} = \mathbf{K}_j \left(S_j^{-1} \frac{\partial S_j}{\partial x_i} + F_j^{-1} \frac{\partial F_j}{\partial x_i} + \mathbf{\Phi}_j^{-1} \frac{\partial \mathbf{\Phi}_j}{\partial x_i} \right). \tag{16.18}$$

Both Equation 16.17 or 16.18 are used internally in the code base.

To extend the rough expression a little bit, the line strength variable can be thought of as

$$S_j = n_j(T,p) S_{j,0} f(T, \cdots), \tag{16.19}$$

where n_j is the number density, $S_{j,0}$ is the line strength at a reference temperature in LTE, and $f(T,\cdots)$ is a function that mostly depends on the temperature (but also other line parameters and non-LTE effects are included here). The line shape variable can be thought of as

$$F_j = F_n F_m F_u F_s, \tag{16.20}$$

where F_n is a renormalization term far from the line center (e.g., van Vleck and Huber renormalization), F_m is the line mixing, F_u is a unit conversion term, and F_s is the classical definition of line shape (e.g., Lorentz or Voigt distributions). The polarization matrix is the unit matrix except for the Zeeman effect and for Faraday rotation. In scalar simulations, it is simply the 1×1 unit matrix, or a scalar 1.

16.3.3 Separation of terms

The overall task is to calculate (Eq. 16.1 with subscript p left out)

$$\frac{\partial \mathbf{y}}{\partial x}$$

where x is the element of $\mathbf{x}$ for which we want to obtain the weighting function. This is a column of the complete weighting function matrix.

A first step is to identify how sensor characteristics can be incorporated? To make the nomenclature simpler, we assume here that the simulations cover only a single measurement block and we have (cf. Eq. 5.2)

$$\mathbf{y} = \mathbf{H}\mathbf{i}_b. \tag{16.21}$$

We can then apply the chain-rule for a first time to obtain

$$\frac{\partial \mathbf{y}}{\partial x} = \frac{\partial \mathbf{y}}{\partial \mathbf{i}_b}\frac{\partial \mathbf{i}_b}{\partial x} = \mathbf{H}\frac{\partial \mathbf{i}_b}{\partial x} = \mathbf{H}\mathbf{k}_\mathbf{i}. \tag{16.22}$$

Hence, sensor characteristics can be handled by calculating the weighting function column matching all monochromatic pencil beam calculation of the measurement block ($\mathbf{k}_\mathbf{i}$) and perform a multiplication with $\mathbf{H}$, a parallel procedure compared to how $\mathbf{i}_b$ is compiled to obtain $\mathbf{y}$. The calculation procedure expands to allow that the complete weighting function matrix (for quantities covered by the analytical calculation procedure) is calculated as

$$\frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \mathbf{H}\mathbf{K}_\mathbf{i}, \tag{16.23}$$

where $\mathbf{K}_\mathbf{i} = \partial \mathbf{i}_b/\partial \mathbf{x}$. One column of this matrix is $\mathbf{k}_\mathbf{i}$.

The vector $\mathbf{i}_b$ consists of a number of Stokes vectors appended, $\mathbf{i}_b = [\mathbf{s}_1^T, \mathbf{s}_2^T, \ldots, \mathbf{s}_n^T]^T$, and the calculation of $\mathbf{k}_\mathbf{i}$ can schematically be written as

$$\mathbf{k}_\mathbf{i} = \sum_{j=1}^{n} \frac{\partial \mathbf{i}_b}{\partial \mathbf{s}_j}\frac{\partial \mathbf{s}_j}{\partial x} \tag{16.24}$$

The terms $\partial \mathbf{i}/\partial \mathbf{s}_j$ are formally matrices. However, these matrices are not calculated explicitly as they only contain information on where $\mathbf{s}_i$ is stored inside $\mathbf{i}$. That is, these matrices are of bookkeeping character, consisting only of zeros and ones. In practice, results matching $\partial \mathbf{s}_j/\partial x$ are simply inserted in correct place of $\mathbf{k}_\mathbf{i}$, mimicking how is $\mathbf{s}_i$ put into $\mathbf{i}$.

Accordingly, the core task is to calculate $\partial \mathbf{s}_j/\partial x$, where for simplicity the subscript j is dropped below. This calculation is expanded as

$$\frac{\partial \mathbf{s}}{\partial x} = \sum_{i=1}^{n} \frac{\partial \mathbf{s}}{\partial x_i}\frac{\partial x_i}{\partial x}, \tag{16.25}$$

where x_i is the value of the quantity of concern, at each point of the propagation path. That is, i indexes the path points. The actual radiative transfer enters by the terms $\partial \mathbf{s}/\partial x_i$, and they are discussed separately below.

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 Δl_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. Accordingly, the practical calculations are done by first determine all $\partial \mathbf{s}/\partial x_i$, of the given propagation path. These data are then used to determine $\partial \mathbf{s}/\partial \mathbf{x}$, for the retrieval quantity of concern. That is, each $\partial \mathbf{s}/\partial x_i$ is combined with $\partial x_i/\partial x$ for all elements of $\mathbf{x}$. Most of these combinations yield a zero result. The terms $\partial x_i/\partial x$ are determined with help of ARTS' internal interpolation grid position routines.

16.3.4 $\partial \mathbf{s}/\partial x_i$, general case

The term $\partial \mathbf{s}/\partial x_i$ is here outlined for the general case of vector radiative transfer. In this case, the Stokes elements can not be treated separately, and matrix-vector notation is required. The final Stokes vector obtained through Eq. 9.8 can be expressed as

$$\mathbf{s} = \mathbf{s}' + \mathbf{T}' \left[\mathbf{T}_i \mathbf{s}_i + (\mathbf{1} - \mathbf{T}_i) \left(\bar{\mathbf{b}}_i + \bar{\mathbf{K}}_i^{-1} \bar{\mathbf{j}}_{n,i} \right) \right], \tag{16.26}$$

where $\mathbf{s}'$ is the Stokes vector for all emission generated between the sensor and point $i+1$, $\mathbf{T}'$ is the transmission Mueller matrix for the same part of the propagation path and $\mathbf{T}_i$ is defined in Eq. 9.9. The quantities $\mathbf{s}'$, $\mathbf{T}'$ and $\mathbf{s}_i$ are not function of x_i. This gives

$$\begin{aligned} \frac{\partial \mathbf{s}}{\partial x_i} \;=\; & \mathbf{T}' \left[\frac{\partial \mathbf{T}_i}{\partial x_i} \left(\mathbf{s}_i - \bar{\mathbf{b}}_i - \bar{\mathbf{K}}_i^{-1} \bar{\mathbf{j}}_{n,i} \right) + \right. \\ & \left. (\mathbf{1} - \mathbf{T}_i) \left(\frac{\partial \bar{\mathbf{b}}_i}{\partial x_i} + \bar{\mathbf{K}}_i^{-1} \frac{\partial \bar{\mathbf{K}}_i}{\partial x_i} \bar{\mathbf{K}}_i^{-1} \bar{\mathbf{j}}_{n,i} + \bar{\mathbf{K}}_i^{-1} \frac{\partial \bar{\mathbf{j}}_{n,i}}{\partial x_i} \right) \right]. \end{aligned} \tag{16.27}$$

The terms $\partial \mathbf{T}_i/\partial x_i$ and $\partial \bar{\mathbf{b}}_i/\partial x_i$ are both calculated in a pure numerical manner, for different reasons discussed below.

16.3.5 Including the surface

For scattered down-welling radiation the effective transmission matrix is

$$\mathbf{T}' = \mathbf{T}_2 \mathbf{R}_i \mathbf{T}_1, \tag{16.28}$$

where $\mathbf{T}_2$ is the transmission between the surface and the sensor, $\mathbf{R}_i$ is defined in Eq. 11.7 and $\mathbf{T}_1$ is the transmission between the point $i+1$ and the surface.

16.3.6 $\partial \mathbf{s}/\partial x_i$, locally unpolarized absorption

Eq. 16.27 can be simplified for some conditions. A first case is scalar radiative transfer, i.e. only the first element of the Stokes vector is considered and all terms of Eq. 16.27 are scalar quantities. For all other cases, in principle vector radiative transfer must be performed, as $\mathbf{T}'$ can always have off-diagonal elements. Even if atmospheric absorption is totally unpolarized, $\mathbf{T}'$ can be non-diagonal due to the surface (Eq. 16.28). However, if the absorption locally lacks polarization, the calculations can be handled by analytical expressions in a higher degree.

To focus on the analytical expressions dealing with local radiative transfer effects, let us write Eq. 16.26 as

$$\mathbf{s} = \mathbf{s}' + \mathbf{T}' \mathbf{s}_s. \tag{16.29}$$

In the nomenclature of Eq. 9.8, $\mathbf{s}_s = \mathbf{s}_{i+1}$ (but would be confusing to here use $\mathbf{s}_{i+1}$). Hence

$$\frac{\partial \mathbf{s}}{\partial x_i} = \mathbf{T}' \frac{\partial \mathbf{s}_s}{\partial x_i}.$$

The first element of $\mathbf{s}_s$, I, can be written as (cf. Eq. 9.1)

$$I = e^{-\tau_i} I_i + (1 - e^{-\tau_i}) \left(\bar{B}_i + \bar{j_{n,i}} / \bar{\alpha} \right). \tag{16.30}$$

By again using the chain rule, the derivative of I with respect to x_i can be written as

$$\frac{\partial I}{\partial x_i} = \frac{\partial I}{\partial \tau_i}\frac{\partial \tau_i}{\partial x_i} + \frac{\partial I}{\partial j_i}\frac{\partial j_i}{\partial x_i} = e^{-\tau_i}\frac{\partial \tau_i}{\partial x_i}(j_i - I_i) + (1 - e^{-\tau_i})\frac{\partial j_i}{\partial x_i}, \tag{16.31}$$

where j_i is short for $\bar{B}_i + \bar{j_{n,i}} / \bar{\alpha}$. The two remaining derivatives $\partial \tau_i / \partial x_i$ and $\partial j_i / \partial x_i$ depend on the quantity considered, and are discussed further below.

For higher Stokes components of $\mathbf{s}_s$, or in general if emission is totally ignored, Eq. 16.30 is simplified to (here exemplified for the second Stokes element, Q)

$$Q = e^{-\tau_i} Q_i, \tag{16.32}$$

and the chain rule expression is correspondingly shorter:

$$\frac{\partial Q}{\partial x_i} = -e^{-\tau_i}\frac{\partial \tau_i}{\partial x_i} Q_i. \tag{16.33}$$

16.3.7 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 and refraction. Significant impact of a gas through these mechanisms should only be found for water vapour in the lower troposphere, but is a general concern for temperature as discussed in Sec. 16.6.

16.4 Absorption species

16.4.1 Common practicalities

To obtain the Jacobian for absorption species, use jacobianAddAbsSpecies. The method handles one species at the time. The calculations can either be done in an "analytical" or "perturbation" manner. For gases, weighting functions (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.

For the "rel" option 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.

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

16.4.2 Perturbation calculations

For pure numerical calculations, also the size of the perturbation must be specified (Δx in Eq. 16.2). The perturbation shall given following the unit selected. The same value is applied for all WFs (which can cause practical problems ...).

16.4.3 Analytical expressions

If not made clear above, the only term that differs between the Jacobian quantities is $\partial \mathbf{s}/\partial x_i$.

All parameterized models in ARTS are parameterized on basis of cross-section. These models' contribution to Equation16.15 are simply going to be

$$\frac{\partial \mathbf{K}_j}{\partial x_i} = \frac{1}{x_i}\mathbf{K}_j, \tag{16.34}$$

for the parameterized model contributions $\mathbf{K}_j$ that are functions of x_i. One thing of importance to note here is that in collision induced absorption, the absorption caused by the secondary species is also considered in the above expression.

For line-by-line contributions in Equation 16.17, we only consider $\partial S_j/\partial x_i$ as important (when the line is from the absorption species). The same expression as for parameterized models still apply for the partial derivative.

Note that we ignore that $\partial F_j/\partial x_i$ is mathematically non-trivial since pressure broadening and line mixing is changed if the atmosphere is changed. In fact, $\partial F_j/\partial x_i$ is mathematically non-trivial even for lines other than those from the absorption species itself since it changes the total pressure. We justify ignoring this effect because the term is very small in most reasonable atmospheric setups.

16.5 Winds

Calculation of wind weighting functions are triggered by jacobianAddWind. Each wind component (see Sec. 13.1) is treated as an individual retrieval variable. That is, if you want to retrieve all three wind components, jacobianAddWind must be called three times. If you want the total wind Jacobian, this can also be attained. Only the analytically inclined calculation approach is at hand for winds.

Theoretically, the Doppler shift induced by winds affects the emission source term, but this impact is extremely small and the related terms are ignored. This gives a case basically identical to the one above for absorption species.

By design, the absorption calculations of ARTS are unaware of wind velocities. Therefore, only the frequency derivative is calculated at the lowest level. That is, for each component, this equation is evaluated

$$\frac{\partial \mathbf{K}_j}{\partial x_i} = \frac{\partial \mathbf{K}_j}{\partial \nu}\frac{\partial \nu}{\partial x_i}. \tag{16.35}$$

The frequency partial derivative term is

$$\frac{\partial \nu}{\partial x_i} = \frac{1}{c}\frac{\partial v}{\partial x_i}, \tag{16.36}$$

where c the speed of light and v is the wind velocity relative to the observational path of the component of interest. The last partial derivative is there due to the influence of

the angle of observation on the shift in frequency. Look along the direction of the wind component and it is the only component that matters so the partial derivative above is unity. The other two components are therefore perpendicular to the observational path and have partial derivatives that are nil. We will not write the full expression since it depends on atmospheric dimensionality and is different for each of the three components so there are a total of ten different expressions depending on user input.

For Equation 16.17, only $\partial F_j/\partial\nu$ is non-zero. The contribution is

$$\frac{\partial \mathbf{K}_j}{\partial \nu} = \mathbf{K}_j \left(\frac{1}{F_n}\frac{\partial F_n}{\partial \nu} + \frac{1}{F_s}\frac{\partial F_s}{\partial \nu} \right). \tag{16.37}$$

For efficiency, the second term is calculated slightly differently due to efficiency.

There is a way to parameterize classical line shapes by two terms and write

$$F_s = w(a + ib) = F_A + iF_B, \tag{16.38}$$

where a and b are parameters used to determine how much pressure and temperature influence the line shape (see details in the theory guide), F_A is the Voigt line shape and F_B is the Faraday-Voigt line shape. This is usually called the Faddeeva line shape. Using the Faddeeva line shape,

$$\frac{\partial w}{\partial x_i} = \frac{\partial w}{\partial a}\frac{\partial a}{\partial x_i} + i\frac{\partial w}{\partial b}\frac{\partial b}{\partial x_i}, \tag{16.39}$$

and a great convenience is that

$$\frac{\partial w}{\partial b} = -\frac{\partial w}{\partial a}, \tag{16.40}$$

with

$$\frac{\partial w}{\partial a} = 2b \cdot F_B - 2a \cdot F_A - i\left[2b \cdot F_A + 2a \cdot F_B - \frac{2}{\Delta\nu_D \cdot \pi}\right], \tag{16.41}$$

where the last term is from F_u and $\Delta\nu_D$ is the Doppler half width in frequency units. Since the partial derivation within the line shape is often repeated, using just one calculation of $\partial w/\partial a$ and then recalculating $\partial a/\partial x_i$ and $\partial b/\partial x_i$ for different request Jacobian quantities is efficient.

16.6 Atmospheric temperatures

16.6.1 Common practicalities

To obtain WFs for temperatures, 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 originating in 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).

16.6.2 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 3.5). 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.

16.6.3 Analytical expressions

Unpolarized absorption

Compared to atmospheric species, the expressions become here more complex as temperature also affects the propagation path length (Δl_i) and the emission source term (j_i). Accordingly, all terms of Eq. 16.31 are relevant, and the expansion of $\partial\tau_i/\partial x_i$ generates additional terms

$$\frac{\partial I}{\partial x_i} = e^{-\tau_i}\left[\frac{\partial\tau_i}{\partial\alpha_i}\frac{\partial\alpha_i}{\partial x_i} + \frac{\partial\tau_i}{\partial\Delta l_i}\frac{\partial\Delta l_i}{\partial x_i}\right](j_i - I_i) + (1 - e^{-\tau_i})\frac{\partial j_i}{\partial x_i}. \tag{16.42}$$

Terms part of expressions found above are not discussed separately here. The term, $\partial\Delta l_i/\partial x_i$, originates in the constrain of hydrostatic equilibrium, and is set to zero when `hse` is set to "off". Otherwise it is set as derived below. The term $\partial\alpha_i/\partial x_i$ is calculated in a pure numerical manner, by perturbing the temperature. Eq. 9.5 gives

$$\frac{\partial\tau_i}{\partial\Delta l_i} = \frac{\alpha_i + \alpha_{i+1}}{2}. \tag{16.43}$$

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

$$\frac{\partial\Delta l_i}{\partial\bar{T}} = \frac{\Delta l_i}{\bar{T}}. \tag{16.44}$$

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

$$\frac{\partial\Delta l_i}{\partial x_i} = \frac{\Delta l_i}{2T_i}. \tag{16.45}$$

In summary (assuming `hse` set to "on"):

$$\frac{\partial I}{\partial x_i} = e^{-\tau_i}\frac{\Delta l_i}{2}\left[\frac{\partial\alpha_i}{\partial x_i} + \frac{\alpha_i + \alpha_{i+1}}{2T_i}\right](j_i - I_i) + (1 - e^{-\tau_i})\frac{\partial j_i}{\partial x_i}. \tag{16.46}$$

The derivative of the Planck function ($\partial B/\partial x_i$) can be expressed analytically [*Eriksson et al.*, 2002].

The expression for higher Stokes elements, or if emission is totally ignored, is obtained by setting j_i (and $\partial j_i/\partial x_i$) to zero.

General case

Both line strength and line shape change with temperature. The rough expression is

$$\frac{\partial \mathbf{K}_i}{\partial x_i} = \mathbf{K}_i \left(S_j^{-1} \frac{\partial S_j}{\partial x_i} + F_j^{-1} \frac{\partial F_j}{\partial x_i} \right), \tag{16.47}$$

where individual terms of both line shape and line strength are treated separately.

Hydrostatic equilibrium and limitations

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

As mentioned, if `hse` is set to "off", the term $\partial \Delta l_i / \partial x_i$ 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 should be insufficient for limb sounding, where changes even at altitudes below the tangent point can have an influence as the geometrical altitudes of all higher layers is changed through hydrostatic equilibrium. 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.

Non-LTE is presently not supported for HSE.

16.7 Magnetic field

16.7.1 Common practicalities

To obtain WFs for magnetic field components, use jacobianAddMagField. Retrieval grids must be specified. There are six input options for component. These are "u", "v", "w", "theta", "eta", and "strength". The last of these three are derived analytically, the other three small perturbations. Note that the Faraday effect is not supporting magnetic field weighting function calculations at the time of writing this.

16.7.2 The analytical solutions

Strength component

Magnetic field strength affects the center of the line. This means it affects the line strength through changing the state distribution by changing the energy differences between states. It also means it affects the line shape by moving the line away from its original line center. The former effect is ignored because it is very small compared to the latter effect for reasonable atmospheric magnetic fields.

In rough estimation, the magnetic field change the energy state of the molecule by

$$\Delta E = -gH, \tag{16.48}$$

where g is a constant (see the theory guide) and H is the field strength. The frequency change is then

$$\Delta\nu_0 = \frac{1}{h}\left(g'' - g'\right) H, \tag{16.49}$$

where h is Planck's constant and the primes stand for upper (single prime) and lower (double prime) of the state.

Since we ignore line strength partial derivations, both Equations 16.37 and 16.39 applies for the magnetic field strength. The remaining partial derivative is then just

$$\frac{\partial a}{\partial x_i} = \frac{\Delta\nu_0}{\Delta\nu_D}; \quad \frac{\partial b}{\partial x_i} = 0. \tag{16.50}$$

Angular components

The angle along the path of observation (θ) only changes the polarization state. The angle of linear polarization rotation (η) also only changes the polarization state. So

$$\frac{\partial \mathbf{K}_i}{\partial x_i} = \mathbf{K}_i \mathbf{\Phi}^{-1} \frac{\partial \mathbf{\Phi}}{\partial x_i}. \tag{16.51}$$

The derivation of a matrix with respect to a variable is just the derivation of individual parameters, so see the theory guide for the original shape and content of $\mathbf{\Phi}$.

16.8 Non-LTE effects

This is handled passively whenever it is encountered. If non-LTE is off, then the source function is the Planck function, so $\mathbf{j}_{n,i} = 0$ and $\partial \mathbf{j}_{n,i}/\partial x_i = 0$ in Equation 16.27. Otherwise, we must note two things. First is that the non-LTE effect is included in $f(T, \cdots)$ as described above, so we know $\mathbf{K}_i$ and $\partial \mathbf{K}_i/\partial x_i$ equally well as before. The second part is that we have defined $\mathbf{j}_{n,i}$ from Equation 4.11. By this definition, for all effects the NLTE contribution is

$$\frac{\partial \mathbf{j}_{n,i}}{\partial x_i} = B\left[\frac{\partial \mathbf{a}_s}{\partial x_i} \oslash \mathbf{a} - \frac{\partial \mathbf{a}}{\partial x_i} \odot \mathbf{a}_s \oslash (\mathbf{a} \odot \mathbf{a})\right] + \frac{\partial B}{\partial x_i}\left(\mathbf{a}_s \oslash \mathbf{a} - 1\right). \tag{16.52}$$

In practice, the above is only an overview and the calculations are done at a much lower level. In fact

$$\mathbf{j}_{n,i} = \left(\frac{f_s(T, \cdots)}{f(T, \cdots)} - 1\right) B\mathbf{\Phi} \cdot [1,\ 0,\ 0,\ 0], \tag{16.53}$$

where the ratio of f_s/f gives the ratio of the source emission to the absorption due to non-LTE effects. These fs are otherwise as in Equation 16.19 for calculations of $\mathbf{K}_j$.

16.9 Sensor pointing

The term "sensor 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 Jacobians is jacobianAddPointingZa.

The pointing deviation is treated as a time varying variable, then having a polynomial variation. hence, the basis functions described in Section 16.2.2 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 so highly varying that an off-set must be assigned to each spectrum (sometime called "pointing jitter").

The Jacobian 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 Jacobian is 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 the zenith angles in mblock_dlos. The average of a positive and negative shift is determined. The shift to apply (`dza`) should be smaller than the minimum spacing of the zenith angles inmblock_dlos for accurate results. As interpolation is a relative fast operation and a "double-sided" disturbance is used, this option should in general be preferred.

16.10 Sensor frequencies

This class of Jacobians 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 methods to define such Jacobians are jacobianAddFreqShift and jacobianAddFreqStretch. These Jacobians can so far only be determined by applying an interpolation of existing monochromatic data, then shifted `df` from the nominal values.

The methods treat either the "shift" or "stretch" effects. 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 "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 16.2.2, shift and stretch correspond to polynomial order 0 and 1.

Both frequency shift and stretch can be assumed to be time varying, where exactly the same polynomial approach as for pointing is applied. This including the case of setting the order to -1.

16.11 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

$$\mathbf{y} = \mathbf{y}' + \sum_{i=0}^{n} x_i \mathbf{z}_i, \tag{16.54}$$

where $\mathbf{y}'$ is the "baseline free" spectrum, and x_i and $\mathbf{z}_i$ are introduced in Section 16.2.2.

The Jacobian 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 polarizations, viewing directions or measurement blocks, and flags can be set to mimic such assumptions.

For a given set of retrieved x_i, the simplest way to determine the estimated baseline is to perform a multiplication between the relevant parts of the Jacobian and the state vector:

$$\mathbf{K}_p \mathbf{x}_p, \tag{16.55}$$

where p indicates the $n+1$ columns/elements corresponding to x_i.

16.12 Sinusoidal baseline fit

If the baseline has components of sinusoidal character there is also a second option, provided the method jacobianAddSinefit. The baseline is in this method modelled as [*Kuntz et al.*, 1997]

$$\mathbf{y} = \mathbf{y}' + \sum_{i=1}^{n} a_i \sin\left(\frac{2\pi(\nu-\nu_1)}{l_i}\right) + b_i \cos\left(\frac{2\pi(\nu-\nu_1)}{l_i}\right) \tag{16.56}$$

where a_i and b_i are the coefficients to be retrieved, ν is frequency, ν_i is a reference frequency and l_i is period length. The reference frequency (ν_i) is in practice taken as the first frequency of the spectrometer.

The period lengths (l_i) are user input. For each given period length, the corresponding sine and cosine terms are included in the Jacobian. As for the polynomial fit, there exist options to set the baseline to be common between different polarizations, viewing directions or measurement blocks. Equation 16.55 is also applicable for sinusoidal baseline fits.

An alternative way to express the expression above is

$$\mathbf{y} = \mathbf{y}' + \sum_{i=1}^{n} A_i \sin\left(\frac{2\pi(\nu-\nu_1)}{l_i} + \phi_i\right) \tag{16.57}$$

where

$$A_i = \sqrt{a_i^2 + b_i^2} \tag{16.58}$$

and

$$\tan\phi_i = \frac{a_i}{b_i}. \tag{16.59}$$

Equation 16.56 is used to define the weighting functions as this gives a linear retrieval problem, in contrast to if Equation 16.57 would be used, that would require an iterative process to determine A_i and ϕ_i.

Chapter 17

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.

17.1 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, 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.

17.2 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.

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.

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, lat_grid,
                        lon_grid,z_field, refellipsoid, z_surface,
                        25e3 )
}

ybatchCalc

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

Part IV

Radiative transfer: dedicated scattering methods

Chapter 18

Scattering calculations – The DOIT module

The Discrete Ordinate ITerative (DOIT) method is one of the scattering algorithms in ARTS. 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 (Chapter 19) 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 is given in *ARTS Theory*, Chapter 9 and has been published in *Emde et al.* [2004] and in *Emde* [2005].

The workspace methods required for DOIT calculations are implemented in the files `m_scatrte.cc`, `m_cloudbox.cc` and `m_optproperties.cc`. Here we introduce the steps to be performed in a DOIT calculation along with the relevant workspace methods by means of a controlfile example.

18.1 The 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 `controlfiles/artscomponents/doit` directory. The file is called `TestDOIT.arts`. For detailed descriptions of the workspace methods and variables please refer also to the online documentation (start doc-server on your own computer with `arts -s` or use https://www.radiativetransfer.org/stubs/).

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
141009	Moved general model parts to theory guide

18.2 DOIT frame

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

```
DoitInit
```

As the next step we have to calculate the incoming field on the boundary of the cloudbox. This is done using the workspace method DoitGetIncoming:

```
DoitGetIncoming
```

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

```
cloudbox_fieldSetClearsky
```

The grid discretization plays a very significant role in discrete ordinate methods. In spherical geometry the zenith angular grid is of particular importance (cf. *ARTS Theory*, Section 9.6.1). The angular discretization is defined in the workspace method DOAngularGridsSet:

```
DOAngularGridsSet( 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 za_grid and 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 DoitCalc:

```
DoitCalc
```

18.2.1 The DOIT main agenda

Although there are alternatives, the most elegant usage of DOIT involves specifying it within the agenda doit_mono_agenda, which calculates the radiation field inside the cloudbox. The 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
 # Perform iterations: 1. scattering integral. 2. RT calculations with
 #   fixed scattering integral field, 3. convergence test
   cloudbox_field_monoIterate
 }
```

The first method DoitScatteringDataPrepare prepares the single scattering data for use in a DOIT scattering calculation. Namely, it interpolates the data on the requested frequency and 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 iteration is performed in the method cloudbox_field_monoIterate, which includes

- the calculation of the scattering integral field doit_scat_field (requires agendas pha_mat_spt_agenda and doit_scat_field_agenda),
- the radiative transfer calculations in the cloudbox with fixed scattering integral (requires agendas spt_calc_agenda, ppath_step_agenda, and doit_rte_agenda), and
- the convergence test (requires agenda doit_conv_test_agenda).

For details on the agendas involved see Section 18.2.2.

18.2.2 Agendas used in cloudbox_field_monoIterate

There are several methods which can be used in cloudbox_field_monoIterate, 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 (*ARTS Theory*, Equation 9.7) 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 DOAngularGridsSet. 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 DOAngularGridsSet 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 a fine zenith angle grid resolution around 90°is not needed.

Radiative transfer with fixed scattering integral term:

With a fixed scattering integral field the radiative transfer equation can be solved (*ARTS Theory*, Equation 9.9). The workspace method to be used for this calculation is defined in doit_rte_agenda. The most efficient and recommended workspace method is cloudbox_fieldUpdateSeq1D where the sequential update which is described in *ARTS Theory*, Section 9.5 is applied. The workspace method cloudbox_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 cloudbox_fieldUpdateSeq1DPP. This method is not much more efficient than cloudbox_fieldUpdateSeq1D, therefore it is usually better to use cloudbox_fieldUpdateSeq1D since it is more accurate.

```
AgendaSet( doit_rte_agenda ){
     cloudbox_fieldUpdateSeq1D
   # Alternatives:
   # cloudbox_fieldUpdateSeq1DPP
   # i_fieldUpdate1D
   }
```

The optical properties of the particles, i.e., extinction matrix and absorption vector (for all scattering elements) 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 cloudbox_field_monoIterate (see Section 18.2.1).

```
AgendaSet( spt_calc_agenda ){
   opt_prop_sptFromMonoData
}
```

The work space method opt_prop_bulkCalc is then used internally to derived the bulk absorption vector abs_vec and extinction matrix ext_mat from the workspace variable ext_mat_spt. The gas absorption is added internally.

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 *ARTS Theory*, Equation 9.19. 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,
                         cloudbox_field, cloudbox_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,
    #                      cloudbox_field, cloudbox_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, cloudbox_field,
    #                           [2, 4] )
}
```

18.2.3 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
}
```

18.3 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 (cloudbox_fieldUpdateSeq3D). All other workspace methods adapt automatically to the atmospheric dimensionality.

Chapter 19

Scattering calculations – The Monte Carlo scattering module

(This is just a stub for this chapter. So far just rescuing some text from an old (and now deleted) Wiki page.)

The ARTS Monte Carlo scattering module offers an efficient method for single viewing direction radiative transfer calculations in arbitrarily complex 3D cloudy cases. When simulating the observation of detailed 3D cloudy scenarios, reversed Monte Carlo algorithms have several advantages over other methods, such as discrete ordinate and Forward Monte Carlo methods. These features include:

- All computational effort is dedicated to calculating the Stokes vector at the location of interest and in the direction of interest. This is particularly relevant for space-borne remote sensing, where we are only interested in a narrow field of view. This is contrast to DOM methods where the whole radiation field is calculated.
- CPU and memory cost scales more slowly than other methods with grid size. so that large or detailed 3D scenarios are not a problem. This stems from the suitability of Monte Carlo Integration (MCI) for evaluating integrals over highly dimensioned spaces. As well as CPU cost increasing dramatically in 3D DOM applications with the number of grid-points in each dimension, the memory requirements becomes prohibitive at moderate grid sizes due to the requirement that the radiance in every direction must be stored at each grid point.
- Optically thick media are no problem. A feature of reversed Monte Carlo algorithms is that only parts of the atmosphere that actually contribute to the observed radiance are considered in the computation. So where the medium is optically thick due to absorption or scattering, only the parts of the atmosphere closest to the sensor are visited by the algorithm. This contrasts with DOM methods, where, as mentioned above, the whole radiation field is computed. Also, a requirement of DOM methods is that the optical thickness between adjacent grid points must be ¡= 1, which increases the grid-size, and hence cost, of the method.

History

140924 Started by Patrick Eriksson

For the theory behind the ARTS MC module see Section 10 of *ARTS Theory*.

Chapter 20

Radar measurements

ARTS provides some support for modelling measurements of backscattering inside the atmosphere. These radar (or lidar?) measurements deviate for the transmission observations discussed in the previous chapter, as in this case the transmitter and receiver are placed at the same position. Here backscattering is recorded, while in the transmission case the attenuation through the atmosphere is probed. This has the consequence that the measurement data not only span the frequency and a polarisation dimensions, but also has a time (or distance) dimension. That is, the backscattering is basically reported as a function of distance from the sensor, for one or several frequencies frequencies and combinations of transmitted and received polarisations. Accordingly, these data differ in nature from the other types of measurements, and the standard main function (yCalc) is not applicable.

20.1 Single scattering

The fastest method for this observation approach is iyRadarSingleScat. As the name of the method indicates a very important restriction applies, only single scattering is treated (but the attenuation due to particles is also considered). This is frequently an acceptable simplification for precipitation and cloud radars observations, and such observations should be the main applications of this workspace method. The basic measurement approach is the same for lidars, but the assumption of single scattering is much more restrictive for such instruments.

20.1.1 Theory

The transmitted pulses are treated to be monochromatic pencil beams. Effects due to antenna patterns and the geometrical distance between the instrument and the scattering particles are assumed to be treated as a separate "calibration", and the "forward model" treats only the actual backscattering and atmospheric extinction.

For the conditions given above, the backscattered radiation, $\mathbf{s}_b$, can be written as

$$\mathbf{s}_b = \mathbf{T}_h \mathbf{Z}^b \mathbf{T}_a \mathbf{s}_t. \tag{20.1}$$

History	
170609	Started updates following recent changes.
121108	Written by Patrick Eriksson.

The terms in this equation are:

$\mathbf{s}_t$ Stokes vector describing the transmitted pulse.

$\mathbf{T}_a$ A matrix describing the transmission from the receiver to the scattering point (the away direction).

$\mathbf{Z}^b$ The bulk scattering (or phase) matrix value for the backward direction.

$\mathbf{T}_h$ As $\mathbf{T}_a$, but for the reversed direction (the home direction). Note that for vector radiative transfer, in general $\mathbf{T}_a \neq \mathbf{T}_h$.

The (attenuated) "backscatter coefficient" recorded by the receiver is

$$\beta' = \mathbf{p} \cdot \mathbf{s}_b \tag{20.2}$$

where $\mathbf{p}$ is the (normalised) vector describing the polarisation response of the receiver and $\cdot$ signifies the dot product.

The corresponding unattenuated backscattering coefficient is

$$\beta = \mathbf{p} \cdot [\mathbf{Z}(\Omega = 0)\mathbf{s}_t] \tag{20.3}$$

20.1.2 Units

The unit of β (and β') is 1/(m sr^{-1}). For radar applications this is not the most common choice, but is here preferred as it directly matches $\mathbf{Z}$. It is also the standard definition in the lidar community. The more common definition of radar reflectivity is simply $4\pi\beta$.

However, even more common is to report radar data in unit of equivalent reflectivity, Z_e. This quantity is defined as [e.g. *Donovan and van Lammeren*, 2001]

$$Z_e = \frac{4\lambda_r}{\pi^4 |K|^2}\beta \tag{20.4}$$

where λ_r is wavelength of the radar and the "reference dielectric factor" is calculated using the complex refractive index of ice or liquid water, n:

$$K = \frac{n^2 - 1}{n^2 + 2}. \tag{20.5}$$

20.1.3 Practical usage

As for other measurements, the main radiative transfer calculations are performed inside iy_main_agenda. When using iyRadarSingleScat, $\mathbf{s}_b$ is returned for each point of the propagation path, and for each frequency in f_grid. The calculated data are packed into iy. Here the difference to other measurement types emerge. For example, the second row holds $\mathbf{s}_b$ corresponding to the second point of the propagation path (not the second frequency). If f_grid contains several frequencies, the data for the second frequency are placed below (in the row dimension) the data for the first frequency etc.

The polarisation of the transmitted pulses ($\mathbf{s}_b$) are taken from iy_transmitter. iy_transmitter shall be a Stokes vector for each frequency in f_grid, of unit intensity.

The unit of returned data is selected by iy_unit. There are two options "1" and "Ze". For the first option no unit conversion is performed, while for the second option Equation 20.4

is applied on all Stokes elements of $\mathbf{s}_b$. In the later case, liquid water at a user specified temperature is used as reference. That is, K is calculated for the refractive index of water at the specified temperature.

Hence, the basic calculations are performed in standard manner, using iy_main_agenda. However, the deviating data pattern in iy results in that an alternative to yCalc is needed and it is yRadar. The method applies two instrument effects. Firstly, the polarisation response of the receiver is incorporated (Eq. 20.2). Secondly, the data are averaged as a function of "range", following the range_bins. These range bins can be specified either as geometrical altitude or the two-way propagation time. The range binning is described further in the built-in documentation. Further, the data are rearranged into a vector and returned as y.

A number of auxiliary data can be obtained by iyRadarSingleScat, this including β. For a list of possible variables:

```
arts -d iyRadarSingleScat
```

A difference of yRadar, compared to yCalc, is that all auxiliary quantities provided by iyRadarSingleScat are treated.

20.1.4 Jacobian calculations

For better clarity, Eq. 20.1 is expanded to make dependency on particle number densities clear:

$$\mathbf{s}_b = \mathbf{T}_h \sum_i \left(n_i < \mathbf{Z}_i^b > \right) \mathbf{T}_a \mathbf{s}_t. \tag{20.6}$$

where n_i is the particle number density of scattering element i and $< \mathbf{Z}_i^b >$ is the back-scattering cross-section of individual particles.

For an element of the state vector, x_p, that influences $\mathbf{Z}^b$ (beside temperature, see below) the main expression for the Jacobian is:

$$\frac{\partial \mathbf{s}_b}{\partial x_p} = \mathbf{T}_h \frac{\partial \mathbf{Z}^b}{\partial x_p} \mathbf{T}_a \mathbf{s}_t = \mathbf{T}_h \sum_i \left(\frac{\partial n_i}{\partial x_p} < \mathbf{Z}_i^b > \right) \mathbf{T}_a \mathbf{s}_t. \tag{20.7}$$

If for example x_p represents ice water content at an altitude, the expression above covers the effect on $\mathbf{Z}^b$ at the same altitude. There is no effect of x_p on $\mathbf{Z}^b$ at other altitudes, and this part of the Jacobian becomes zero. The terms $\partial n_i / \partial x_p$ are taken from the workspace variable dpnd_field_dx.

The back-scattering has a weak temperature dependency. Some particle size distributions have temperature as input, and Eq. 20.7 is then also valid for temperature. So far both these aspects are ignored, and the temperature Jacobian is set to zero.

Changes in both absorption and scattering species can affect $\mathbf{T}_a$ and $\mathbf{T}_h$. The Jacobian corresponding to this effect is:

$$\frac{\partial \mathbf{s}_b}{\partial x_p} = \frac{\partial \mathbf{T}_h}{\partial x_p} \mathbf{Z}^b \mathbf{T}_a \mathbf{s}_t + \mathbf{T}_h \mathbf{Z}^b \frac{\partial \mathbf{T}_a}{\partial x_p} \mathbf{s}_t \approx 2 \frac{\partial \mathbf{T}_h}{\partial x_p} \mathbf{Z}^b \mathbf{T}_a \mathbf{s}_t. \tag{20.8}$$

The last, approximate, expression is used, which assumes that the attenuation not induces significant polarisation. For this condition $\mathbf{T}_a \approx \mathbf{T}_h$. The derivative $\partial \mathbf{T}_h / \partial x_p$ is calculated following Sec. 16.3

20.2 Multiple scattering

Radar measurements involving multiple scattering can be simulated by using MCRadar. To be written ...

Part V

Retrieval calculations

Chapter 21

Optimal estimation method

ARTS provides functionality to perform retrievals of remote sensing observations using the optimal estimation method (OEM). This section describes the basic usage of the OEM implementation available inside ARTS and aims to complement the example controlfiles distributed with the ARTS source code. For a more detailed presentation of the method itself, the reader may refer to the book by *Rodgers* [2000].

21.1 Formulation

The optimal estimation method solves the retrieval problem of finding an atmospheric state $\vec{x} \in \mathrm{R}^n$ that best matches a given observation vector $\vec{y} \in \mathrm{R}^m$. This is done by fitting a forward model $\mathbf{F} : \vec{x} \mapsto \vec{y}_f$ to the observations $\vec{y}$ in a way consistent with a-priori-assumed properties of the retrieval quantities within $\vec{x}$.

21.1.1 Fundamental assumptions

The OEM is based on a Bayesian formulation of the retrieval problem. The three fundamental assumptions of this formulation are:

1. That the a priori assumed properties of all quantities in $\vec{x}$ can be described by a multivariate Gaussian distribution with mean vector $\vec{x}_a$ and covariance matrix $\mathbf{S}_x$.

2. That the forward model $\mathbf{F}$ is exact up to a zero-mean, Gaussian measurement error with covariance matrix $\mathbf{S}_e$.

3. That the forward model is linear or at most moderately non-linear.

21.1.2 The retrieval as optimization problem

Under the assumptions listed above, the a posteriori distribution $p(\vec{x}|\vec{y})$ of the atmospheric state $\vec{x}$ given the observations in $\vec{y}$ is Gaussian as well. The negative log-likelihood of the a posteriori distribution is found to be

---

History

190903 Created and written by Simon Pfreundschuh

$$\mathcal{L}(\vec{x}) \quad = \frac{1}{2}\left((\vec{x}-\vec{x}_a)^T\mathbf{S}_x^{-1}(\vec{x}-\vec{x}_a)+(\vec{y}-\mathbf{F}(\vec{x}))^T\mathbf{S}_e^{-1}(\vec{y}-\mathbf{F}(\vec{x}))\right). \tag{21.1}$$

The most likely vector $\vec{x}$ given the observations $\vec{y}$, also referred to as *the maximum a posteriori estimator* of $\vec{x}$, can then be found by minimizing $\mathcal{L}(\vec{x})$. The ARTS OEM implementation provides multiple methods to minimize this cost function using ARTS itself as the forward model $\mathbf{F}$.

21.2 Overview

The main OEM functionality within ARTS is implemented by the OEM workspace method. A diagram of the data flow of the OEM WSM is given in Fig. 21.1.

The main input arguments to the WSM call are the workspace variables xa, covmat_sx, y and covmat_se together with the agenda inversion_iterate_agenda. As their names imply, the variables xa and covmat_sx describe the a priori distribution given by mean vector $\vec{x}_a$ (xa) and covariance matrix $\mathbf{S}_x$ (covmat_sx). Similarly, the variables y and covmat_se represent the observation vector $\vec{y}$ and the covariance matrix $\mathbf{S}_e$ (covmat_se). Finally, the jacobian_quantities input variable contains a list of the quantities that should be retrieved from the given observations.

In addition to the variables described above, OEM takes as additional input argument the inversion_iterate_agenda agenda. The role of this agenda is to define the forward model used by the OEM method. This agenda is executed repeatedly during the optimization procedure to simulate the measurement vector and its Jacobian corresponding to the current atmospheric state.

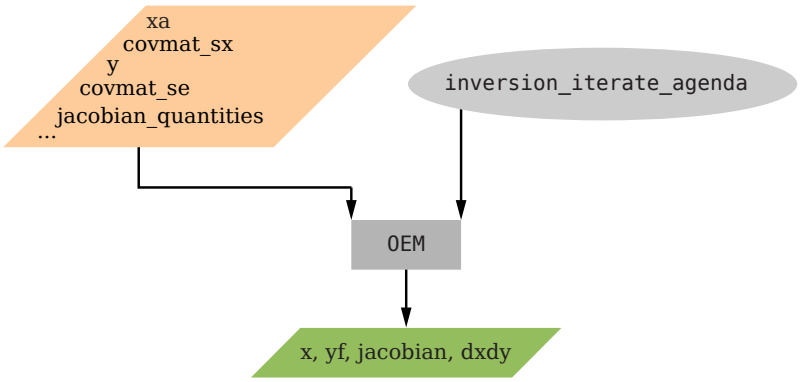


Figure 21.1: Data flow related to the OEM WSM.

21.3 Usage

There are three main steps involved in running the OEM within ARTS:

1. Setup: Defining the measurement space and forward model
2. Running the OEM WSM

3. Calculating relevant diagnostic quantities

Arguably the most complex of these is the setting up of the input variables. Running the OEM requires only a single call to the OEM WSM. Also the calculation of diagnostic quantities can then be performed with a small number of WSM calls.

21.3.1 Setup

Setting up an OEM calculation in ARTS involves three main steps:

1. Defining the state space and a priori distribution (jacobian_quantities, xa, covmat_sx)
2. Defining the observations and measurement errors (y, covmat_se)
3. Defining the forward model (inversion_iterate_agenda)

Defining the state space

In the OEM formulation, the state vector $\vec{x}$ holds the values of all variables that are retrieved. ARTS therefore needs to know which variables from the workspace should be included in $\vec{x}$. This information is contained in the jacobian_quantities WSV. Each entry in jacobian_quantities represents a potentially multi-dimensional retrieval variable that is retrieved using the OEM. The state vector $\vec{x}$ is formed from these variables by concatenation of the flattened values of each retrieval variable.

The setup of jacobian_quantities follows the principle of the setup for regular Jacobian calculations. Before retrieval quantities can be added to jacobian_quantities, the retrievalDefInit WSM must be called. This ensures that jacobian_quantities is empty and that required internal variables are properly initialized. After the call to retrievalDefInit retrieval quantities can be registered by calling any of the available retrievalAdd... methods (retrievalAddAbsSpecies, retrievalAddFreqShift, retrievalAddFreqStretch, retrievalDefInit, retrievalAddCatalogParameter, retrievalAddMagField, retrievalAddPointingZa, retrievalAddPolyfit, retrievalAddScatSpecies, retrievalAddSinefit, retrievalAddSpecialSpecies, retrievalAddSurfaceQuantity, retrievalAddTemperature, retrievalAddWind).

Also within this retrieval definition block, the a priori covariance matrix covmat_sx must be set. This is done using the covmat_sxAddBlock WSM. More details on the handling of covariance matrix and the special requirements on covmat_sx can be found in the group documentation for covariance matrices and the WVS documentation for covmat_sx.

Finally, the retrieval definition is finalized by calling the retrievalDefClose WSM. This method checks that the defined retrieval quantities are consistent with covmat_sx.

An example of how to add ozone as a retrieval quantity for a simple ozone retrieval is given below:

```
retrievalDefInit

# Add retrieval species
retrievalAddAbsSpecies(species = "O3", unit = "vmr")

# Create diagonal block and add to covariance matrix
nelemGet(nelem, p_ret_grid)
VectorSetConstant(vars, nelem, 0.5)
DiagonalMatrix(sparse_block, vars)
```

```
covmat_sxAddBlock(block = sparse_block )

retrievalDefClose
```

Instead of retrieving a quantity directly, it is not uncommon to instead perform the retrieval in a transformed state space. ARTS provides support for arbitrary affine and linear transforms, such as for example retrieving principal components, as well as functional transforms, such as retrieving the logarithm of a quantity. More information on the usage of this functionality can be found in the WSM documentation of the jacobianSetAffineTransformation and jacobianSetFuncTransformation WSMs.

An important point for the user to keep in mind is that for the OEM to work the units of the vectors x and xa and the covariance matrix covmat_sx must be the same. To be able to keep track of how the different retrieval variables are mapped to the x vector the user should refer to the documentation of the available retrievalAdd... methods.

Defining observations and measurement error

The observations to fit should be copied into the y workspace variable and the observation error covariance $\mathbf{S}_e$ into covmat_se. The most important point to keep in mind here is that the dimensions of y and covmat_se must match.

Inversion iteration agenda

The inversion_iterate_agenda constitutes the forward model used by the OEM WSM. This agenda is executed every time Jacobians and observations need to be computed from the forward model during the OEM iterations.

Within inversion_iterate_agenda the following steps must be performed:

1. Transform values in x to the corresponding ARTS WSVs using one of the x2Arts... WSMs.
2. Perform the radiative transfer simulation (for example using yCalc)
3. Store resulting $\mathbf{y}$ vector in yf
4. If any transformation are applied or one of the retrieval quantities is retrieved in relative units, call jacobianAdjustAndTransform for the computed Jacobian to be transformed accordingly

As an example, the definition of the inversion_iterate_agenda for the ozone retrieval is given below.

```
AgendaSet( inversion_iterate_agenda ){

  Ignore(inversion_iteration_counter)

  # Map x to ARTS' variables
  x2artsAtmAndSurf
  x2artsSensor   # No need to call this WSM if no sensor variables retrieved

  # Calculate yf and Jacobian matching x.
  yCalc( y=yf )
```

```
  #Not needed in this simple case.
  #jacobianAdjustAndTransform
}
```

21.3.2 Running OEM

If the ARTS workspace has been setup properly, running an OEM calculation simply requires calling the OEM WSM.

The ARTS OEM calculation can be run in linear or non-linear mode. In linear mode, only a single optimization step is performed, because in this case the Gauss-Newton iteration yields an exact solution already after the first step. If the forward model is non-linear, however, multiple minimization steps are performed. The most important available configuration options relating to the optimization procedure are described below.

Gauss-Newton optimization

In the standard formulation, the Gauss-Newton iteration for the OEM problem takes the following form:

$$\vec{x}_{i+1} = \vec{x}_i - \underbrace{\left(\mathbf{K}_i^T \mathbf{S}_e^{-1} \mathbf{K}_i + \mathbf{S}_x^{-1}\right)}_{\mathbf{H}}{}^{-1} \left(\mathbf{K}_i^T \mathbf{S}_e^{-1} (\mathbf{F}(\vec{x}_i) - \vec{y}) + \mathbf{S}_x^{-1} (\vec{x}_i - \vec{x}_a)\right) \quad (21.2)$$

Here, $\vec{x}_i$ is the state vector of the current iteration step and $\mathbf{K}_i$ the corresponding Jacobian of the forward model $\mathbf{F}$. Each optimization step involves solving a linear system of equations of size $n \times n$, where n is the dimensionality of the state space. The linear system is defined by the matrix $\mathbf{H}$, which approximates the Hessian of the cost function $\mathcal{L}$. The ARTS OEM method provides two methods to solve these linear systems: By default, a QR solver is used which requires to explicitly compute $\mathbf{H}$. If n is very large, however, calculating $\mathbf{H}$ can become computationally very expensive. In this cases it may be advantageous to use a conjugate gradient (CG) solver, which does not require explicitly computing $\mathbf{H}$. More information on how to use the CG solver is given in Sec.21.3.2 below.

Additionally, two further forms of the GN iteration can be derived, the so called *m- and n-form*:

$$\vec{x}_{i+1} = \vec{x}_a - \left(\mathbf{K}_i^T \mathbf{S}_e^{-1} \mathbf{K}_i + \mathbf{S}_x^{-1}\right)^{-1} \mathbf{K}_i^T \mathbf{S}_e^{-1} \left(\mathbf{F}(\vec{x}) - \vec{y} - \mathbf{K}_i(\vec{x} - \vec{x}_a)\right) \quad (21.3)$$

$$= \vec{x}_a + \mathbf{S}_x \mathbf{K}_i^T \left(\mathbf{K}_i \mathbf{S}_x \mathbf{K}_i^T + \mathbf{S}_e\right)^{-1} \left(\mathbf{F}(\vec{x}_i) - \vec{y} - \mathbf{K}_i(\vec{x}_i - \vec{x}_a)\right) \quad (21.4)$$

The names of these two forms derive from the size of the linear system of equations that must be solved in each iteration step, which is $n \times n$ for the first form and $m \times m$ for the second form. Moreover, the forms differ in whether or not they involve the covariance only as its inverse or only directly. In ARTS both the standard and m-form are implemented. More details on the available OEM configurations are provided in the documetation of the OEM WSM.

Levenberg-Marquardt

In the Levenberg-Marquardt method, an additional damping term is added to the matrix $\mathbf{H}$ in Eq. (21.2):

$$\mathbf{H} = \mathbf{K}^T\mathbf{S}_e^{-1}\mathbf{K} + \mathbf{S}_x^{-1} + \gamma\mathbf{D} \tag{21.5}$$

The additional term $\gamma\mathbf{D}$ may be viewed as an adaptive regularization that can help to avoid convergence problems if the forward model is non-linear. The implementation inside ARTS uses the diagonal of the inverse of the covariance matrix $\mathbf{S}_x^{-1}$.

For high values of γ, the optimization method tends to perform gradient-descent-like steps that ensure that the cost $\mathcal{L}$ is reduced in each step. If γ is zero, the iteration step is identical to that of the GN method.

The following logic determines how the value of γ is adapted during the iteration process: When the reduction in $\mathcal{L}$ from an iteration step matches that expected from a quadratic fit to the loss function, the value of γ is decreased. If that is not the case but $\mathcal{L}(\vec{x}_{i+1})$ is still lower than $\mathcal{L}(\vec{x}_i)$, $\vec{x}_{i+1}$ is accepted as next step but γ is kept constant. If, contrarily, $\mathcal{L}(\vec{x}_{i+1})$ is higher than $\mathcal{L}(\vec{x}_i)$, γ is increased and $\vec{x}_{i+1}$ recomputed with the new γ. γ is increased up to a user-defined threshold. If no step is found that leads to a reduction of $\mathcal{L}$ until this threshold is reached, the OEM iteration is aborted.

Conjugate gradient solver

As mentioned above, the ARTS OEM implementation also allows using a conjugate gradient (CG) solver to solve the linear systems occurring in each iteration step of the OEM. Using a CG solver is advantageous for retrieval problems with high-dimensional state and measurement spaces. The CG solver can be used for any of the optimization methods described above and is enable simply by appending the `_cg` suffix to the `method` GIN of the OEM WSM.

Convergence

In the non-linear case the optimization iterations are continued until a convergence criterion is met. This implementation uses the approximation of the χ^2 value from Eq. (5.28) in *Rodgers* [2000], which is described in Eq. (5.31) on the same page:

$$\chi^2 \approx (\vec{x}_{i+1} - \vec{x}_i)^T(\mathbf{K}_i^T\mathbf{S}_e^{-1}(\vec{y} - \mathbf{F}(\vec{x}_i)) - \mathbf{S}_x^{-1}(\vec{x} - \vec{x}_a) \tag{21.6}$$

.

21.3.3 Diagnostic quantities

One of the advantages of the OEM is that it provides several diagnostic quantities that allow a thorough characterization of the retrieval. After a successful iteration the OEM returns the Jacobian in jacobian as well as the gain matrix

$$\mathbf{G} = (\mathbf{K}^T\mathbf{S}_e\mathbf{K} + \mathbf{S}_a^{-1})^{-1}\mathbf{K}^T\mathbf{S}_e^{-1} \tag{21.7}$$

in dxdy. A successful OEM calculation is a precondition for the calculation of any of the diagnostic quantities described below.

Averaging kernel matrix

The averaging kernel matrix is defined as

$$\mathbf{A} = \mathbf{GK}. \tag{21.8}$$

and provides information on the contribution of the measurement and a priori information on the retrieved state. In ARTS, the averaging kernel matrix is computed using the avkCalc WSM.

Smoothing error

The covariance matrix of the smoothing error, i.e. the contribution to the overall error caused by the finite resolution of the observation system is given by

$$\mathbf{S}_s = (\mathbf{A} - \mathbf{I})\mathbf{S}_a(\mathbf{A} - \mathbf{I}) \tag{21.9}$$

In ARTS, the smoothing error can be computed using the covmat_ssCalc workspace method.

Retrieval noise

The second component that contributes to the overall retrieval error is the error caused by random errors in the measurement $\vec{y}$. The covariance of this retrieval noise is given by

$$\mathbf{S}_o = \mathbf{G}\mathbf{S}_e\mathbf{G}^T. \tag{21.10}$$

In ARTS, the retrieval error covariance matrix can be computed using the covmat_soCalc workspace method. For this, the user should keep in mind that for $\mathbf{S}_o$ to provide a realistic description of the error caused by measurement errors, $\mathbf{S}_e$ must take into account all forward model errors that cause deviations between the simulated and true observations. The covariance of the total retrieval error can be obtained by adding the covariance matrices of the smoothing error and the retrieval noise covariance matrix.

Part VI

Bibliography and Appendices

Bibliography

Brančík, L., Techniques of matrix exponential function derivative for electrical engineering simulations, in *Proceedings of IEEE International Conference on Industrial Technology*, pp. 2608–2613, Mumbai (India): Indian Institute of Technology Bombay, 2006.

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.

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.

Donovan, D. P., and A. C. A. P. van Lammeren, Cloud effective particle size and water content profile retrievals using combined lidar and radar observations 1. Theory and examples, *Journal of Geophysical Research*, *106*, 27425–27448, 2001.

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.

Goldstein, D., *Polarized light*, chap. The Stokes parameters and Mueller matrices for optical activity and Faraday rotation, Marcel Dekker, Inc., USA, 2003.

Gordon, I. E., et al., The HITRAN2016 molecular spectroscopic database, *Journal of Quantitative Spectroscopy and Radiative Transfer*, *203*, 3–69, 2017.

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.

Kraus, J. D., *Radio astronomy*, McGraw-Hill Book Company, 1966.

Kuntz, M., G. Hochschild, and R. Krupa, Retrieval of ozone mixing ratio profiles from ground-based millimeter wave measurements disturbed by standing waves, *Journal of Geophysical Research*, *102*, 21965–21975, 1997.

Larsson, R., and B. Lankhaar, Zeeman effect splitting coefficients for ClO, OH and NO in some earth atmosphere applications, *Journal of Quantitative Spectroscopy and Radiative Transfer*, *224*, 107050, 2020.

Larsson, R., S. A. Buehler, P. Eriksson, and J. Mendrok, A treatment of the Zeeman effect using Stokes formalism and its implementation in the Atmospheric Radiative Transfer Simulator (ARTS), *Journal of Quantitative Spectroscopy and Radiative Transfer*, *133*, 445–453, 2014.

Larsson, R., B. Lankhaar, and P. Eriksson, Updated Zeeman effect splitting coefficients for molecular oxygen in planetary applications, *Journal of Quantitative Spectroscopy and Radiative Transfer*, *224*, 432–438, 2019.

Mätzler, C., MATLAB functions for mie scattering and absorption, *Tech. rep.*, 2002, iAP Res. Rep. No. 02-08.

Meissner, T., and F. J. Wentz, Polarization rotation and the third stokes parameter: the effects of spacecraft attitude and faraday rotation, *IEEE Transactions on Geoscience and Remote Sensing*, *44*, 506–515, 2006.

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.

Newell, A. C., and R. C. Baird, Absolute determination of refractive indices of gases at 47.7 Gigahertz, *J. Appl. Phys.*, *36*, 1965.

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.

Richard, C., et al., New section of the HITRAN database: Collision-induced absorption (CIA), *Journal of Quantitative Spectroscopy and Radiative Transfer*, *113*, 1276–1285, 2012.

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`.

Rybicki, G. B., and A. P. Lightman, *Radiative processes in astrophysics*, chap. Plasma effects, John Wiley and Sons, Inc., USA, 1979.

Wiscombe, W. J., Improved Mie scattering algorithms, *Applied Optics*, *19*, 1505–1509, 1980.

Wright, P., S. Quegan, N. Wheadon, and C. Hall, Faraday rotation effects on l-band spaceborne sar data, *IEEE Transactions on Geoscience and Remote Sensing*, *41*, 2735–2744, 2003.

Yurkin, M. A., and A. G. Hoekstra, The discrete-dipole-approximation code ADDA: Capabilities and known limitations, *Journal of Quantitative Spectroscopy and Radiative Transfer*, *112*, 2234–2247, 2011.

Part VII

Index

Index