ARTS User Guide

by

Stefan Bühler[1], Patrick Eriksson[2], Wolfram Haas, Nikolay Koulev[1], Thomas Kuhn[1], Oliver Lemke[1]

February 4, 2009
ARTS Version 1.0.214

This is a working document. The implementation approaches and the algorithms are preliminary and can be subject to changes. In addition, not all features described in this document are implemented in ARTS.

We welcome gladly comments and reports on errors in the document. Send then an e-mail to: `patrick@rss.chalmers.se` **or** `sbuehler@uni-bremen.de`**.**

[1]Institute of Environmental Physics, University of Bremen, Germany
[2]Department of Radio and Space Science, Chalmers University of Technology, Sweden

Copyright (C) 2000,2001
Stefan Buehler <sbuehler@uni-bremen.de>
Patrick Eriksson <patrick@rss.chalmers.se>

The ARTS program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with the program; if not, write to the Free Software Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.

Contents

Chapter 1

The ARTS concept

This section describes the basic ideas underlying ARTS. It also introduces some terminology. You should read it if you want to understand how the program works and how it can be used efficiently. At the end of the chapter, there is also some practical information about useful command line parameters and such things.

1.1 Introduction

The number of satellite sensors in the millimeter and sub-millimeter spectral range is rapidly growing. They use various frequency bands and observation geometries. Two important groups of sensors are for example the nadir viewing millimeter wave sensors like AMSU[1] and the limb viewing sub-millimeter wave sensors like the planned SMILES[2].

For the data analysis all such sensors require accurate and fast forward models, which can simulate measurements for a given atmospheric (and maybe ground) state. Depending on the objective of the sensor, the measurement will depend for example on the distribution of atmospheric temperature, water vapor, ozone, and many other trace gases.

So far, a lot of effort has been wasted in developing dedicated forward models for different sensors, although all these models have many features in common. Moreover, existing models were not easily modifiable and extendable. Hence, it was decided to develop a new model which emphasizes modularity, extendibility, and generality.

[1]The **A**dvanced **M**icrowave **S**ounding **U**nit is a sensor on board the polar orbiting satellites of the US-American National Aeronautics and Space Administration.

[2]The **S**uperconducting Sub-**Mi**llimeter Wave **L**imb **E**mission **S**ounder is a Japanese Sensor which will be flown for the first time on the International Space Station.

History	
000616	Created by Stefan Buehler, based on my DPG2000 poster.
011121	Practical hints added by Stefan Buehler.

1.2 The scope of ARTS

The present version of ARTS is limited to cases where scattering can be neglected and local thermodynamic equilibrium applies. ARTS has been developed having passive emission measurements in mind, put pure transmission (that is, the emission is neglected) observations are also handled. The forward model can be used to simulate measurements for all (normal?) observation geometries: ground-based, nadir looking, limb sounding and balloon/aircraft measurements. It can be noted that ARTS handles measurements from a point inside the atmosphere, such as an aircraft or a balloon, in a downward direction. ARTS covers so far only monochromatic pencil beam calculations, that is, no sensor characteristics can be included. This part is presently covered by the AMI (ARTS Matlab interface) set of Matlab functions (see below). Sensor characteristics will be included in ARTS.

Beside providing set of spectra, ARTS calculates weighting functions for a number of variables. Analytical expressions for the weighting functions are used for species, continuum absorption and ground emission, and for temperature if hydrostatic equilibrium is *not* assumed. Weighting functions are also provided for pointing off-sets, calibration and temperature (with hydrostatic equilibrium).

For Matlab users there are two accompanying packages called AMI and Qpack[3] which extends the usage of ARTS considerably. First of all, AMI has functions to include sensor characteristics in the calculations. AMI has further functions to read and write ARTS data file, and various functions that are of general usage. Qpack is an Matlab environment to perform OEM inversions and producing set of spectra to test the inversions, where ARTS is used as calculating engine.

1.3 Enter: ARTS

The most important notion in ARTS is the *workspace*. All physical quantities (for example absorption coefficients) are *workspace variables*. But workspace variables can also be of a more technical nature, for example various grids.

The program performs a calculation by executing a list of *workspace methods*, which are specified in a controlfile. These workspace methods take workspace variables as input, and generate workspace variables as output. Additional input parameters can be specified as *keyword parameters* in the controlfile (Figure 1.1).

It is important to note that the controlfile has a fixed and well-defined syntax. This syntax is understood by the ARTS parser. The great advantage of this concept is that it is very easy to add new workspace variables and new workspace methods. The program has an internal lookup table which lists all workspace methods, as well as their input variables, output variables, and keyword parameters. To add a new method, one just has to add an entry to this lookup table, and write the code for the method itself. No further changes to the program are necessary. In particular, no changes to the program logic or to the parser. How such an extension can be made practically is described in Section 11.

[3] AMI is distrubuted by ARTS, while Qpack is a separate package

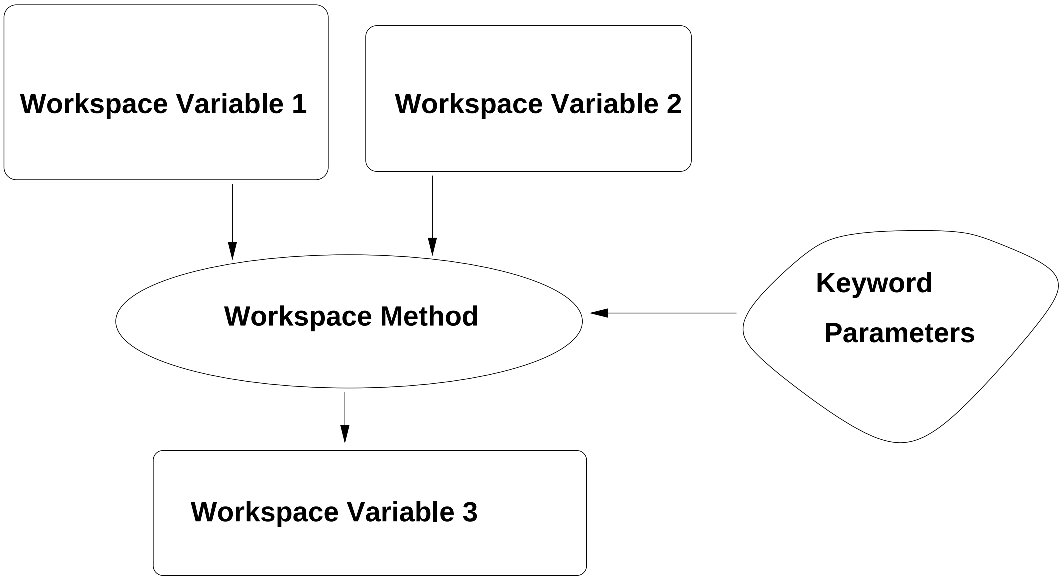

Figure 1.1: *Specific* workspace methods act on specific workspace variables to generate other specific workspace variables. Additional input parameters can be specified as keyword parameters in the controlfile.

1.4 Generic Workspace Methods

Generic methods (Figure 1.2) allow the user of the program even more freedom than specific methods. A generic method is for example `VectorReadAscii`, which can be used to read any workspace variable which is a vector from an ASCII file. For example

```
VectorReadAscii(f_mono){"freqeuency_grid.aa"}
```

will read the specified file and generate the workspace variable `f_grid`.

Generic methods are particularly useful for IO operations like in the example above. No new IO functions are necessary for new workspace variables, as long as they are of standard types already known to the program (for example vectors or matrices).

1.5 Practical hints

The subdirectory `examples` of the `doc` directory contains some example controlfiles. You should study them to learn more about how the program works. You can also run these controlfiles like this:

```
arts absorption_example.arts
```

This assumes that you are inside the directory where the controlfiles are, and that the `arts` executable is in your path. You can also run all of the examples, by saying

```
make check
```

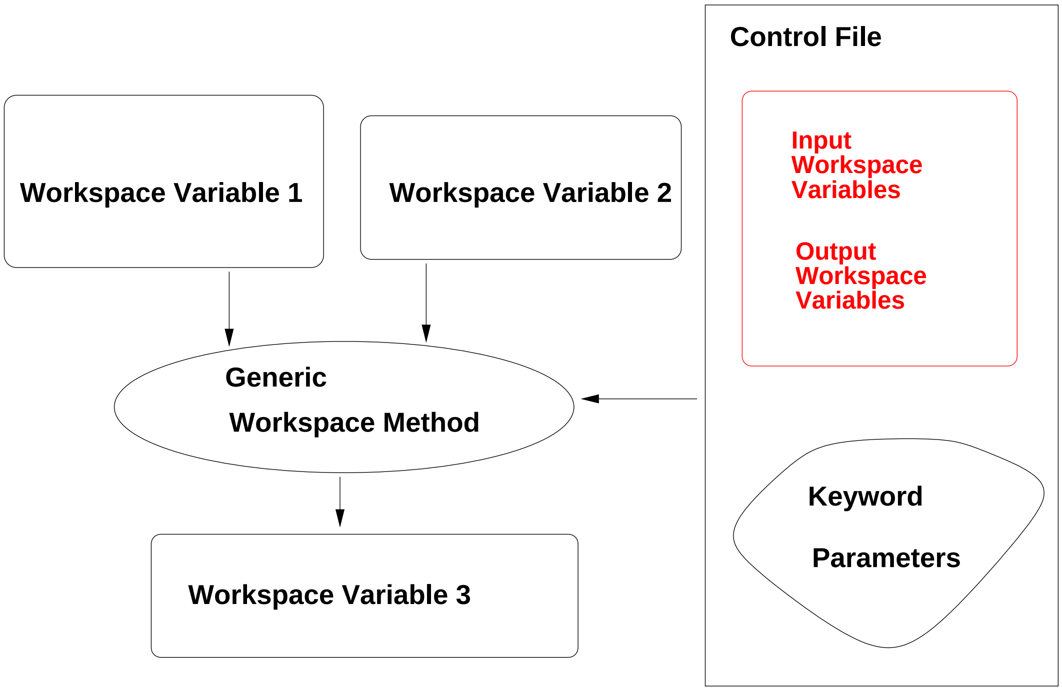

Figure 1.2: For *generic* workspace methods the workspace variables to act on are specified in the controlfile.

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, type

```
arts -h
```

or

```
arts --help
```

Most useful at the beginning should be the `-d` (`--describe`), `-m` (`--methods`), `-w` (`--workspacevariables`), and `-i` (`--input`) flags. For instance, the `-d` (`--describe`) flag gives you online documentation for any workspace method or workspace variable. Usage:

```
arts -d f_mono
```

will print documentation about the workspace variable `f_mono`, which happens to be the monochromatic frequency grid.

But what methods and variables are available? You can find out by typing

```
arts -m all
```

which will list all workspace methods, or by typing

```
arts -w all
```

which will list all workspace variables. As you can see, these lists are quite long. But you can get more specific information:

```
arts -m f_mono
```

will give you a list of all methods that can generate the workspace variable `f_mono`. Specific and generic methods are listed separately. Generic methods are in this case all methods producing a Vector, since `f_mono` belongs to this group. A similar task is performed by the `-i` (`--input`) flag, with the difference that `arts -i f_mono` will list those methods that require `f_mono` as *input*, whereas `arts -m f_mono` lists those that produce `f_mono` as output. Finally,

```
arts -w absCalc
```

will give you all variables required by the method `absCalc` (the variable `f_mono` happens to be one of them).

Using these command line parameters, it is easy to build up a controlfile. The trick is, to start at the end. Say you want to compute absorption coefficients. First of all, you have to find out in which workspace variable these are stored. Look at the list produced by `arts -w all`. You can use `arts -d` to look at some candidates a bit more closely. This way, you will find out that `abs` is the variable you are looking for.

In the next step, you can use `arts -m abs` to find all methods that can calculate `abs`. So, you will find the method `absCalc`. Now you can use `arts -w absCalc` to find out the required input variables of that method. Then you can use the `-m` flag again, to find the methods producing these variables, and so on.

Chapter 2

Theoretical formalism

In this section a theoretical framework for the forward model is presented. The presentation follows *Rodgers* [1990], but some extensions are made, for example, the distinction between the atmospheric and sensor parts of the forward model is also discussed. After this chapter was written, C.D. Rodgers published a textbook [*Rodgers*, 2000] presenting the formalism in more detail than *Rodgers* [1990]. Modelling of sensor characteristics is not yet included in ARTS (this part is so far covered by AMI, see Section 1.2), but treatment of the sensor is here included for completness.

2.1 The forward model

The radiative intensity, I, at a point in the atmosphere, r, for frequency ν and traversing in the direction, ϕ, is dependent on a variety of physical processes and continuous variables such as the temperature profile, T:

$$I = F(r, \nu, \phi, T, \ldots) \tag{2.1}$$

To detect the spectral radiation some kind of sensor, having a finite spatial and frequency resolution, is needed, and the observed spectrum becomes a vector, $\mathbf{y}$, instead of a continuous function. The atmospheric radiative transfer is simulated by a computer model using a limited number of parameters as input, and the forward model, $\mathcal{F}$, used in practice can be expressed as

$$\mathbf{y} = \mathcal{F}(\mathbf{x}_\mathcal{F}, \mathbf{b}_\mathcal{F}) + \varepsilon(\mathbf{x}_\varepsilon, \mathbf{b}_\varepsilon) \tag{2.2}$$

where $(\mathbf{x}_\mathcal{F}, \mathbf{b}_\mathcal{F})$ and $(\mathbf{x}_\varepsilon, \mathbf{b}_\varepsilon)$ together give a total description of both the atmospheric and sensor states, and ε is the measurement errors. The parameters are divided in such way that

History

000306 Written by Patrick Eriksson, partly based on *Eriksson* [1999] and *Eriksson et al.* [2000].

$\mathbf{x}$, the state vector, contains the parameters to be retrieved, and the remainder is given by $\mathbf{b}$, the model parameter vector. The total state vector is

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_{\mathcal{F}} \\ \mathbf{x}_{\varepsilon} \end{bmatrix} \tag{2.3}$$

and the total model parameter vector is

$$\mathbf{b} = \begin{bmatrix} \mathbf{b}_{\mathcal{F}} \\ \mathbf{b}_{\varepsilon} \end{bmatrix} \tag{2.4}$$

The actual forward model consists of either empirically determined relationships, or numerical counterparts of the physical relationships needed to describe the radiative transfer and sensor effects. The forward model described here is mainly of the latter type, but some parts are more based on empirical investigations, such as the parameterisations of continuum absorption.

Both for the theoretical formalism and the practical implementation, it is suitable to make a separation of the forward model into two main sections, a first part describing the atmospheric radiative transfer for pencil beam (infinite spatial resolution) monochromatic (infinite frequency resolution) signals [*Eriksson*, 1999],

$$\mathbf{i} = \mathcal{F}_a(\mathbf{x}_r, \mathbf{b}_r) \tag{2.5}$$

and a second part modelling sensor characteristics,

$$\mathbf{y} = \mathcal{F}_s(\mathbf{i}, \mathbf{x}_s, \mathbf{b}_s) + \varepsilon(\mathbf{x}_\varepsilon, \mathbf{b}_\varepsilon) \tag{2.6}$$

where $\mathbf{i}$ is the vector holding the spectral values for the considered set of frequencies and viewing angles ($\mathbf{i}^i = I(\nu^i, \phi^i)$, where i is the vector index), and $\mathbf{x}_{\mathcal{F}}$ and $\mathbf{b}_{\mathcal{F}}$ are separated correspondingly, that is, $\mathbf{x}_{\mathcal{F}}^T = [\mathbf{x}_r^T, \mathbf{x}_s^T]$ and $\mathbf{b}_{\mathcal{F}}^T = [\mathbf{b}_r^T, \mathbf{b}_s^T]$. The vectors $\mathbf{x}$ and $\mathbf{b}$ can now be expressed as

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_r \\ \mathbf{x}_s \\ \mathbf{x}_\varepsilon \end{bmatrix} \tag{2.7}$$

and

$$\mathbf{b} = \begin{bmatrix} \mathbf{b}_r \\ \mathbf{b}_s \\ \mathbf{b}_\varepsilon \end{bmatrix}, \tag{2.8}$$

respectively.

The subscripts of $\mathbf{x}$ and $\mathbf{b}$ are below omitted as the distinction should be clear by the context.

2.2 The sensor transfer matrix

The modelling of the different sensor parts can be described by a number of of analytical expressions (see *Eriksson and Merino* [1997]) that together makes the basis for the sensor model. These expressions are throughout linear operations and it possible, as suggested in *Eriksson et al.* [2000], to implement the sensor model as a straightforward matrix multiplication:

$$\mathbf{y} = \mathbf{H}\mathbf{i} + \varepsilon \tag{2.9}$$

where $\mathbf{H}$ is here denoted as the sensor transfer matrix. The matrix $\mathbf{H}$ further incorporate effects of a data reduction and the total transfer matrix is then

$$\mathbf{H} = \mathbf{H}_d\mathbf{H}_s \tag{2.10}$$

as

$$\mathbf{y} = \mathbf{H}_d\mathbf{y}' = \mathbf{H}_d(\mathbf{H}_s\mathbf{i} + \varepsilon') = \mathbf{H}\mathbf{i} + \varepsilon \tag{2.11}$$

where $\mathbf{H}_d$ is the reduction matrix, $\mathbf{H}_s$ the sensor matrix, and $\mathbf{y}'$ and ε' are the measurement vector and the measurement errors, respectively, before data reduction. The matrices $\mathbf{H}_d$ and $\mathbf{H}_s$ are described in Section 7 and 8, respectively.

2.3 Weighting functions

2.3.1 Basics

A weighting function is the partial derivative of the spectrum vector $\mathbf{y}$ with respect to some variable used by the forward model. As the input of the forward model is divided between $\mathbf{x}$ or $\mathbf{b}$, the weighting functions are divided correspondingly between two matrices, the state weighting function matrix

$$\mathbf{K}_\mathbf{x} = \frac{\partial \mathbf{y}}{\partial \mathbf{x}} \tag{2.12}$$

and the model parameter weighting function matrix

$$\mathbf{K}_\mathbf{b} = \frac{\partial \mathbf{y}}{\partial \mathbf{b}} \tag{2.13}$$

For the practical calculations of the weighting functions, it is important to note that the atmospheric and sensor parts can be seperated. For example, if $\mathbf{x}$ only hold atmospheric and spectroscopic variables, $\mathbf{K}_\mathbf{x}$ can be expressed as

$$\mathbf{K}_\mathbf{x} = \frac{\partial \mathbf{y}}{\partial \mathbf{i}}\frac{\partial \mathbf{i}}{\partial \mathbf{x}} = \mathbf{H}\frac{\partial \mathbf{i}}{\partial \mathbf{x}} \tag{2.14}$$

This equation shows that the new parts needed to calculate atmospheric weighting functions, are functions giving $\partial\mathbf{i}/\partial\mathbf{x}$ where $\mathbf{x}$ can represent the vertical profile of a species, atmospheric temperature, spectroscopic data etc.

2.3.2 Transformation between vector spaces

It could be of interest to transform a weighting function matrix from one vector space to another[1]. The new vector, $\mathbf{x}'$, is here assumed to be of length n ($\mathbf{x}' \in \mathbf{R}^{n\text{x}1}$), while the original vector, $\mathbf{x}$ is of length p ($\mathbf{x} \in \mathbf{R}^{p\text{x}1}$). The relationship between the two vector spaces is described by a transformation matrix $\mathbf{B}$:

$$\mathbf{x} = \mathbf{B}\mathbf{x}' \tag{2.15}$$

where $\mathbf{B} \in \mathbf{R}^{p\text{x}n}$. For example, if $\mathbf{x}'$ is assumed to be piecewise linear, then the columns of $\mathbf{B}$ contain tenth functions, that is, a function that are 1 at the point of interest and decreases linearly down to zero at the neighbouring points. The matrix can also hold a reduced set of eigenvectors.

The weighting function matrix corresponding to $\mathbf{x}'$ is

$$\mathbf{K}_{\mathbf{x}'} = \frac{\partial \mathbf{y}}{\partial \mathbf{x}'} \tag{2.16}$$

This matrix is related to the weighting function matrix of $\mathbf{x}$ (Eq. 2.12) as

$$\mathbf{K}_{\mathbf{x}'} = \frac{\partial \mathbf{y}}{\partial \mathbf{x}}\frac{\partial \mathbf{x}}{\partial \mathbf{x}'} = \frac{\partial \mathbf{y}}{\partial \mathbf{x}}\mathbf{B} = \mathbf{K}_{\mathbf{x}}\mathbf{B} \tag{2.17}$$

Note that

$$\mathbf{K}_{\mathbf{x}'}\mathbf{x}' = \mathbf{K}_{\mathbf{x}}\mathbf{B}\mathbf{x}' = \mathbf{K}_{\mathbf{x}}\mathbf{x} \tag{2.18}$$

However, it should be noted that this relationship only holds for those $\mathbf{x}$ that can be represented perfectly by some $\mathbf{x}'$ (or vice versa), that is, $\mathbf{x} = \mathbf{B}\mathbf{x}'$, and not for all combinations of $\mathbf{x}$ and $\mathbf{x}'$.

If $\mathbf{x}'$ is the vector to be retrieved, we have that [*Rodgers*, 1990]

$$\hat{\mathbf{x}}' = \mathcal{I}(\mathbf{y}, \mathbf{c}) = \mathcal{T}(\mathbf{x}, \mathbf{b}, \mathbf{c}) \tag{2.19}$$

where $\mathcal{I}$ and $\mathcal{T}$ are the inverse and transfer model, respectively.

The contribution function matrix is accordingly

$$\mathbf{D}_{\mathbf{y}} = \frac{\partial \hat{\mathbf{x}}'}{\partial \mathbf{y}} \tag{2.20}$$

that is, $\mathbf{D}_{\mathbf{y}}$ corresponds to $\mathbf{K}_{\mathbf{x}'}$, not $\mathbf{K}_{\mathbf{x}}$.

We have now two possible averaging kernel matrices

$$\mathbf{A}_{\mathbf{x}} = \frac{\partial \hat{\mathbf{x}}'}{\partial \mathbf{x}} = \frac{\partial \hat{\mathbf{x}}'}{\partial \mathbf{y}}\frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \mathbf{D}_{\mathbf{y}}\mathbf{K}_{\mathbf{x}} \tag{2.21}$$

$$\mathbf{A}_{\mathbf{x}'} = \frac{\partial \hat{\mathbf{x}}'}{\partial \mathbf{x}'} = \frac{\partial \hat{\mathbf{x}}'}{\partial \mathbf{y}}\frac{\partial \mathbf{y}}{\partial \mathbf{x}}\frac{\partial \mathbf{x}}{\partial \mathbf{x}'} = \mathbf{D}_{\mathbf{y}}\mathbf{K}_{\mathbf{x}'} = \mathbf{A}_{\mathbf{x}}\mathbf{B} \tag{2.22}$$

where $\mathbf{A}_{\mathbf{x}} \in \mathbf{R}^{p\text{x}n}$ and $\mathbf{A}_{\mathbf{x}'} \in \mathbf{R}^{p\text{x}p}$, that is, only $\mathbf{A}_{\mathbf{x}'}$ is square. If $p > n$, $\mathbf{A}_{\mathbf{x}}$ gives more detailed information about the shape of the averaging kernels than the standard matrix ($\mathbf{A}_{\mathbf{x}'}$). If the retrieval grid used is coarse, it could be the case that $\mathbf{A}_{\mathbf{x}'}$ will not resolve all the oscillations of the averaging kernels, as shown in *Eriksson* [1999, Figure 11].

[1] This subject is also discussed in *Rodgers* [2000] (published after writing this).

Part I

Algorithm Descriptions

Chapter 3

Gas Absorption

In general there are three types of absorption/emission spectra:

- sharp lines of finite width
- aggregations (series) of lines called bands
- continua extending over a broad range of wavelengths, with no single lines in it.

This section is therefore firstly devised to give a short theoretical overview of the absorption quantities which are used in ARTS. The presentation of the calculation methods within ARTS of these quantities is the second goal. There are two major subsections which deal separately with the respective types of spectra regarded by ARTS, lines and continua. The second one extends beyond the scope of mere treatment of the continua and thus gives an integrated view of continua and spectral lines.

3.1 Line Absorption

We will introduce here the main concepts concerning line absorption. The approach, however, does not aim at the derivation, but rather at the presentation the following expressions.

Any line is presented by the corresponding profile of the absorption/emission coefficient as a function of the frequency, in our case that of the absorption - given by the quantity $\alpha(\nu)$. The latter dependency is related to other quantities, which express the three characteristics of a single line uniquely describing it - position, strength and shape. The strength, or better the line intensity, is given by the quantity $S(T)$, where T is absolute temperature. The shape and the position are expressed through the line-shape function $F(\nu)$. Thus we have the relation

$$\alpha(\nu) = nS(T)F(\nu) \tag{3.1}$$

History

2001-07-05	Template created by Stefan Buehler.
2001-10-05	Line absorption part written, Nikolay Koulev
2001-11-21	Continuum absorption part written, Thomas Kuhn.

where n is the number density of the absorber, *Goody and Yung* [1989]. The line-shape function is normalized as follows

$$\int F(\nu)d\nu = 1 \tag{3.2}$$

The values of $S(T)$ at reference temperature T_0 are contained in spectroscopic databases. The conversion to different temperatures is done by

$$S(T) = S(T_0)\,\frac{Q(T_0)}{Q(T)}\,\frac{e^{-E_f/(kT)} - e^{-E_i/(kT)}}{e^{-E_f/(kT_0)} - e^{-E_i/(kT_0)}} \tag{3.3}$$

given the energies E_f and E_i of the two levels between which the transition occurs as well as the partition function $Q(T)$, *Rothman et al.* [1998]. The databases contain the lower state energy E_l tabulated along with the S and the transition frequency ν, so that the upper state energy can be computed by E_u=E_l+hν. Partition functions for all molecular species are also available along with spectroscopic databases, either in the form of tabulated values for a set temperatures, or in the form of FORTRAN routines. One can obtain the total absorption coefficient by adding the absorption of all spectral lines of all molecular species.

The problem of determining the explicit kind of the line-shape functions is treated in the following subsection. Another one is dedicated to the partition functions and their calculation..

3.1.1 Line Shape Functions

Up to date there is no way to calculate the line-shape functions analytically just from the theoretical expressions of quantum dynamics, statistical physics, and theoretical mechanics. Certain approximations have to be made to account for different physical phenomena related to the absorption/emission process or to suit better the calculation of different parts of the spectral line.

There are three phenomena which contribute to the line-shape. These are, in increasing order of importance, the finite lifetime of an excited state in an isolated molecule, the thermal movement of the gas molecules, and their collisions with each other. They result in corresponding effects to the line-shape: natural broadening, Doppler, and pressure broadening. Of these, the first one is completely negligible compared to the other two. Nevertheless, we will pay a special attention to the natural broadening because its implications are of a conceptual importance for the broadening processes.

The spectral line shape can be derived in the case of natural broadening from basic physical considerations and a well-known Fourier transform theorem from the time to the frequency domain, *Thorne et al.* [1999]. If we consider classically the spontaneous decay of the excited state of two-level system in the absence of external radiation, then population n of these level decreases according to

$$\frac{dn(t)}{dt} = -A\,n(t) \tag{3.4}$$

where A is Einstein A coefficient, which equation can also be interpreted as the rate of the spontaneously emitted photons because of decay. This relation is more usefully in this case to be written in the following manner

$$n(t) = n(0)\,e^{-At} = n(0)e^{-t/\tau} \tag{3.5}$$

where τ is the mean lifetime of the excited state. Thus the number of spontaneously emitted photons and in this way the flux of the emitted radiation then will be proportional to n. Therefore we can write for the flux L that

$$L(t) = L(0)\ e^{-t/\tau} = L(0)\ e^{-\gamma t} \tag{3.6}$$

By the afore mentioned theorem, multiplying in the time domain by $e^{-\gamma t}$ is equivalent to convolving in the frequency domain with a function $1/[\nu^2 - (\gamma/4\pi)^2]$. Accordingly the line profile of a spectral line at frequency ν_0 as a normalized line-shape function will be, as difined in *Thorne et al.* [1999],

$$F(\nu) = \frac{1}{\pi} \frac{\gamma/4\pi}{(\nu - \nu_0)^2 + (\gamma/4\pi)^2} \tag{3.7}$$

This gives a bell-shaped profile and the function itself is called Lorentzian. The dependence on the position of the line is apparent through ν_0, that is why some authors prefer to denote the function by $F(\nu, \nu_0)$. The result is important because of two major reasons. Firstly, without the natural broadening the line would be the delta function $\delta(\nu - \nu_o)$, as pointed out in *Bernath* [1995]. So the spontaneous decay of the excited state is responsible for the finite width and the certain shape of the line-shape function. Secondly, the Lorentzian type of function comes significantly into play when explaining some of the other broadening effects or the complete picture of the broadened line, *Thorne et al.* [1999].

The second effect, Doppler broadening, is important for the upper stratosphere and mesosphere for microwave frequencies. The line-shape follows the velocity distribution of the particles. Under the conditions of thermodynamic equilibrium, we have a probability distribution for the relative velocity u between the gas molecule and the observer of Maxwell type

$$p(u) = \sqrt{\frac{m}{2\pi kT}}\ exp\left[-\frac{mu^2}{2kT}\right] \tag{3.8}$$

where m is the mass of the molecule. Using then the formula for the Doppler shift for the non-relativistic region ν- ν_0 = $\nu_0 u$ / c , one can easily derive the line-shape function, *Bernath* [1995],

$$F_D(\nu) = \frac{1}{\gamma_D\sqrt{\pi}}\ exp\left[-\left(\frac{\nu - \nu_0}{\gamma_D}\right)^2\right] \tag{3.9}$$

where the quantity γ_D is called Doppler line width and equals

$$\gamma_D = \frac{\nu}{c}\sqrt{\frac{2kT}{m}} \tag{3.10}$$

In contrast to the afore mentioned line-shape function for the natural broadening the Doppler broadening is expressed by a Gaussian line-shape function $F(\nu)$. The Doppler line width γ_D is so defined that it is equal to the half width at half of the maximum (HWWM) of the line-shape function. The same way of notating is used for all other width parameters γ_{xy} below.

It can be said without any exaggeration that the pressure broadening effect is the most complicated one among the others and still represents a complex theoretical task to be tackled, and is in the same time of major experimental importance. So far, there is no way to calculate analytically from the basics the profile of a pressure, or collisional, line through a single approach near the line center as well as in the far wing region. The various approximations, which are therefore used, are immanently limited to the certain line regions they deal with. The most popular among these approximations is the *impact approximation* which postulates that the duration of the collisions of the gas particles is very small compared to the average time between the collisions. Lorentz was the first to achieve a result exploiting this approach, the Lorentz line-shape function:

$$F_L(\nu) = \frac{\gamma_L}{\pi} \frac{1}{(\nu - \nu_0)^2 + \gamma_L^2} \tag{3.11}$$

where γ_L is the Lorentz line width, *Thorne et al.* [1999]. As one can see, the result Eq. 3.11 is pretty similar to Eq. 3.7 but the specific line parameters γ and γ_L make them differ significantly in the corresponding frequency regions of interest. For atmospheric pressures γ_L is much greater and because of that, of experimental significance in contrast to γ.
Elaborating the model of Lorentz, van Vleck and Weisskopf made a correction to it, *Van Vleck and Weisskopf* [1945],particularly for the microwave region:

$$F_{VVW}(\nu) = \left(\frac{\nu}{\nu_0}\right)^2 \frac{\gamma_L}{\pi} \left[\frac{1}{(\nu - \nu_0)^2 + \gamma_L^2} + \frac{1}{(\nu + \nu_0)^2 + \gamma_L^2}\right] \tag{3.12}$$

which can be reduced to a Lorentzian for $(\nu - \nu_0) << \nu_0$ and $0 << \nu_0$. Except for the additional factor $(\nu/\nu_0)^2$, F_{VVW} can be regarded as the sum of two F_L, or respectively lines, one with its center frequency at ν_0, the other at $-\nu_o$.

The van Vleck and Huber lineshape [*Van Vleck and Huber*, 1977] is similar to Eq. 3.12, except for the factor $(\nu/\nu_0)^2$ which is replaced by $(\nu * \tanh(h * \nu/(2kT)))/(\nu_0 * \tanh(h * \nu_0/(2kT)))$, with k the Boltzmann constant, h the Planck constant, and T the atmospheric temperature (the denominator is actually a consequence of the line strength definition in the spectroscopic catalogs). The lineshape Eq. 3.12 with this factor can be used for the entire frequency range, since the microwave approximation: $\tanh(x) = x$, that leads to the factor $(\nu/\nu_0)^2$, is not made.

The combined picture of a simultaneously Doppler and pressure broadened line is the next step of the approximations development. The line-shape function has to approximated in this case by the Voigt line-shape function

$$F_{Voigt}(\nu, \nu_0) = \int F_L(\nu, \nu')\; F_D(\nu', \nu_0)\; d\nu' \tag{3.13}$$

though there's no strict justification for its use - the two processes are assumed to act independently, which in reality is not the fact. Regardless of this flaw, it is the only way up to date to model the combination of the broadening processes. The integral in Eq. 3.13 can not be computed analytically, so certain approximation algorithms must be used.

Another possibility would be the combination of the last two equations Eq. 3.12 and Eq. 3.13. The respective result then will be

$$F_S = \left(\frac{\nu}{\nu_0}\right)^2 \; [F_{Voigt}(\nu, \nu_0) + F_{Voigt}(\nu, -\nu_0)] \tag{3.14}$$

The advantage of such a model is that it behaves like a van Vleck-Weisskopf line-shape function in the high pressure limit and like a Voigt one in the low pressure limit. There is one important caveat to the equation Eq. 3.14: it has to be made sure that the algorithm that is used to compute the Voigt function really produces a Lorentz line in the high pressure limit. Another point of significance is the demand that the model yields meaningful results far from the line center, since the line center from the "mirror" line at -ν_0 is situated approximately $2\nu_0$ away from the frequency ν_0 of computation. The algorithm of *Drayson* [1976] and *Oliveiro and Longbothum* [1977] was explicitly checked to satisfy both requirements, while this was found to be not true for some other algorithms, commonly used for Voigt-shape computation. In particular, the it is not true for the Hui-Armstrong-Wray Formula, as defined in *Hui et al.* [1978] and in Equation 2.60 of *Rosenkranz* [1993]. So, provided the stated above is fulfilled, the F_S line shape gives a smooth transition from high tropospheric pressures to low stratospheric ones, and should be valid near the line centers throughout the microwave region.

3.1.2 Partition Functions

The treatment of the partition functions is directly related to the molecular energy states and their statistical distribution during the radiation process.

In any case of spectroscopic interest the free molecules of a gas are not optically thick at all frequencies, so the radiation energy is not represented by blackbody radiation. The most common assumption made, which is sufficient in the case of tropospheric and low stratospheric research, is the *local thermodynamic equilibrium* or *LTE* . According to it, it's possible to find a common temperature, which may vary from place to place, that fits the Boltzmann energy population distribution and the Maxwell velocities distribution. This practically means, that under LTE the collisional processes must be of greater importance than radiative ones. In other words, an excited state must have a higher probability of de-excitation by collision than by spontaneous radiation. This is the important factor which makes natural broadening differ quantitatively so much from the pressure (collisional) one, though both are described qualitatively almost identically by Lorentzian line- shape functions.

According to the Maxwell-Boltzmann distribution law, in LTE the total number of gas particles N_n in a state E_n is given by

$$N_n = N_0 \frac{g_n}{g_0} e^{-E_n/kT} \tag{3.15}$$

where N_0 is particle number in the ground state, and g_n, g_o are the statistical weights (degeneracies) of the $n-$state and the ground state, *Gordy and Cook* [1970]. Thus the total particle number N is given by

$$N = \frac{N_0}{g_0} \sum_{n=0}^{\infty} g_n \, e^{-E_n/kT} = \frac{N_0}{g_0} \, Q(T) \tag{3.16}$$

The quantity $Q(T)$ is the *partition function* of the gas, which generally speaking describes the energy states distribution of the gas particles.

The partition function for a perfect gas molecule can be represented by the product of the *translational* and the *internal* partition functions, as defined in *Herzberg* [1945],

$$Q = Q_{tr} \, Q_{int} \tag{3.17}$$

bearing in mind that the respective energies, translational and internal, are independent of each other. The first quantity Q_{tr} accounts for the distribution of the translational energy of the gas particles - it takes into account that the translational velocities of the particles fulfill the Maxwell distribution. The quantity, however, which we are interested in in (3.3) is the *internal* partition function (or the *total internal partition function* because the transitions between the discrete internal energy states are responsible for the absorption or emittance of radiation. Accordingly Q_{int} describes the distribution of energy among the internal energy states of the gas particles.

The internal partition function for free gaseous molecules is a function of the electronic, the vibrational, the rotational, and the nuclear spin states. An approximation is used in *Gordy and Cook* [1970] in order to display the individual contribution explicitly

$$Q_{int} = Q_e \, Q_v \, Q_r \, Q_n \tag{3.18}$$

and thus the interaction between these various states is neglected. For practically all polyatomic molecules the excited electronic states are entirely negligible to those of the ground states, i.e. $Q_e = 1$. Only for the very few polyatomic molecules with a multiplet ground state (NO_2 , ClO_2 , and free radicals) has the electronic contribution to be considered.
If we neglect the anharmonicities, the vibrational partition function, with vibrational energy levels measured with respect to the ground state for the *harmonic oscillator*, is acording *Herzberg* [1945]

$$Q_v = \left(\sum_{\nu_1} e^{-\nu_1 h\omega_1/kT} \right) \left(\sum_{\nu_2} e^{-\nu_2 h\omega_2/kT} \right) ... \tag{3.19}$$

where ν_1, ν_2,..., the vibrational quantum numbers, can each have the values 0,1,2,... and ω_1, ω_2,..are the frequencies of the fundamental modes of vibration. The summation is taken over all values of ν_1, ν_2,..., and each fundamental mode is counted separately. This result is valid for non-degenerate vibrations. If we use the simple expression for geometric progression

$$\sum_{\nu_i} e^{-\nu_i h\omega_i/kT} = \frac{1}{1 - e^{h\omega_i/kT}} \tag{3.20}$$

and the degeneracies d_1, d_2,... of the fundamental modes, we get finally for the vibrational partition function

$$Q_v = \left(1 - e^{h\omega_1/kT}\right)^{-d_1} \left(1 - e^{h\omega_2/kT}\right)^{-d_2} ... \tag{3.21}$$

The rotational partition function looks differently for the different symmetry types of molecules. For diatomic and linear polyatomic molecules with no center of symmetry the corresponding expression is, as defined in *Gordy and Cook* [1970]

$$\begin{aligned} Q_r &= \sum_{J=0}^{\infty} (2J+1) e^{-hBJ(J+1)/kT} \\ &= \frac{kT}{hB} + \frac{1}{3} + \frac{1}{15}\frac{hB}{kT} + \frac{4}{315}\left(\frac{hB}{kT}\right)^2 + ... \\ &\cong \frac{kT}{hB} \end{aligned} \tag{3.22}$$

For *rigid symmetric-*, *asymmetric-*, and *spherical* top molecules there are also other factors to be taken into consideration, such as the spatial structure of the molecules, nuclear spin, inversion and internal rotation. The general expression in the case of a *rigid symmetric-* top molecule according *Herzberg* [1945] is

$$Q_r = \frac{1}{\sigma} \sum_{J=0}^{\infty} \sum_{K=-J}^{J} (2J+1)\, e^{-h[BJ(J+1)+(A-B)K^2]/kT} \tag{3.23}$$

where σ is measure of the degree of symmetry. The usual symmetric top has C_3 or $C_{3\nu}$ symmetry, therefore σ = 3. To a good approximation, the summation above can expressed as in *Gordy and Cook* [1970]

$$Q_r = \frac{1}{\sigma} \left[\left(\frac{\pi}{B^2 A} \right) \left(\frac{kT}{h} \right)^3 \right]^{1/2} = \frac{5.34 \times 10^6}{\sigma} \left(\frac{T^3}{B^2 A} \right)^{1/2} \tag{3.24}$$

For an *asymmetric* top the formula would then be

$$Q_r = \frac{5.34 \times 10^6}{\sigma} \left(\frac{T^3}{ABC} \right)^{1/2} \tag{3.25}$$

and for a *spherical* top, using the current notation of *Gordy and Cook* [1970] in the respective expression in *Herzberg* [1945],

$$Q_r = \frac{5.34 \times 10^6}{\sigma} \left(\frac{T^3}{A^3} \right)^{1/2} \tag{3.26}$$

3.1.3 Line Catalogs

There are several spectroscopic catalogs implemented in ARTS: HITRAN, JPL, MYTRAN, and the ARTS inherent catalog format.

Special attention will be paid here only to the inherent format in ARTS. To keep track with the changes in the catalog format, every catalog file must start with $ARTSCAT - x$, where for current version $x = 2$. Files with different or missing version will be rejected. The current version is stored in the private member variable mversion. It can be read with the member function Version, which returns a String 'ARTSCAT-x'. After the version tag (ARTSCAT-x), ARTS outputs the number of lines when catalog files are written. This number is not used by reading routines, though.

The line catalog should not have any fixed column widths because the precision of the parameters should not be limited by the format. The catalog can then be stored in principle as binary or ASCII, though currently are implemented only ASCII files. In the ASCII version the columns are separated by one or more blanks. The line format is then specified by only the order and the units of the columns. As the catalog entry for each transition can be quite long, it can be broken across lines in the ASCII file. Each new transition is marked with an '@' character. The first column will contain the species and isotope, following the naming scheme described below. Scientific notation is allowed, e.g. 501.12345e9. Note that starting with ARTSCAT-2, the intensity is per molecule, i.e., it does not contain the isotopic ratio. This is similar to JPL, but different to HITRAN. The line format is:

```
Col  Variable                        Label    Unit
-----------------------------------------------
 0   `@'                             ENTRY       -
 1   name                             NAME       -
 2   center frequency                    F      Hz
 3   pressure shift of F               PSF   Hz/Pa
 4   line intensity                     I0  m^2/Hz
 5   reference temp. for I0           T_I0       K
 6   lower state energy               ELOW       J
 7   air broadened width              AGAM   Hz/Pa
 8   self broadened width             SGAM   Hz/Pa
 9   AGAM temp. exponent              NAIR       -
10   SGAM temp. exponent             NSELF       -
11   ref. temp. for AGAM, SGAM       T_GAM       K
12   number of aux. parameters       N_AUX       -
13   auxiliary parameter              AUX1       -
14   ...
15   error for F                        DF      Hz
16   error for I0                      DI0       %
17   error for AGAM                  DAGAM       %
18   error for SGAM                  DSGAM       %
19   error for NAIR                  DNAIR       %
20   error for NSELF                DNSELF       %
21   error for PSF                    DPSF       %
```

The parameters 0-12 must be present, the others can be missing, since they are not needed for the calculation. For the error fields (15-21), a -1 means that no value exists.
Thus a valid ARTS line file would be:

```
ARTSCAT-2 2
@ CH4-211 1011349857.063 0 2.96070344144819e-27 296
2183.6851 13314.2468393782 21302.7949430052 0.75 0.75 296 0
@ O3-666 1088246622.54 0 2.82913939200384e-22 296
522.5576 21361.9693734024 27723.2206411054 0.76 0.76 296 0
```

Some species need special parameters that are not needed by other species (for example overlap coefficients for O_2). In the case of oxygen two parameters are sufficient to describe the overlap, but other species, e.g., methane, may need more coefficients. The default for N_AUX is zero. In that case, no further AUX fields are present.

3.1.4 Species specific data

The following part treats the species related data in ARTS - all currently implemented species with the respective molecular masses, isotopic ratios, partition functions, and the sources for this information.

Table 3.1 lists the implemented species in ARTS. The first row gives the ARTS molecule name, the second the ARTS isotope name (these two identify the species within ARTS). The third row gives the number of this species in the MYTRAN catalog, the fourth the one used in HITRAN96, and the fifth the corresponding tag numbers of the JPL00 catalog.

ARTS Name	*ARTS Isotope*	*MYTRAN Tag*	*HITRAN Tag*	*JPL00 Tag*
H2O	161	11	11	18003, 18005
	181	12	12	20003
	171	13	13	19003
	162	14	14	19002
	182	-1	15	21001
	172	-1	16	
	262	-1	-1	20001
	SelfContStandardType	-1	-1	
	ForeignContStandardType	-1	-1	
	ForeignContMaTippingType	-1	-1	
	ContMPM93	-1	-1	
	SelfContCKD24	-1	-1	
	ForeignContCKD24	-1	-1	
	ForeignContATM01	-1	-1	
	CP98	-1	-1	
	MPM87	-1	-1	
	MPM89	-1	-1	
	MPM93	-1	-1	
	PWR98	-1	-1	
CO2	626	21	21	
	636	22	22	
	628	23	23	46013
	627	24	24	45012
	638	25	25	
	637	26	26	
	828	27	27	
	728	28	28	
	SelfContPWR93	-1	-1	
	ForeignContPWR93	-1	-1	
O3	666	31	31	48004, 48005, 48006, 48007, 48008
	668	32	32	50004, 50006
	686	33	33	50003, 50005
	667	34	34	49002
	676	35	35	49001
N2O	446	41	41	44004, 44009, 44012

Table 3.1: (continued)

ARTS Name	*ARTS Isotope*	*MYTRAN Tag*	*HITRAN Tag*	*JPL00 Tag*
	456	42	42	45007
	546	43	43	45008
	448	44	44	46007
	447	-1	45	
CO	26	51	51	28001
	36	52	52	29001
	28	53	53	30001
	27	-1	54	29006
	38	-1	55	
	37	-1	56	
CH4	211	-1	61	
	311	-1	62	
	212	-1	63	17003
O2	66	71	71	32001, 32002
	68	72	72	34001
	67	73	73	33002
	SelfContStandardType	-1	-1	
	SelfContMPM93	-1	-1	
	SelfContPWR93	-1	-1	
	PWR98	-1	-1	
	PWR93	-1	-1	
	PWR88	-1	-1	
	MPM93	-1	-1	
	MPM92	-1	-1	
	MPM89	-1	-1	
	MPM85	-1	-1	
NO	46	81	81	30008
	56	-1	82	
	48	-1	83	
SO2	626	91	91	64002, 64005
	646	92	92	66002
	636	93	-1	65001
	628	94	-1	66004
NO2	646	101	101	46006
NH3	4111	111	111	17002, 17004
	5111	112	112	18002
	4112	-1	-1	18004
HNO3	146	121	121	63001, 63002, 63003, 63004, 63005, 63006
OH	61	131	131	17001
	81	132	132	19001

Table 3.1: (continued)

ARTS Name	*ARTS Isotope*	*MYTRAN Tag*	*HITRAN Tag*	*JPL00 Tag*
	62	133	133	18001
HF	19	141	141	20002
	29	-1	-1	21002
HCl	15	151	151	36001
	17	152	152	38001
	25	-1	-1	37001
	27	-1	-1	39004
HBr	19	161	161	80001
	11	162	162	82001
HI	17	-1	171	
ClO	56	181	181	51002, 51003
	76	182	182	53002, 53006
OCS	622	191	191	60001
	624	192	192	62001
	632	193	193	61001
	822	194	194	62002
	623	195	195	
H2CO	1126	201	201	30004
	1136	202	202	31002
	1128	203	203	32004
	1226	-1	-1	31003
	2226	-1	-1	32006
HOCl	165	211	211	52006
	167	212	212	54005
N2	44	-1	221	
	SelfContMPM93	-1	-1	
	SelfContPWR93	-1	-1	
	SelfContStandardType	-1	-1	
	SelfContBorysow	-1	-1	
	DryContATM01	-1	-1	
HCN	124	231	231	27001, 27003
	134	232	232	28002
	125	233	233	28003
	224	-1	-1	28004
CH3Cl	215	241	241	50007
	217	242	242	52009
H2O2	1661	251	251	34004
C2H2	1221	-1	261	
	1231	-1	262	
C2H6	1221	-1	271	
PH3	1111	281	281	34003
COF2	269	291	291	66001

Table 3.1: (continued)

ARTS Name	*ARTS Isotope*	*MYTRAN Tag*	*HITRAN Tag*	*JPL00 Tag*
SF6	29	-1	301	
H2S	121	311	311	34002
	141	-1	312	
	131	-1	313	
	122	-1	-1	35001
HCOOH	1261	321	321	46005
	1361	-1	-1	47002
	2261	-1	-1	47003
	1262	-1	-1	47004
HO2	166	331	331	33001
O	6	341	341	16001
ClONO2	5646	351	351	97002
	7646	352	352	99001
NO+	46	-1	361	30011
OClO	656	431	-1	67001
	676	432	-1	69001
BrO	96	401	-1	95001
	16	402	-1	97001
H2SO4	126	481	-1	98001
Cl2O2	565	491	-1	102001
	765	492	-1	104001
HOBr	169	371	371	96001
	161	372	372	98002
C2H4	221	381	381	
	231	382	382	
CH3CN	211124	-1	-1	41001
	311124	-1	-1	42006
	211134	-1	-1	42007
	211125	-1	-1	42001
	211224	-1	-1	42008
HNC	142	-1	-1	27002
HNC	143	-1	-1	28005
HNC	152	-1	-1	28006
HNC	242	-1	-1	28007
liquidcloud	MPM93	-1	-1	
icecloud	MPM9	-1	-1	
rain	MPM93	-1	-1	

Table 3.1: Implemented species in ARTS

Table 3.2 gives an overview of the isotopic ratios and their sources, and the molecular mass of the species used in used in ARTS. The isotopic ratio of the species was in general taken from HITRAN00, even for species where JPL00 and HITRAN00 give isotopic ratios.

Species not present in HITRAN00 were extracted from JPL00. JPL00 isotopic ratios are normalized to 1 for some molecules, for these cases, the HITRAN00 isotopic ratio of the major isotope was used to scale the JPL00 isotopic ratio to relative number.

As in 3.1 the first two columns give the ARTS molecule name and the ARTS isotope name. The next two ones give the isotopic ratio and the source it is taken from respectively. An entry of 1 corresponds to HITRAN00, 2 to JPL00, 3 to JPL00 multiplied with the maximum abundance of this molecule in HITRAN. The last column contains the molecular masses of the isotopes. An entry of -1 is prescribed to the continuum tags.

ARTS Name	*ARTS Isotope*	*Isotopic Ratio*	*Isotopic Ratio Source*	*Mass*
H2O	161	0.99731702	1	18
	181	0.00199983	1	20
	171	0.00037200	1	19
	162	0.00031069	1	19
	182	6.23003E-07	1	21
	172	1.15853E-07	1	20
	262	2.2430204E-08	3	20
	SelfContStandardType	-1	-1	-1
	ForeignContStandardType	-1	-1	-1
	ForeignContMaTippingType	-1	-1	-1
	ContMPM93	-1	-1	-1
	SelfContCKD24	-1	-1	-1
	ForeignContCKD24	-1	-1	-1
	ForeignContATM01	-1	-1	-1
	CP98	-1	-1	-1
	MPM87	-1	-1	-1
	MPM89	-1	-1	-1
	MPM93	-1	-1	-1
	PWR98	-1	-1	-1
CO2	626	0.98420	1	44
	636	0.0110574	1	45
	628	0.00394707	1	46
	627	0.000733989	1	45
	638	0.000044346	1	47
	637	0.0000082462	1	46
	828	0.00000395734	1	48
	728	0.0000014718	1	47
	SelfContPWR93	-1	-1	-1
	ForeignContPWR93	-1	-1	-1
O3	666	0.992901	1	48
	668	0.00398194	1	50
	686	0.00199097	1	50
	667	0.000740	1	49
	676	0.000370	1	49

Table 3.2: (continued)

ARTS Name	*ARTS Isotope*	*Isotopic Ratio*	*Isotopic Ratio Source*	*Mass*
N2O	446	0.990333	1	44
	456	0.00364093	1	45
	546	0.00364093	1	45
	448	0.00198582	1	46
	447	0.000369	1	46
CO	26	0.986544	1	28
	36	0.0110836	1	29
	28	0.00197822	1	30
	27	0.00036867	1	29
	38	0.000022225	1	31
	37	0.0000041329	1	30
CH4	211	0.988274	1	16
	311	0.0111031	1	17
	212	0.000615751	1	17
O2	66	0.995262	1	32
	68	0.00399141	1	34
	67	0.00074235	1	33
	SelfContStandardType	-1	-1	-1
	SelfContMPM93	-1	-1	-1
	SelfContPWR93	-1	-1	-1
	PWR98	-1	-1	-1
	PWR93	-1	-1	-1
	PMR88	-1	-1	-1
	MPM93	-1	-1	-1
	MPM92	-1	-1	-1
	MPM89	-1	-1	-1
	MPM87	-1	-1	-1
	MPM85	-1	-1	-1
NO	46	0.993974	1	30
	56	0.00365431	1	31
	48	0.00199312	1	32
SO2	626	0.945678	1	64
	646	0.0419503	1	66
	636	0.0074989421	2	65
	628	0.0020417379	2	66
NO2	646	0.991616	1	46
NH3	4111	0.9958715	1	17
	5111	0.00366129	1	18
	4112	0.00044792294	3	18
HNO3	146	0.989110	1	63
OH	61	0.997473	1	17
	81	0.00200014	1	19

Table 3.2: (continued)

ARTS Name	*ARTS Isotope*	*Isotopic Ratio*	*Isotopic Ratio Source*	*Mass*
	62	0.000155371	1	18
HF	19	0.99984425	1	20
	29	0.00014994513	3	21
HCl	15	0.757587	1	36
	17	0.242257	1	38
	25	0.00011324004	2	37
	27	3.6728230E-05	2	39
HBr	19	0.50678	1	80
	11	0.49306	1	82
HI	17	,0.99984425	2	128
ClO	56	0.755908	1	51
	76	0.24172	1	53
OCS	622	0.937395	1	60
	624	0.0415828	1	62
	632	0.0105315	1	61
	623	0.00739908	1	61
	822	0.0018797	1	61
H2CO	1126	0.986237	1	30
	1136	0.0110802	1	31
	1128	0.00197761	1	32
	1226	0.00029578940	3	31
	2226	2.2181076E-08	3	32
HOCl	165	0.75579	1	52
	167	0.241683	1	54
N2	44	0.9926874	1	28
	SelfContMPM93	-1	-1	-1
	SelfContPWR93	-1	-1	-1
	SelfContStandardType	-1	-1	-1
	SelfContBorysow	-1	-1	-1
	DryContATM01	-1	-1	-1
HCN	124	0.985114	1	27
	134	0.011076	1	28
	125	0.00362174	1	28
	224	0.00014773545	3	28
CH3Cl	215	0.748937	1	50
	217	0.239491	1	52
H2O2	1661	0.994952	1	34
C2H2	1221	0.977599	1	26
	1231	0.021966	1	27
C2H6	1221	0.97699	1	30
PH3	1111	0.99953283	1	34
COF2	269	0.986544	1	66

Table 3.2: (continued)

ARTS Name	*ARTS Isotope*	*Isotopic Ratio*	*Isotopic Ratio Source*	*Mass*
SF6	29	0.95018	1	146
H2S	121	0.949884	1	34
	141	0.0421369	1	36
	131	0.00749766	1	35
	122	0.00029991625	2	35
HCOOH	1261	0.983898	1	46
	1361	0.010913149	3	47
	2261	0.00014755369	3	47
	1262	0.00014755369	3	47
HO2	166	0.995107	1	33
O	6	0.997628	1	16
ClONO2	5646	0.74957	1	97
	7646	0.239694	1	99
NO+	46	0.993974	1	30
OClO	656	0.75509223	2	67
	676	0.24490632	2	69
BrO	96	0.50582466	2	95
	16	0.49431069	2	97
H2SO4	126	0.95060479	2	98
Cl2O2	565	0.57016427	2	102
	765	0.36982818	2	104
HOBr	169	0.505579	1	102
	161	0.491894	1	104
C2H4	221	0.977294	1	28
	231	0.0219595	1	29
CH3CN	211124	0.97366840	3	41
CH3CN	311124	0.011091748	3	42
CH3CN	211134	0.011091748	3	42
CH3CN	211125	0.0036982817	3	42
CH3CN	211224	0.00044977985	3	42
HNC	142	0.98505998	3	27
HNC	143	0.011091748	3	28
HNC	152	0.0036982817	3	28
HNC	242	0.00014996849	3	28
liquidcloud	-1	-1	-1	
icecloud	-1	-1	-1	
rain	-1	-1	-1	

Table 3.2: Isotopic ratios within ARTS, source of the ratios and molecular mass

Table 3.3 refers to the partition function data. It is calculated following HITRAN96, where the coefficients of a third order polynomial in temperature of the partition function are given. Mainly HITRAN96 coefficients are used, the coefficients of species not covered in HITRAN96, but present in ARTS, were generated by a polynomial fit to the partition function given in JPL00. The source of the partition function coefficients is given in the third column, where 1 indicates the HITRAN96 source, and 2 the JPL00 source.

Within the generation of partition function coefficients from the JPL00 catalog, a general comparison of the partition function ratio (the important quantity for the conversion of the catalog intensity to other temperatures) for species present in JPL00 and HITRAN96 was performed. The maximum error found between the JPL00 and HITRAN96 partition function ratios within the temperature range of 150 K to 300 K is given in the table when both catalogs cover the species. Otherwise, the quality of the polynomial fit is given as the maximum difference found between 150 K and 300 K for the original JPL00 partition function and the polynomial fit.

Errors of more than 10 % were found for some species when no correction for populated vibrational energy levels was applied. The correction for vibrational energy levels was performed with data mainly extracted from the *Chase et al.* [1985] compilation, except for O_2, CO, NH_3 , and ClO which were taken from *Rosenkranz* [1993]. The vibrational modes for the minor isotopes were taken from the main isotope when no other data was available. A few species still have errors of about 10 %, which could either be explained by incorrect/missing vibrational modes or differences in the partition functions of the two catalogs. All species with no vibrational information are marked as 'no info'.

The continuum tags for the respective species, listed out in the two tables above, are omitted in this case. The simple reason for this is because continua calculations do not need any partition functions.

The partition function polynomials (PF) themselves for the individual species are to be found in `/arts/src/partition_function_data.cc}`.

ARTS Name	*ARTS Isotope*	*PF Source*	*No Corr. %*	*Corr. %*	*Vibrational Modes* cm^{-1}
H2O	161	1	0.33	0.28	1594.7, 3651.1, 3755.9
	181	1	0.33	0.28	1594.7, 3651.1, 3755.9
	171	1	0.39	0.35	1594.7, 3651.1, 3755.9
	162	1	0.50	0.46	1594.7, 3651.1, 3755.9
	182	2	0.33	0.32	1594.7, 3651.1, 3755.9
	262	2	0.35	0.34	1594.7, 3651.1, 3755.9
CO2	626	1			no info
	636	1			no info
	628	1	8.76	4.40	667.3, 1384.9, 2349.3
	627	1	8.67	4.32	667.3, 1384.9, 2349.3
	638	1			no info
	637	1			no info
	828	1			no info
	728	1			no info
O3	666	1	1.30		705, 1043, 1110
	668	1	5.82	2.64	693.0
	686	1	5.88	2.49	678.0
	667	1	5.25	1.25	705, 1043, 1110

Table 3.3: (continued)

ARTS Name	*ARTS Isotope*	*PF Source*	*No Corr. %*	*Corr. %*	*Vibrational Modes* cm^{-1}
	676	1	5.29	1.28	705, 1043, 1110
N2O	446	1	12.71	0.89	588.8, 588.8, 1284.9, 2223.8
	456	1	13.33	1.26	588.8, 588.8, 1284.9, 2223.8
	546	1	12.83	0.95	588.8, 588.8, 1284.9, 2223.8
	448	1	12.11	0.82	588.8, 588.8, 1284.9, 2223.8
	447	1			588.8, 588.8, 1284.9, 2223.8
CO	26	1	0.03	0.03	2143.5
	36	1	0.01	0.01	2143.5
	28	1	0.01	0.01	2143.5
	27	1	0.02	0.02	2143.5
	38	1			2143.5
	37	1			2143.5
CH4	211	1			1306, 1306, 1306, 1534, 1534, 2917, 3019
	311	1			1306, 1306, 1306, 1534, 1534, 2917, 3019
	212	1	22.18	21.43	1306, 1306, 1306, 1534, 1534, 2917, 3019
O2	66	1	0.06	0.11	1556.5
	68	1	0.68	0.62	1556.5
	67	1	0.71	0.65	1556.5
NO	46	1	1.56	1.56	no info
	56	1			no info
	48	1			no info
SO2	626	1	9.28	1.26	517.7, 1151.4, 1361.8
	646	1	9.25	1.23	517.7, 1151.4, 1361.8
	636	2	0.35	1.32	517.7, 1151.4, 1361.8
	628	2	0.35	1.32	517.7, 1151.4, 1361.8
NO2	646	1	3.46	0.98	756.8, 1357.8, 1665.5
NH3	4111	1	2.07	1.09	950, 1629, 1629, 3335, 3414, 3414
	5111	1	22.22	21.12	950, 1629, 1629, 3335, 3414, 3414
	4112	2	0.52	0.78	950, 1629, 1629, 3335, 3414, 3414
HNO3	146	2	0.38	4.00	465, 583, 680, 765, 886, 1320, 1335, 1710, 3560
OH	61	1	0.85	0.85	no info
	81	1	0.84	0.84	no info
	62	1	1.04	1.04	no info
HF	19	1	0.02	0.02	no info
	29	2	0.04	0.04	no info
HCl	15	1	2.32	2.32	no info

Table 3.3: (continued)

ARTS Name	*ARTS Isotope*	*PF Source*	*No Corr. %*	*Corr. %*	*Vibrational Modes cm^{-1}*
	17	1	2.30	2.30	no info
	25	2	0.03	0.03	no info
	27	2	0.03	0.03	no info
HBr	19	1	1.89	1.89	no info
	11	1	1.89	1.89	no info
HI	17	1			no info
ClO	56	1	0.83	1.94	842.4
	76	1	0.81	1.96	842.4
OCS	622	1	19.14	1.49	524, 524, 859, 2064
	624	1	19.14	1.66	524, 524, 859, 2064
	632	1	20.48	2.43	524, 524, 859, 2064
	822	1	20.07	2.15	524, 524, 859, 2064
H2CO	1126	1	3.55	2.80	1163.5, 1247.4, 1500.6, 1746.1, 2766.4, 2843.4
	1136	1	3.94	3.18	1163.5, 1247.4, 1500.6, 1746.1, 2766.4, 2843.4
	1128	1	1.39	0.74	1163.5, 1247.4, 1500.6, 1746.1,2766.4, 2843.4
	1226	2	0.34	0.18	1163.5, 1247.4, 1500.6, 1746.1,2766.4, 2843.4
	2226	2	0.34	0.21	1163.5, 1247.4, 1500.6, 1746.1, 2766.4, 2843.4
HOCl	165	1	3.89	0.95	725, 1239.4, 3609.5
	167	1	3.89	0.95	725, 1239.4, 3609.5
N2	44	1			no info
HCN	124	1	6.84	0.51	713.5, 713.5, 2096.3, 3311.5
	134	1	7.05	0.38	713.5, 713.5, 2096.3, 3311.5
	125	1	7.13	0.45	713.5, 713.5, 2096.3, 3311.5
	224	2	1.42	1.72	713.5, 713.5, 2096.3, 3311.5
CH3Cl	215	1	5.86	1.65	732, 1017, 1017, 1355, 1455, 1455, 2968, 3054, 3054
	217	1	5.92	1.71	732, 1017, 1017, 1355, 1455, 1455, 2968, 3054, 3054
H2O2	1661	1	14.46	14.46	no info
C2H2	1221	1			no info
	1231	1			no info
C2H6	1221	1			no info
PH3	1111	1	3.70	2.06	992, 1122, 1122, 2323, 2328, 2328
COF2	269	1	16.72	13.92	381, 381, 1074, 1978
SF6	29	1			no info
H2S	121	1	0.60	0.29	1183,2615,2627
	141	1			no info

Table 3.3: (continued)

ARTS Name	*ARTS Isotope*	*PF Source*	*No Corr. %*	*Corr. %*	*Vibrational Modes cm^{-1}*
	131	1			no info
	122	2	0.40	0.32	1183, 2615, 2627
HCOOH	1261	1	12.54	12.54	no info
	1361	2	6.53	6.54	no info
	2261	2	0.55	0.55	no info
	1262	2	0.57	0.57	no info
HO2	166	1	1.10	1.10	no info
O	6	1			no info
ClONO2	5646	2	1.87	1.87	no info
	7646	2	1.85	1.85	no info
NO+	46	1	0.01	0.01	no info
OClO	656	2	1.55	1.09	447.4, 945.3, 1109
	676	2	0.75	1.13	447.4, 945.3, 1109
BrO	96	2	0.09	0.09	no info
	16	2	0.09	0.09	no info
H2SO4	126	2	0.39	0.39	no info
Cl2O2	565	2	0.29	0.29	no info

Table 3.3: Partition function (PF) within ARTS, comparison with HTRAN96 and JPL00 PFs, vibrational modes (Note: O3-666 in JPL00 includes vibrational modes).

3.1.5 ARTS Workspace Variables and Methods

Both expressions (3.1) and (3.3) are used for the line by line calculations of the absorption. In order to calculate it certain *workspace variables* and *methods* are used. It has to be remembered that the certain values given in the following examples are arbitrarily chosen and given only for a better illustration.

There are two alternative ways to calculate the absorption coefficients in ARTS.The first one is through the workspace method

```
absCalc{}
```

This method actually works through calling internally in ARTS two other workspace methods:

```
xsec_per_tgCalc{}
absCalcFromXsec{}
```

The first one calculates the cross section x per tag group, the second one - the absorption coefficients α from the cross sections and the volume mixing ratios *VMR* at altitude i through the expression $\alpha_i = x_i \times VMR_i$. The second alternative way of computing the absorption coefficients is by using the afore mentioned methods explicitly in the control file in combination with one another

```
xsec_per_tgCalc{}
absCalcFromXsec{}
```

Therefore we will pay special attention to the functioning, the input and output workspace variables of `absCalc`. This method calculates the total absorption coefficient as well as the ones per defined tag group. Thus the two output workspace variables are `abs` and `abs_per_tag`. The input workspace variables are :

`tgs` : defining the available tag groups for the calculation of the absorption coefficients;

`f_mono` : the monochromatic frequency grid [Hz];

`p_abs` : the pressure grid for the absorption coefficients [Pa];

`t_abs` : temperature associated with the pressures in `p_abs` [K];

`n2_abs` : the total nitrogen profile associated with the pressures in `p_abs` [-];

`h2o_abs` : the total water profile associated with the pressures in `p_abs` [-];

`vmrs` : the VMRs (unit: absolute number) on the `p_abs` grid;

`lines_per_tg` : a list of spectral line data for each tag;

`lineshape` : lineshape specification: function, norm, cutoff;

`cont_description_names` : a list of names of continuum models;

`cont_description_parameters` : continuum model parameters.

All these input workspace variables have to be defined and set by the respective workspace methods for the proper functioning of `absCalc`. Any exception of this will lead to an error message.
First, the species of interest should be determined. This is done by the workspace method:

```
tgsDefine {
     [ "O3",
       "ClO",
       "N2",
       "H2O" ]
}
```

which produces as output `tgs`. In our case case this will create a list of four tag groups. For each of them the absorption coefficients will be calculated later on. The monochromatic grid is created through the method

```
VectorNLinSpace(f_mono){
        start = 1e9
        stop  = 200e9
        n     = 200
}
```

thus producing `f_mono` . The grid can be optionally saved into a file through `VectorWriteAscii (f_mono) {""}` . The pressure grid is created through the method

```
VectorNLogSpace(p_abs){
        start = 100000
        stop  = 10
        n     = 140
}
```

with output p_abs. Again, it can be optionally written into a file with

```
VectorWriteAscii(p_abs){""}
```

The corresponding temperatures to the pressure as well as the VMRs is realized through loading the respective input profiles, defined by the workspace variables raw_ptz and raw_vmrs. The corresponding methods are

```
MatrixReadAscii (raw_ptz)
{"/arts-data/atmosphere/fascod/midlatitude-summer.tz.aa"}
```

and

```
raw_vmrsReadFromScenario
{"/arts-data/atmosphere/fascod/midlatitude-summer"}
```

where the profiles are read from the atmospheric scenarios, in this case the one midlatitude-summer. Both physical profiles of H_2O and N_2 are loaded through similar workspace methods

```
h2o_absSet{}
n2_absSet{}
```

needed for the input variables h2o_abs and n2_abs. Though they are at the bottom of the above given list of input workspace variables, it is better to treat the continuum variables at this place. In our case of line by line calculations we are not interested in the continuum description, that's why we should just call a workspace method to assign empty arrays to the variables:

```
cont_descriptionInit{}
```

The next variable to be set is lines_per_tg , done by the method

```
lines_per_tgReadFromCatalogues{
  filenames = ["/arts-data/spectroscopy/jpl00/jpl00.cat",
    "/arts-dat/spectroscopy/hitran96/hitran96_lowfreq.par"]
   formats = [ "JPL", "HITRAN96" ]
   fmin    = [ 0,         0      ]
   fmax    = [ 200e9,     200e9  ]
 }
```

where lines from individual catalogs (here JPL and HITRAN96) are assigned to the different defined tag groups. Another possibility of doing this is to call two other methods one after another:

```
linesReadFromHitran {
filename = "/arts-data/spectroscopy/hitran96/hitran96_lowfreq.par"
fmin      = 1e9
fmax      = 200e9
}
```

and thus creating the variable `lines` , used afterward as input in

```
lines_per_tgCreateFromLines{}
```

in order to get finally `lines_per_tg` defined. Here is the proper place to give the individual methods for reading the available spectroscopic catalogs in ARTS, i.e. creating the variable `lines` from them. The ARTS own catalog format is read by

```
linesReadfromArts {
   filename = "ozone.al"
   fmin     = 0
   fmax     = 1e12
}
```

A similar procedure is done for the other catalogs:

```
linesReadfromHitran {
   filename = "hitran96_lowfreq.par"
   fmin     = 0
   fmax     = 1e12
}
```

for HITRAN96–01,

```
linesReadfromHitran2004 {
   filename = "hitran04.par"
   fmin     = 0
   fmax     = 1e12
}
```

for HITRAN04,

```
linesReadfromJpl {
   filename = "jpl100.cat"
   fmin     = 0
   fmax     = 1e12
}
```

for JPL, and

```
linesReadfromMytran {
   filename = "mytran98.my"
   fmin     = 0
   fmax     = 1e12
}
```

for MYTRAN.
The last input variable for `absCalc` to be discussed is `lineshape` . It is created either generally for all tag groups through

```
lineshapeDefine{
  shape = "Voigt_Kuntz6"
  normalizationfactor = "linear"
  cutoff = -1
}
```

or to set it individually for each tag group

```
lineshape_per_tgDefine{
  shape = ["Voigt_Kuntz6","no_shape","Voigt_Kuntz6",]
  normalizationfactor = ["quadratic","no_norm","linear",]
  cutoff = [-1,-1,-1]
}
```

More elucidation on the three parameters `shape` , `normalizationfactor` and `cutoff` is needed. The first one accounts for the different lineshape models now available in ARTS. It's done by setting `shape` to a given model name :

`Lorenz` : Lorenz line shape;

`Doppler` : Doppler line shape

`Voigt_Kuntz6` : Kuntz approximation to the Voigt line shape with relative accuracy better than 2×10^{-6};

`Voigt_Kuntz4` : Kuntz approximation to the Voigt line shape with relative accuracy better than 2×10^{-4};

`Voigt_Kuntz3` : Kuntz approximation to the Voigt line shape with relative accuracy better than 2×10^{-3};

`Voigt_Drayson` : Drayson approximation to the Voigt profile;

`Rosenkranz_Voigt_Kuntz6` : Rosenkranz oxygen absorption including overlap correction, at high altitudes Kuntz routine, requires the ARTS catalog with auxiliary overlap parameters; profile;

`Rosenkranz_Voigt_Drayson` : Rosenkranz oxygen absorption including overlap correction, at high altitudes Drayson routine, requires the ARTS catalog with auxiliary overlap parameters

The `normalizationfactor` sets the different normalization factors of the line shape:

`no_norm` : 1 (unity);

`linear` : (ν/ν_0);

`quadratic` : $(\nu/\nu_0)^2$;

`VVH` : $(\nu * \tanh(h * \nu/(2kT)))/(\nu_0 * \tanh(h * \nu_0/(2kT)))$

The `cutoff` can be set to -1 for no cutoff or to a positive number for a cutoff at a given frequency in [Hz], e.g. 650e9.

Thus we have generated all input workspace variables for `absCalc`. Therefore this part of the subsection is a little bit short of being able to serve as a control file in ARTS for absorption, provided that just one method is used where more alternatives are available and superfluous methods are skipped. The only addition to make it complete is to use after all input variable generating methods the

```
absCalc{}
```

method, the result of which can be saved into files through

```
MatrixWriteAscii (abs) {""}
```

or

```
ArrayOfMatrixWriteAscii (abs_per_tg) {""}
```

There is also an alternative version of `absCalc`: `absCalcSaveMemory`, which does not calculate `abs_per_tg` (only `abs` is calculated), thus is not suited for the calculation of weighting functions. But `absCalcSaveMemory` is able to handle larger absorption jobs than `absCalc` if the jobs is close to the machine specific memory limitations.

3.2 Continuum Absorption

As pointed out above, some molecules show beside the resonant line absorption also non-resonant continuum absorption. The main qualitative difference is the smooth dependence on frequency of the non-resonant absorption part in contrast to the resonant absorption part who shows strong local maxima and minima.

The implemented continuum absorption modules are connected with water vapor (H_2O), oxygen (O_2), nitrogen (N_2), and carbon dioxide (CO_2). Since these molecules have various permanent electric or magnetic multipoles, the physical explanations for the continuum absorption is different for each of these molecules.

Water Vapor has a strong electric dipole moment and posses therefore a wealth of rotational transitions in the microwave up to the submillimeter range. One explanation for the H_2O- continuum absorption is the inadequate formulation of the far wings of a spectral line, since the usually employed *Van Vleck and Weisskopf* [1945] line shape is according to its derivation only valid in the near wing zone. Other explanations are (see *Rosenkranz* [1993] for details) far wing contribution from far-infrared water vapor lines, collision induced absorption (CIA), and water polymer absorption. At present one can not definitively decide which of these possibilities is the correct one, probably all of them play a more or less important role, depending on the frequency range.

oxygen is one of the rare molecules in the Earth's atmosphere where a permanent magnetic dipole moment is present. The aligned spins of the two valence electrons gives a $^3\Sigma$ ground state of molecular oxygen. Due to the selection rules for magnetic dipole transitions, transitions with resonance frequency equal to zero are allowed. Such transitions have a characteristic Debye line shape function.

The homonuclear nitrogen molecule has in lowest order an electric quadrupole moment of modest magnitude. For the frequency range below 1 THz the collision induced rotation absorption band *Goody and Yung* [1989] is of most importance. The band center is around 3 THz and at 1 THz the band strength is approximately 1/6 of the maximum value (see Figure 5.2 of *Goody and Yung* [1989]). The electric field of the quadrupole moment of one molecule induces a dipole moment in the second molecule. This allows rotational transitions according to the electric quadrupole selection rules, $|\Delta J| = 0{,}2$ (see *Rosenkranz* [1993] for details). In a similar way exhibits carbon dioxide a collision induced absorption band (maximum around 1.5 THz, Figure 5.10 of *Goody and Yung* [1989]). Characteristic for collision induced absorption is the dependency on the square of the molecular density.

3.2.1 Water Vapor Continuum Models

As shown by *Liebe and Layton* [1987], *Rosenkranz* [1998], and *Ma and Tipping* [1990], the water vapor continuum absorption can be well described by

$$\alpha_{\mathrm{c}} = \nu^2 \cdot \Theta^3 \cdot (C^{\mathrm{o}}_{\mathrm{H_2O}} \cdot P^2_{\mathrm{H2O}} \cdot \Theta^{\mathrm{ns}} + C^{\mathrm{o}}_{\mathrm{d}} \cdot P_{\mathrm{H2O}} \cdot P_{\mathrm{d}} \cdot \Theta^{\mathrm{n}}\mathrm{f}) \tag{3.27}$$

where the microwave approximation ($h\nu \ll k_B T$) of the radiation field term is already applied. The adjustment of Eq. 3.27 to the data is performed through the parameter set $C^{\mathrm{o}}_{\mathrm{H_2O}}$, $\mathrm{n_s}$, $C^{\mathrm{o}}_{\mathrm{d}}$, and $\mathrm{n_f}$. Table 3.4 gives some commonly used continuum parameter sets.

model	$C^{\mathrm{o}}_{\mathrm{H_2O}}$ $\left[\frac{\mathrm{dB/km}}{\mathrm{hPa^2\ GHz^2}}\right]$	$\mathrm{n_s}$ [1]	$C^{\mathrm{o}}_{\mathrm{d}}$ $\left[\frac{\mathrm{dB/km}}{\mathrm{hPa^2\ GHz^2}}\right]$	$\mathrm{n_d}$ [1]	ref.
MPM87	$6.50 \cdot 10^{-8}$	7.5	$0.206 \cdot 10^{-8}$	0.0	*Liebe and Layton* [1987]
MPM89	$6.50 \cdot 10^{-8}$	7.3	$0.206 \cdot 10^{-8}$	0.0	*Liebe* [1989]
CP98	$8.04 \cdot 10^{-8}$	7.5	$0.254 \cdot 10^{-8}$	0.0	*Cruz Pol et al.* [1998]
PWR98	$7.80 \cdot 10^{-8}$	4.5	$0.236 \cdot 10^{-8}$	0.0	*Rosenkranz* [1998]
MPM93*	$7.73 \cdot 10^{-8}$	4.55	$0.253 \cdot 10^{-8}$	1.55	*Liebe et al.* [1993]

Table 3.4: Values of commonly used continuum parameter sets. The last line (MPM93*) represents an approximation of the pseudo-line continuum of MPM93 in the form of Eq. 3.27.

The MPM93 Continuum Parameterization

In the MPM93 model [*Liebe et al.*, 1993], the water vapor continuum is treated as a pseudo-line located in the far infrared around 2 THz. The pseudo-line continuum has therefore not four but seven parameters, the pseudo-line center frequency (ν^*) and the six pseudo-line parameters ($\mathrm{b}_1^*,\cdots,\mathrm{b}_6^*$):

$$\alpha_{\mathrm{c}}^{\mathrm{MPM93}} = 0.1820 \cdot \frac{\mathrm{b}_1^*}{\nu^*} \cdot P_{\mathrm{H2O}} \cdot \Theta^{3.5} \cdot \exp\left(\mathrm{b}_2^* \cdot (1-\Theta)\right) \cdot \nu^2 \cdot F_c(\nu,\nu_{\mathrm{k}}) \tag{3.28}$$

$$F_c(\nu,\nu_{\mathrm{k}}) = \left[\frac{\gamma_{\mathrm{c}}}{(\nu^*+\nu)^2+\gamma_{\mathrm{c}}^2} + \frac{\gamma_{\mathrm{c}}}{(\nu^*-\nu)^2+\gamma_{\mathrm{c}}^2}\right] \tag{3.29}$$

$$\gamma_{\mathrm{c}} = \mathrm{b}_3^* \cdot \left(\mathrm{b}_4^* \cdot \mathrm{P_{H2O}} \cdot \Theta^{\mathrm{b}_6^*} + \mathrm{P_d} \cdot \Theta^{\mathrm{b}_5^*}\right) \tag{3.30}$$

Table 3.5 lists the values of this continuum parameter set. It is remarkable that all these parameters are much larger compared to the physical water vapor line parameters of the same model. The only exception is b_2^*, the parameter which governs the exponential temperature behavior of the line strength. The magnitude of the pseudo-line width is shown for four

ν^*	b_1^*	b_2^*	b_3^*	b_4^*	b_5^*	b_6^*
[GHz]	$[\frac{\mathrm{kHz}}{\mathrm{hPa}}]$	[1]	$[\frac{\mathrm{MHz}}{\mathrm{hPa}}]$	[1]	[1]	[1]
1780.000	2230.000	0.952	17.620	30.50	2.00	5.00

Table 3.5: List of the MPM93 pseudo-line water vapor continuum parameters.

different cases in Table 3.6.

	contribution		total
	H_2O–H_2O	H_2O–air	
$\gamma_\mathrm{c}(200\,\mathrm{K})$	40.8 GHz	80.4 GHz	121.2 GHz
$\gamma_\mathrm{c}(300\,\mathrm{K})$	5.4 GHz	23.0 GHz	28.4 GHz

Table 3.6: Magnitude of the line width of the pseudo-line of the continuum term in MPM93. Assumed is a total pressure of 1000 hPa and a water vapor partial pressure of 10 hPa.

This change of continuum parameterization makes it difficult to compare MPM93 with the models which use Eq. (3.27). However, with respect to microwave frequencies, the line shape function, $F_c(\nu)$, can be approximated since the magnitude of the pseudo-line width is much smaller compared to the distance between microwave frequencies and ν^*, as shown for four different cases in Table 3.6:

$$F_c(\nu, \nu_\mathrm{k}) \approx 2 \cdot \frac{\gamma_\mathrm{c}}{\nu_\mathrm{c}^2} \tag{3.31}$$

Inserting Eq. (3.31) into Eq. (3.28) gives a quadratic frequency dependence of the MPM93 continuum, similar to the continuum parameterization expressed in Eq. (3.27). By additionally approximating the temperature dependence to the simple form

$$\begin{aligned}
\mathrm{n_s} \cdot \ln(\Theta) &= \ln\left(\Theta^{3.5} \cdot e^{\mathrm{b}_2^* \cdot (1-\Theta)}\right) \\
\mathrm{n_s} &= 3.5 + \mathrm{b}_2^* \cdot \frac{1-\Theta}{\ln(\Theta)} \\
\mathrm{n_s} &\approx 3.5 - \mathrm{b}_2^* = 2.55 \qquad \text{with} \quad \ln(\Theta) \approx (\Theta - 1)
\end{aligned} \tag{3.32}$$

one can rearrange the pseudo-line continuum to fit Eq. (3.27) (denoted by MPM93*). The so deduced continuum parameter set is given in Table 3.4.
The MPM93* continuum parameters $C^\mathrm{o}_{\mathrm{H_2O}}$ and C^o_d are 20 % and 15 % larger, respectively, than in the case of MPM87/MPM89. Large discrepancies exist for the temperature exponents $\mathrm{n_s}$ and $\mathrm{n_d}$ between MPM93* and earlier model versions. The exponent $\mathrm{n_s}$ is in MPM93* only 60 % of the corresponding value in MPM89 and the temperature dependence of the H_2O-air term is significant larger than for earlier MPM versions. This reduction of

n_s is mainly due to additional measurements considered in MPM93 (Refs. *Becker and Autler* [1946]; *Godon et al.* [1992]), while the continuum parameters in MPM87/MPM89 are determined by a single laboratory measurement at 138 GHz.

3.2.2 Oxygen Continuum Absorption

As pointed out by *Van Vleck* [1987], the standard theory for non-resonant absorption is that of Debye (see also *Townes and Schawlow* [1955]). The Debye line shape is obtained from the VVW line shape function by the limiting case $\nu_k \rightarrow 0$. Both, *Liebe et al.* [1993] and *Rosenkranz* [1993] adopted the Debye theory for their models. The only difference is the formulation of the line broadening, where the influence of water vapor is treated slightly different:

$$\alpha_c = C \cdot P_d \cdot \Theta^2 \cdot \frac{\nu^2 \cdot \gamma}{\nu^2 + \gamma^2} \tag{3.33}$$

$$\gamma = w \cdot (P_d \cdot \Theta^{0.8} + 1.1 \cdot P_{H2O} \cdot \Theta) \quad : \text{Rosenkranz} \tag{3.34}$$

$$\gamma = w \cdot P_{tot} \cdot \Theta^{0.8} \quad : \text{MPM93} \tag{3.35}$$

where P_d denotes the dry air partial pressure ($P_d = P_{tot} - P_{H2O}$). The value for the strength is $C = 2.56{\cdot}10^{-20}$ 1/(m Pa Hz) in the case of the Rosenkranz model and $C = 2.57{\cdot}10^{-20}$ 1/(m Pa Hz) in the case of the MPM93 model. The MPM93 value for C is therefore about 0.4 % larger than in the Rosenkranz model. Since the volume mixing ratio of oxygen in dry air is constant in the lower Earth atmosphere (0.20946 [*Goody*, 1995]), both models incorporate the oxygen VMR (VMR_{O_2}) in the constant C. In the arts model the separation between the oxygen VMR and the constant C is explicitely done. In this case follows:

$$C = 0.20946 \cdot \widehat{C} \tag{3.36}$$

$$\widehat{C} = 1.22 \cdot 10^{-19}\ [1/(\text{m Hz Pa})] \quad : \text{Rosenkranz} \tag{3.37}$$

$$\widehat{C} = 1.23 \cdot 10^{-19}\ [1/(\text{m Hz Pa})] \quad : \text{MPM93} \tag{3.38}$$

The width parameter w is in both models the same, $w = 5.6{\cdot}10^3$ Hz/Pa. If we define the width γ in a more general way like

$$\gamma = w \cdot (A \cdot P_d \cdot \Theta^{n_d} + B \cdot P_{H2O} \cdot \Theta^{n_w}) \tag{3.39}$$

we can fit both models, the Rosenkranz and the MPM93 model, into the same parameterization with ($A = 1$, $B = 1.1$, $n_d = 0.8$, $n_w = 1.0$) for the Rosenkranz model and ($A = 1.0$, $B = 1.0$, $n_d = 0.8$, $n_w = 0.8$) for MPM93.

The oxygen continuum absorption term is proportional to the collision frequency of a single oxygen molecule with other air molecules and thus proportional to the dry air pressure[1].

[1] The absorption due to weakly bound complexes of O_2–X with $X = H_2O$, N_2 is treated separately and therefore not included in this Debye formula.

3.2.3 Nitrogen Continuum Absorption

Since molecular nitrogen has in its unperturbed state no electric or magnetic dipole moment (but an electric quadrupole moment), it shows no rotational spectral signature in the microwave region. Regardless of this, nitrogen absorbs radiation in this frequency range due to collision induced absorption (CIA). Far–infrared roto-translational band structures from free–free interactions give rise to far wing absorption below 1 THz.

Different parameterizations of this absorption term for the frequency range below 1 THz are available *Rosenkranz* [1993]; *Liebe et al.* [1993]; *Borysow and Frommhold* [1986]. Common to all these models is the quadratic dependency on N_2 partial pressure which is a direct consequence of the underlying CIA processes involved. The simplest model is given by *Rosenkranz* [1993], which uses the same parameterization as for the water vapor continuum, described in Equation 3.27:

$$\alpha_{\mathrm{c}} = C \cdot \nu^{n_\nu} \cdot \Theta^{n_T} \cdot P_{\mathrm{N2}}^{n_p} \tag{3.40}$$

with $C = 4.56 \cdot 10^{-13}$ dB/(km hPa^2 GHz^2), $n_\nu = 2$, $n_T = 3.55$, and $n_p = 2$, respectively. The laboratory data set for the determination of C is manly from *Dagg et al.* [1975, 1978] around 70 and 140 GHz, respectively.

The MPM models has compared with Equation 3.40 an additional frequency dependent term which leads to the following expression

$$\alpha_{\mathrm{c}} = \widehat{C} \cdot (1.0 - 1.2 \cdot 10^{-5} \cdot \nu^{1.5}) \cdot \nu^2 \cdot \Theta^{3.5} \cdot P_{\mathrm{d}}^2 \quad : \text{MPM89} \tag{3.41}$$

$$\alpha_{\mathrm{c}} = \widehat{C} \cdot \frac{\nu^2}{(1.0 + a \cdot \nu^{n_\nu})} \cdot \Theta^{3.5} \cdot P_{\mathrm{d}}^2 \quad : \text{MPM93} \tag{3.42}$$

$$a =$$

where the parameter is $\widehat{C} = 2.55 \cdot 10^{-13}$ dB/(km hPa^2 GHz^2), a =1.9$\cdot$10^{-5} GHz$^{-n_\nu}$, and $n_\nu = 1.5$. based on data from *Stankevich* [1974] and *Stone et al.* [1984]. With respect to the 22 GHz water vapor line, the additional frequency terms in brackets in Equations 3.41 and 3.42 are nearly unity and therefore not essential. Therefore all three parameterizations have the same frequency and temperature relationship, but the absolute magnitude is in the case of Rosenkranz 80 % higher compared with the MPM models.

The not yet in arts implemented model of Borysow and Frommhold[2] is somewhat different since their focus is mainly on the radiative transfer in the Titan's atmosphere with the infrared interferometer spectrometer, IRIS, on board the Voyager Spacecraft. This detailed model is primarily designed to parameterize each of the roto-translational spectral lines around 200 cm^{-1} ($\approx$ 6 THz) accurately. The analyzed data set incorporate the data source used by the Rosenkranz but is largely extended with measurements in the far–infrared.

3.2.4 Carbon dioxide Continuum Absorption

Rosenkranz [1993] gives a similar parameterization for the CO_2-continuum absorption term as for the nitrogen continuum, with

$$\alpha_{\mathrm{c}} = \cdot \nu^2 \cdot \left[C_s \cdot P_{\mathrm{CO2}}^2 \cdot \Theta^{n_s} + C_f \cdot P_{\mathrm{CO2}} \cdot P_{\mathrm{N2}} \cdot \Theta^{n_f} \right] \tag{3.43}$$

[2]the source code of this model can be downloaded from the home page of A. Borysow: http://www.astro.ku.dk/∼aborysow/

where the parameter values $C_s = 3.23 \cdot 10^{-11}$ dB/(km hPa^2 GHz^2), $C_f = 1.18 \cdot 10^{-11}$ dB/(km hPa^2 GHz^2), $n_s = 5.08$, and $n_f = 4.7$, respectively, are determined from laboratory measurements of *Ho et al.* [1966]; *Dagg et al.* [1975]. Since the foreign term includes only nitrogen as perturber, one can get an estimate for dry air by replacing P_{N2} by the dry air partial pressure in Equation 3.43. Because nitrogen is usually a more efficient perturber than oxygen, this estimation can be regarded as an upper limit. Concerning the Earth's atmosphere, the foreign broadening term is more interesting since the carbon dioxide partial pressure is only approximately 0.04 % of the nitrogen partial pressure up to 90 km.

3.2.5 ARTS Workspace Variables and Methods

This section explains how the above described continua are represented in the structure of the arts source code and how one can invoke them in the arts control file.

The continuum tags need more input specification than normal trace gas tags. Why this is so can be seen from Eq. 3.27 and Table 3.4. For a single function for the water vapor continuum we find several different function parameters in the literature. To solve this ambiguity arts has two methods implemented which helps the user to select a single set of parameters in an easy way. In connection with this input parameters we distinguish generally two types, the referenced models which are taken from the literature (e. g. *Liebe et al.* [1993] or *Rosenkranz* [1993]) and the user model, for which the arts user is providing the necessary parameter values.

After selecting the continuum tag with the `tagDefine` method, the arts user has to setup the arts internal structure (i. e. the workspace variables *cont_description_names, cont_description_models, and cont_description_parameters*) for the selected continuum tags, which can simply be done by putting the following line into the arts control file:

```
cont_descriptionInit{}
```

After this initialization is done, the continuum tag specific information has to be transfered to arts. This is possible with the arts method *cont_descriptionAppend*, which has itself three input variables: *tagname*, *model*, and *userparameters*. The user has to specify these input variables in the arts control file for each selected continuum tag. Below is a list of all the implemented continuum tags and the associated valid range of the input variables for *cont_descriptionAppend*. For a condensed overview of the possible continuum tags and their referenced models see Table 3.7 and the online documentation can be found under *arts/doc/doxygen/html/continua_cc.html*.

One has to note at this place that the two input variables *model* and *userparameters* are to some extend redundant. Therefore one can also produce an ambiguity by giving contradicting values for these two input variables. To avoid such ambiguities the arts user should keep in mind the general rule that only the user model (*model* =*"user"*) needs input parameters via the input variable *userparameters*. All the referenced models need no input via *userparameters*. If you try to run the arts control file with a referenced model and input parameters you will get an error message. Below in the detailed description of *cont_descriptionAppend* you can find correct examples for all the continuum tags.

- The general water vapor continuum function as described in Eq. (3.27) is divided up into two separate tags in arts. The term proportional to P^2_{H2O} has the tag name

`"H2O-SelfContStandardType"`. This tag can be used for eider the referenced models MPM87 [*Liebe and Layton*, 1987], MPM89 [*Liebe*, 1989], CP98 [*Cruz Pol et al.*, 1998], PWR98 [*Rosenkranz*, 1998] or with an arbitrary user defined model parameter set ($C^{\mathrm{o}}_{\mathrm{H_2O}}$, $\mathrm{n_S}$). The information if a referenced or a user model is selected is given to arts via the input parameter *model* of the method *cont_descriptionAppend*. The valid models are "MPM87" (for the MPM87 model), "MPM89" (for the MPM89 model), "CruzPol" (for the CP98 model), "Rosenkranz" (for the PWR98 model), and "user" (for the user defined model). Only in the case of the user defined model has thew user to specify the parameter set ($C^{\mathrm{o}}_{\mathrm{H_2O}}$, $\mathrm{n_S}$) via the input parameter *userparameters*. Here one has to obey the arts units for $C^{\mathrm{o}}_{\mathrm{H_2O}}$ and $\mathrm{n_S}$ which are [1/(m·Hz^2·Pa^2)] and [1], respectively. For all the referenced models the necessary parameters are internally stored in arts and have not to be given by the user. Therefore the input parameter *userparameters* is empty in these cases. However, if you give some valuers via *userparameters* to arts and select simultaneously a referenced model via the input parameter *model* you will get a warning message. It is a general rule that in the method *cont_descriptionAppend* the value of the input parameter *model* dominates over the values specified with input parameter *userparameters*, so the calculation will be performed according to the selected model.

The following list describes all the valid combinations of parameters for the tag `"H2O-SelfContStandardType"` are listed (the values for the model user are just example values):

```
cont_descriptionAppend{
    tagname        = "H2O-SelfContStandardType"
    model          = "Rosenkranz"
    userparameters = [ ]
}
cont_descriptionAppend{
    tagname        = "H2O-SelfContStandardType"
    model          = "CruzPol"
    userparameters = [ ]
}
cont_descriptionAppend{
    tagname        = "H2O-SelfContStandardType"
    model          = "MPM87"
    userparameters = [ ]
}
cont_descriptionAppend{
    tagname        = "H2O-SelfContStandardType"
    model          = "MPM89"
    userparameters = [ ]
}
cont_descriptionAppend{
    tagname        = "H2O-SelfContStandardType"
    model          = "user"
    userparameters = [ 2.046e-33, 5.289 ]
```

```
}
```

- The general water vapor continuum function as described in Equation (3.27) is divided up into two separate tags in arts. The term proportional to P_{H2O}^2 was explained above while the term proportional to $P_{\mathrm{H2O}} \cdot P_{\mathrm{d}}$ is named `"H2O-ForeignContStandardType"` will be described here. This tag can be used for eider the referenced models MPM87 [*Liebe and Layton*, 1987], MPM89 [*Liebe*, 1989], CP98 [*Cruz Pol et al.*, 1998], PWR98 [*Rosenkranz*, 1998] or with an arbitrary user defined model parameter set ($C^{\mathrm{o}}_{\mathrm{H_2O}}$, $\mathrm{n_S}$). The information if a referenced or a user model is selected is given to arts via the input parameter *model* of the method *cont_descriptionAppend*. The valid models are "MPM87"(for the MPM87 model), "MPM89"(for the MPM89 model), "CruzPol"(for the CP98 model), "Rosenkranz"(for the PWR98 model), and "user"(for the user defined model). Only in the case of the user defined model has thew user to specify the parameter set ($C^{\mathrm{o}}_{\mathrm{H_2O}}$, $\mathrm{n_S}$) via the input parameter *userparameters*. Here one has to obey the arts units for $C^{\mathrm{o}}_{\mathrm{H_2O}}$ and $\mathrm{n_S}$ which are $[1/(\mathrm{m}\cdot\mathrm{Hz}^2\cdot\mathrm{Pa}^2)]$ and $[1]$, respectively. For all the referenced models the necessary parameters are internally stored in arts and have not to be given by the user. Therefore the input parameter *userparameters* is empty in these cases. However, if you give some valuers via *userparameters* to arts and select simultaneously a referenced model via the input parameter *model* you will get a warning message. It is a general rule that in the method *cont_descriptionAppend* the value of the input parameter *model* dominates over the values specified with input parameter *userparameters*, so the calculation will be performed according to the selected model.

  The following list describes all the valid combinations of parameters for the tag `"H2O-ForeignContStandardType"` are listed (the values for the model user are just example values):

```
cont_descriptionAppend{
    tagname        = "H2O-ForeignContStandardType"
    model          = "Rosenkranz"
    userparameters = [ ]
}
cont_descriptionAppend{
    tagname        = "H2O-ForeignContStandardType"
    model          = "CruzPol"
    userparameters = [ ]
}
cont_descriptionAppend{
    tagname        = "H2O-ForeignContStandardType"
    model          = "MPM87"
    userparameters = [ ]
}
cont_descriptionAppend{
    tagname        = "H2O-SelfContStandardType"
    model          = "MPM89"
    userparameters = [ ]
```

```
}
cont_descriptionAppend{
    tagname        = "H2O-ForeignContStandardType"
    model          = "user"
    userparameters = [ 5.793e-35, 1.551 ]
}
```

- The MPM version 1993 ("MPM93") [*Liebe et al.*, 1993] has a special treatment of the water vapor continuum. The simple function of Eq. 3.27 is therefore not valid. In MPM93 the H_2O-continuum is described by a pseudo water vapor line in the far-infrared (FIR) region. This means that seven parameters are needed, one central frequency and six line parameters. The original parameter set is given in *Liebe et al.* [1993]. The MPM93 water vapor continuum has the arts tag name `"H2O-ContMPM93"`. Since this parameterization is very special, the valid models in the arts method *cont_descriptionAppend* are only "MPM93" and "user". For the referenced model "MPM93" the input parameter *userparameters* must be empty while for the user model one has to specify all seven continuum parameters in arts units: the pseudo-line center frequency, ν^* (in Hz), and the six pseudo-line parameters $\mathrm{b}_1^*,\cdots,\mathrm{b}_6^*$ (in units of [Hz/Pa], [1], [Hz/Pa], [1], [1], [1]).

  In the following all the valid possibilities for the tag `"H2O-ContMPM93"` are listed (the values for the model user are just example values):

```
cont_descriptionAppend{
    tagname        = "H2O-ContMPM93"
    model          = "MPM93"
    userparameters = [ ]
}
cont_descriptionAppend{
    tagname        = "H2O-ContMPM93"
    model          = "user"
    userparameters = [ 1780.0e9, 22300.0, 0.952,
                       17.60e4, 30.5, 2.0, 5.0 ]
}
```

- The oxygen continuum models of Rosenkranz [*Rosenkranz*, 1993] and MPM93 [*Liebe et al.*, 1993], as described in Section 3.2.2, are only slightly different in some respect so that both can be described with a single arts tag named `"O2-SelfContStandardType"`. The only discrepancy is in the formulation of the width γ. If the user model is selected, then the following parameter set has to be given: ($\widehat{C}$, w, A, B, n_d, n_w) which are explained in Eqs. (3.33) and (3.39). The units of these parameters are: [1/(m·Pa·Hz)], [Hz/Pa], [1], [1], [1], [1]. For a description of these parameters see Eqs. (3.33), (3.36), and (3.39).

  The following list describes all the valid combinations of parameters for the tag `"O2-SelfContStandardType"` are listed (the values for the model user are just example values):

```
cont_descriptionAppend{
    name           = "O2-SelfContStandardType"
    model          = "Rosenkranz"
    userparameters = [ ]
}
cont_descriptionAppend{
    name           = "O2-SelfContStandardType"
    model          = "MPM93"
    userparameters = [ ]
}
cont_descriptionAppend{
    name           = "O2-SelfContStandardType"
    model          = "user"
    userparameters = [ 1.23e-19, 5600.0,
                       1.0, 1.0, 1.0, 1.0 ]
}
```

- The *Rosenkranz* [1993] N_2-N_2 continuum as explained in Section 3.2.3 has the arts tag name "N2-SelfContStandardType". For this tag two possible models are implemented: the referenced model of Rosenkranz and the user model. In case of the user model one has to provide the parameters C, n_ν, n_T , and n_p in units of [1/(m·Pa^2·Hz^2], [1], [1], and [1], respectively. Below are the two possibilities illustrated:

```
cont_descriptionAppend{
    tagname        = "N2-SelfContStandardType"
    model          = "Rosenkranz"
    userparameters = [ ]
}
cont_descriptionAppend{
    tagname        = "N2-SelfContStandardType"
    model          = "user"
    userparameters = [ 1.05e-38, 2.00, 3.55, 2.00 ]
}
```

- The MPM93 model [*Liebe et al.*, 1993] has a different approach for the N_2-N_2 continuum than *Rosenkranz* [1993], as shown by Eq. 3.41. To be able to use this referenced model, one has to use the tag called "N2-SelfContMPM93". Similarly to the N_2-continuum tag described before, two allowed models are implemented: "MPM93" and "user" (see the examples below). In the case of the user model the parameters C, a, and n_ν in units of [1/(m·Pa^2·Hz^2], [GHz$^{-n_\nu}$], and [1], respectively.

```
cont_descriptionAppend{
    tagname        = "N2-SelfContMPM93"
    model          = "MPM93"
    userparameters = [ ]
```

```
}
cont_descriptionAppend{
    tagname        = "N2-SelfContMPM93"
    model          = "user"
    userparameters = [ 5.87e-39, 6.10e-19 1.5 ]
}
```

- The *Rosenkranz* [1993] CO_2-CO_2 continuum term is according to Equation (3.43) of Section 3.2.4 proportional to P^2_{CO2}. This absorption feature has the arts tag name `"CO2-SelfContPWR93"`. For this tag two possible models are implemented: the referenced model of Rosenkranz and the user model. In case of the user model one has to provide the parameters C_s and n_s in units of [1/(m·Pa^2·Hz^2] and [1], respectively. Below are the two possibilities illustrated:

```
cont_descriptionAppend{
    tagname        = "CO2-SelfContPWR93"
    model          = "Rosenkranz"
    userparameters = [ ]
}
cont_descriptionAppend{
    tagname        = "CO2-SelfContPWR93"
    model          = "user"
    userparameters = [ 7.43e-37, 5.08 ]
}
```

- The *Rosenkranz* [1993] CO_2-N_2 continuum term is according to as explained in Equation (3.43) of Section 3.2.4 proportional to $P_{\mathrm{CO2}} \cdot P_{\mathrm{N2}}$. This absorption feature has the arts tag name `"CO2-ForeignContPWR93"`. For this tag two possible models are implemented: the referenced model of Rosenkranz and the user model. In case of the user model one has to provide the parameters C_f and n_f in units of [1/(m·Pa^2·Hz^2] and [1], respectively. Below are the two possibilities illustrated:

```
cont_descriptionAppend{
    tagname        = "CO2-ForeignContPWR93"
    model          = "Rosenkranz"
    userparameters = [ ]
}
cont_descriptionAppend{
    tagname        = "CO2-ForeignContPWR93"
    model          = "user"
    userparameters = [ 2.71e-37, 4.7]
}
```

All the continuum tags have their individual fixed line shape. Therefore the arts methods which controls this input information should be defined as follows: *shape* = "no_shape" , *normalizationfactor* = "no_norm", and *cutoff* = -1. An example for the arts method *lineshape_per_tgDefine* is presented below for the case of two selected continuum tags in the method *tagDefine*:

```
lineshape_per_tgDefine{
        shape               = [ "no_shape",
                                "no_shape"]
        normalizationfactor = [ "no_norm",
                                "no_norm"]
        cutoff              = [ -1,
                                -1]
}
```

continuum			*cont_descriptionAppend* input input parameter	reference/ arts uguide	arts source code function
			water vapor (H_2O)		
Rosenkranz	tagname	=	`"H2O-SelfContStandardType"`	*Rosenkranz* [1998]	Standard_H2O_self_continuum
$H_2O - H_2O$	model	=	”Rosenkranz”	Sec. 3.2.1	
	userparameters	=	[ ]		
Rosenkranz	tagname	=	`"H2O-ForeignContStandardType"`	*Rosenkranz* [1998]	Standard_H2O_foreign_continuum
H_2O−dry air	model	=	”Rosenkranz”	Sec. 3.2.1	
	userparameters	=	[ ]		
Cruz-Pol	tagname	=	`"H2O-SelfContStandardType"`	*Cruz Pol et al.* [1998]	Standard_H2O_self_continuum
$H_2O - H_2O$	model	=	”CruzPol”	Sec. 3.2.1	
	userparameters	=	[ ]		
Cruz-Pol	tagname	=	`"H2O-ForeignContStandardType"`	*Cruz Pol et al.* [1998]	Standard_H2O_foreign_continuum
H_2O−dry air	model	=	”CruzPol”	Sec. 3.2.1	
	userparameters	=	[ ]		
MPM87	tagname	=	`"H2O-SelfContStandardType"`	*Liebe and Layton* [1987]	Standard_H2O_self_continuum
$H_2O - H_2O$	model	=	”MPM87”	Sec. 3.2.1	
	userparameters	=	[ ]		
MPM87	tagname	=	`"H2O-ForeignContStandardType"`	*Liebe and Layton* [1987]	Standard_H2O_foreign_continuum
H_2O−dry air	model	=	”MPM87”	Sec. 3.2.1	
	userparameters	=	[ ]		
MPM89	tagname	=	`"H2O-SelfContStandardType"`	*Liebe* [1989]	Standard_H2O_self_continuum
$H_2O - H_2O$	model	=	”MPM89”	Sec. 3.2.1	
	userparameters	=	[ ]		
MPM89	tagname	=	`"H2O-ForeignContStandardType"`	*Liebe* [1989]	Standard_H2O_foreign_continuum
H_2O−dry air	model	=	”MPM89”	Sec. 3.2.1	
	userparameters	=	[ ]		
MPM93	tagname	=	`"H2O-ContMPM93"`	*Liebe et al.* [1993]	MPM93_H2O_continuum

Table 3.7: (continued)

continuum			*cont_descriptionAppend* input parameter	reference/ arts uguide	arts source code function
H_2O-air	model	=	"MPM93"	Sec. 3.2.1	
	userparameters	=	[ ]		
			oxygen (O_2)		
Rosenkranz	tagname	=	`"SelfContStandardType"`	*Rosenkranz* [1993]	Standard_O2_continuum
O_2-air	model	=	"Rosenkranz"	Sec. 3.2.2	
	userparameters	=	[ ]		
MPM93	tagname	=	`"O2-SelfContStandardType"`	*Liebe et al.* [1993]	Standard_O2_continuum
O_2-air	model	=	"MPM93"	Sec. 3.2.2	
	userparameters	=	[ ]		
			nitrogen (N_2)		
Rosenkranz	tagname	=	`"N2-SelfContStandardType"`	*Rosenkranz* [1993]	Standard_N2_self_continuum
N_2-N_2	model	=	"Rosenkranz"	Sec. 3.2.3	
	userparameters	=	[ ]		
MPM93	tagname	=	`"N2-SelfContMPM93"`	*Liebe et al.* [1993]	MPM93_N2_continuum
N_2-N_2	model	=	"Rosenkranz"	Sec. 3.2.3	
	userparameters	=	[ ]		
			carbon dioxide CO_2		
Rosenkranz	tagname	=	`"CO2-SelfContPWR93"`	*Rosenkranz* [1993]	Rosenkranz_CO2_self_continuum
CO_2-CO_2	model	=	"Rosenkranz"	Sec. 3.2.4	
	userparameters	=	[ ]		
Rosenkranz	tagname	=	`"CO2-ForeignContPWR93"`	*Rosenkranz* [1993]	Rosenkranz_CO2_foreign_continuum
CO_2-N_2	model	=	"Rosenkranz"	Sec. 3.2.4	
	userparameters	=	[ ]		

Table 3.7: This table gives an overview of the implemented referenced continua models and how they are specified in the arts method *cont_descriptionAppend*. Additionally the reference and the arts source code function name (see file *arts/src/continua.cc* are provided. The detailed online documentation can be found under *arts/doc/doxygen/html/continua_cc.html*).

ARTS Example Control File for the Continuum Tags

Below you will find an example of a control file for all the implemented fixed model continua. Please note that to run this example control file you have to specify user specific path and input file names to run it properly. You can find this example in the arts directory *arts/doc/examples/cont_example.arts*

```
# ######## EXAMPLE CONTINUUM TAG CONTROL FILE #######
#-----------------------------------------------------
# define the arts continuum tags of arts
tgsDefine{
      [
        "H2O-SelfContStandardType",
        "H2O-ForeignContStandardType",
        "H2O-ContMPM93",
        "O2-SelfContStandardType",
        "N2-SelfContStandardType",
        "N2-SelfContMPM93",
        "CO2-SelfContPWR93",
        "CO2-ForeignContPWR93"
      ]
}
#-----------------------------------------------------
# initialize the continua tag structures
cont_descriptionInit{}
#
#-----------------------------------------------------
#
# define the continua tags with the appropriate input
#
# ----- H2O continuum
#
# Rosenkranz-type H2O-H2O continuum:
cont_descriptionAppend{
    tagname        = "H2O-SelfContStandardType"
    model          = "Rosenkranz"
    userparameters = [ ]
}
#
# Rosenkranz-type H2O-dry air continuum:
cont_descriptionAppend{
    tagname        = "H2O-ForeignContStandardType"
    model          = "Rosenkranz"
    userparameters = [ ]
}
#
# MPM93-type H2O-air continuum:
cont_descriptionAppend{
    tagname        = "H2O-ContMPM93"
    model          = "MPM93"
    userparameters = [ ]
}
#
# ----- O2 continuum
#
# Standard O2-air continuum:
cont_descriptionAppend{
    tagname        = "O2-SelfContStandardType"
    model          = "Rosenkranz"
    userparameters = [ ]
}
#
# ----- N2 continuum
#
# Rosenkranz N2-N2 continuum (only N2-N2 broadening):
cont_descriptionAppend{
    tagname        = "N2-SelfContStandardType"
    model          = "Rosenkranz"
    userparameters = [ ]
}
# MPM93 N2-N2 continuum (only N2-N2 broadening):
cont_descriptionAppend{
    tagname        = "N2-SelfContMPM93"
    model          = "MPM93"
    userparameters = [ ]
}
#
```

continuum tag selection for calculation.

initialize the continua tag description structure in arts. This is essential for the later use of the method cont_descriptionAppend.

description of every continum tag also mentioned in the tagDefine methode above. Each description has three input variables:

* tag name
* model to select a referenced model or the user model
* user given input parameters (only valid for model "user")

Only in the case where the model "user" is selected, the user given input parameters are considered. All other models neglect these input parameters.

```
# ----- CO2 continuum
#
# Rosenkranz CO2-CO2 continuum:
cont_descriptionAppend{
    tagname        = "CO2-SelfContPWR93"
    model          = "Rosenkranz"
    userparameters = [ ]
}
# Rosenkranz CO2-N2 continuum:
cont_descriptionAppend{
    tagname        = "CO2-ForeignContPWR93"
    model          = "Rosenkranz"
    userparameters = [ ]
}
#
#----------------------------------------------------
#
# Read the pressure, temperature, and altitude
# profiles and create the workspace variable 'raw_ptz'.
# ATTENTION! The path and file names are user specific!
MatrixReadAscii (raw_ptz)
        {"@ac_arts_data@/atmosphere/fascod/midlatitude-summer.tz.aa"}
#
# The same for the input VMR profiles
# ATTENTION! The path and file names are user specific!
raw_vmrsReadFromScenario
         {"@ac_arts_data@/atmosphere/fascod/midlatitude-summer"}
#
# Create the pressure grid 'p_abs' (just an example)
VectorNLogSpace(p_abs){
        start = 100000.000
        stop  =   1000.000
        n     = 100
}
# reads the input profiles
AtmFromRaw{}
#
#----------------------------------------------------
#
# Set the H2O profile
h2o_absSet{}
#
# Set the N2 profile
n2_absSet{}
#
#----------------------------------------------------
#
# Read spectral line data from HITRAN96 catalogue for
# the frequency range from 1 to 2 GHz.
# This in not essential for the continuum tags but
# bust be given as input for absCalc below.
# ATTENTION! THE PATH AND FILE NAMES ARE USER SPECIFIC!
#
lines_per_tgReadFromCatalogues{
  filenames = [ "@ac_arts_data@/spectroscopy/hitran96/hitran96_lowfreq.par" ]
  formats   = [ "HITRAN96" ]
  fmin      = [ 1.0e9 ]
  fmax      = [ 2.0e9 ]
}
#
# Create an example frequency grid 'f_mono'
VectorNLinSpace(f_mono){
        start =    100.0e9
        stop  =    200.0e9
        n     =    100
}
#
#----------------------------------------------------
#
#
```

Information about the model atmosphere. Also the VMR profiles H2O and N2 have to be given seperately.

Spectral line data is also necessary for the method absCalc.

Input frequency grid on which the calculation is performed.

```
# Set the lineshape function for each continuum tag
lineshape_per_tgDefine{
        shape               = [ "no_shape",
                                "no_shape",
                                "no_shape",
                                "no_shape",
                                "no_shape",
                                "no_shape",
                                "no_shape",
                                "no_shape"]
        normalizationfactor = [ "no_norm",
                                "no_norm",
                                "no_norm",
                                "no_norm",
                                "no_norm",
                                "no_norm",
                                "no_norm",
                                "no_norm"]
        cutoff              = [ -1,
                                -1,
                                -1,
                                -1,
                                -1,
                                -1,
                                -1,
                                -1]
}
#----------------------------------------------------
# calculate the absorption coefficients, unit=1/meter
absCalc{}
#----------------------------------------------------
# These we definitely want to write to files:
# 1. absorption coefficient per continuum tag
ArrayOfMatrixWriteAscii (abs_per_tg) {""}
# 2. temperature profile
VectorWriteAscii (t_abs)  {""}
# 3. altitude grid
VectorWriteAscii (z_abs)  {""}
# 4. pressure grid
VectorWriteAscii(p_abs)   {""}
# 5. frequency grid
VectorWriteAscii (f_mono) {""}
# 6. cont_descriptionAppend continuum tagnames
ArrayOfStringWriteAscii (cont_description_names) {""}
# 7. cont_descriptionAppend model selections
ArrayOfStringWriteAscii (cont_description_models) {""}
# 8. cont_descriptionAppend user given input parameters
ArrayOfVectorWriteAscii (cont_description_parameters) {""}
#####################################################
```

The line shape of the continuum tags are all internally set. Therefore The user has not to specify the line shape here.

This is the method which calculates the absorption coefficients in units of 1/meters.

Here the output is written into the output files.

3.3 Complete Absorption Models

The MPM absorption model of Liebe and coworkers consists of modules for water vapor and oxygen absorption. The Rosenkranz (PWR98) absorption model include also H_2O and O_2 while the Cruz-Pol et al. (CP98) absorption models include absorption due to water vapor. Additionally the CP98 model has a strongly reduced parameter set for the H_2O-line absorption since it is especially intended for the range around the 22 GHz water line. The MPM and R98 are valid from the microwave up to the submillimeter frequency range

(1-1000 GHz).

Implemented in ARTS are the following modules of the above mentioned models:

species	model
H_2O	MPM87, MPM89, MPM93, PWR98, CP98
O_2	MPM93, PWR98

3.3.1 Complete Water Vapor Models

In ARTS several complete water vapor absorption models are implemented and can easily be used. Implemented models are the versions MPM87 *Liebe and Layton* [1987], MPM89 *Liebe* [1989], and MPM93 *Liebe et al.* [1993] of the Liebe Millimeter-wave Propagation Model and additionally the models of Cruz-Pol et al. (CP98) *Cruz Pol et al.* [1998] and P. W. Rosenkranz (PWR98) *Rosenkranz* [1998]. MPM and PWR98 are especially desigend for fast absorption calculations in the frequency range of 1-1000 GHz while the CP98 model is a reduced model for a narrow frequency band around the 22 GHz H_2O-line (especially used by ground-based radiometers).

The total water vapor absorption (α_{tot}) is in all the stated models described by a line absorption (α_ℓ) term and a continuum absorption (α_{c}) term:

$$\alpha_{\mathrm{tot}} = \alpha_\ell + \alpha_{\mathrm{c}} \tag{3.44}$$

The main differences between the different models is the line shape used for α_ℓ and the formulation of α_{c}.

It has to be emphasized that, α_ℓ and α_{c} of different models are not necessarily compatible and should therefore not be interchanged between different models.

MPM87 Water Vapor Absorption Model

This version, which is described in *Liebe and Layton* [1987] and follows the general line of the MPM model to divide the total water vapor absorption, $\alpha_{\mathrm{tot}}^{\mathrm{MPM87}}$, into a spectral line term, $\alpha_\ell^{\mathrm{MPM87}}$, and a continuum term not attributed to spectral lines, $\alpha_{\mathrm{c}}^{\mathrm{MPM87}}$:

$$\alpha_{\mathrm{tot}}^{\mathrm{MPM87}} = \alpha_\ell^{\mathrm{MPM87}} + \alpha_{\mathrm{c}}^{\mathrm{MPM87}} \qquad \mathrm{dB/km} \tag{3.45}$$

Water Vapor Line Absorption: The MPM87 *Liebe and Layton* [1987] water vapor line catalog consists of 30 lines from 22 GHz up to 988 GHz. The center frequencies and parameter values are listed in Table 3.8. To describe the line absorption, a set of three parameters ($\mathrm{b}_{1,\mathrm{k}}$ and $\mathrm{b}_{3,\mathrm{k}}$) per line are used: two for the line strength and one for the line width. The total line absorption coefficient (in units of dB/km) is the sum over all individual line absorption coefficients[3]:

$$\alpha_\ell^{\mathrm{MPM87}} = 0.1820 \cdot \nu_{\mathrm{k}} \cdot P_{\mathrm{H2O}} \cdot \sum_k S_{\mathrm{k}}(T) \cdot F(\nu, \nu_{\mathrm{k}}) \qquad \mathrm{dB/km} \tag{3.46}$$

[3]The factor $0.1820 \cdot 10^6$ is equal to $(4\,\pi/c) \cdot 10 \log{(e)}$ (the term $(4\,\pi/c)$ comes from the definition of the absorption coefficient in terms of the dielectric constant and the term $10\ \log{(e)}$ is due to the definition of the Decibel.) The velocity of light is defined as $c = 2.9979 \cdot 10^{-4}$ km GHz. The factor 10^6 is incorporated into the line strength and does therefore not appear in the pre-factor.

where $S_k(T)$ is the line intensity described by the parameterization

$$S_k(T) = \mathrm{b}_{1,k} \cdot \mathrm{P}_{H2O} \cdot \Theta^{3.5} \cdot \exp\left(\mathrm{b}_{2,k} \cdot [1-\Theta]\right) \qquad \text{kHz} \tag{3.47}$$

with ν_k as the line center frequency, P_{H2O} the water vapor partial pressure and $\Theta = 300\,\mathrm{K}/T$.
The line shape function, $F(\nu, \nu_k)$, in Eq. (3.46) is the standard Van Vleck-Weisskopf (VVW) function, given by:

$$F(\nu, \nu_k) \quad = \quad \left(\frac{\nu}{\nu_k}\right) \cdot \left[\frac{\gamma_k}{(\nu-\nu_k)^2+\gamma_k^2} + \frac{\gamma_k}{(\nu+\nu_k)^2+\gamma_k^2}\right] \tag{3.48}$$

(3.49)

The pressure broadened line width, γ_k, is calculated with the single parameter $\mathrm{b}_{3,k}$ in the following way:

$$\gamma_k = \mathrm{b}_{3,k} \cdot (4.80 \cdot \mathrm{P}_{H2O} \cdot \Theta^{1.1} + \mathrm{P}_d \cdot \Theta^{0.6}) \qquad \text{GHz} \tag{3.50}$$

where P_d is the partial pressure of dry air ($P_d = P_{tot} - P_{H2O}$). The parameterizations of $S_k(T)$ and γ_k are already in use for the early version of MPM81 *Liebe* [1981].

k	ν_k [GHz]	$\mathrm{b}_{1,k}$ $[\frac{\text{kHz}}{\text{kPa}}]$	$\mathrm{b}_{2,k}$ [1]	$\mathrm{b}_{3,k}$ $[\frac{\text{GHz}}{\text{kPa}}]$
1	22.235080	0.1090	2.143	$27.84\cdot 10^{-3}$
2	67.813960	0.0011	8.730	$27.60\cdot 10^{-3}$
3	119.995940	0.0007	8.347	$27.00\cdot 10^{-3}$
4	183.310117	2.3000	0.653	$31.64\cdot 10^{-3}$
5	321.225644	0.0464	6.156	$21.40\cdot 10^{-3}$
6	325.152919	1.5400	1.515	$29.70\cdot 10^{-3}$
7	336.187000	0.0010	9.802	$26.50\cdot 10^{-3}$
8	380.197372	11.9000	1.018	$30.36\cdot 10^{-3}$
9	390.134508	0.0044	7.318	$19.00\cdot 10^{-3}$
10	437.346667	0.0637	5.015	$13.70\cdot 10^{-3}$
11	439.150812	0.9210	3.561	$16.40\cdot 10^{-3}$
12	443.018295	0.1940	5.015	$14.40\cdot 10^{-3}$
13	448.001075	10.6000	1.370	$23.80\cdot 10^{-3}$
14	470.888947	0.3300	3.561	$18.20\cdot 10^{-3}$
15	474.689127	1.2800	2.342	$19.80\cdot 10^{-3}$
16	488.491133	0.2530	2.814	$24.90\cdot 10^{-3}$
17	503.568532	0.0374	6.693	$11.50\cdot 10^{-3}$
18	504.482692	0.0125	6.693	$11.90\cdot 10^{-3}$
19	556.936002	510.0000	0.114	$30.00\cdot 10^{-3}$
20	620.700807	5.0900	2.150	$22.30\cdot 10^{-3}$
21	658.006500	0.2740	7.767	$30.00\cdot 10^{-3}$
22	752.033227	250.0000	0.336	$28.60\cdot 10^{-3}$
23	841.073593	0.0130	8.113	$14.10\cdot 10^{-3}$
24	859.865000	0.1330	7.989	$28.60\cdot 10^{-3}$

Table 3.8: (continued)

k	ν_k	$b_{1,k}$	$b_{2,k}$	$b_{3,k}$
25	899.407000	0.0550	7.845	$28.60 \cdot 10^{-3}$
26	902.555000	0.0380	8.360	$26.40 \cdot 10^{-3}$
27	906.205524	0.1830	5.039	$23.40 \cdot 10^{-3}$
28	916.171582	8.5600	1.369	$25.30 \cdot 10^{-3}$
29	970.315022	9.1600	1.842	$24.00 \cdot 10^{-3}$
30	987.926764	138.0000	0.178	$28.60 \cdot 10^{-3}$

Table 3.8: List of H_2O spectral lines and their spectroscopic parameters (H_2O-air mixture) for the MPM87 model *Liebe and Layton* [1987].

Water Vapor Continuum Absorption: The water vapor continuum absorption coefficient in MPM87, $\alpha_c^{\mathrm{MPM87}}$, is determined from laboratory measurements at 137.8 GHz by Liebe and Layton covering the following parameter range:

temperature	282-316 K
relative humidity	0-95 %
dry air pressure	0 - 160 kPa

The mathematical expression of $\alpha_c^{\mathrm{MPM87}}$ is derived from the far wing approximation of the line absorption and is expressed as follows

$$\alpha_c^{\mathrm{MPM87}} = \nu^2 \cdot P_{\mathrm{H2O}} \cdot (C^{\mathrm{o}}_{\mathrm{H_2O}} \cdot P_{\mathrm{H2O}} \cdot \Theta^{\mathrm{ns}} + C^{\mathrm{o}}_{\mathrm{d}} \cdot P_{\mathrm{d}} \cdot \Theta^{\mathrm{n}}\mathrm{f}), \tag{3.51}$$

with the continuum parameter set $C^{\mathrm{o}}_{\mathrm{H_2O}}$, $C^{\mathrm{o}}_{\mathrm{d}}$, $\mathrm{n_s}$, and $\mathrm{n_f}$. The determined values of the continuum parameters are:

$C^{\mathrm{o}}_{\mathrm{H_2O}} = 6.496 \cdot 10^{-6}$ (dB/km) / (hPa·GHz)^2

$\mathrm{n_s} = 10.5$

$C^{\mathrm{o}}_{\mathrm{d}} = 0.206 \cdot 10^{-6}$ (dB/km) / (hPa·GHz)^2

$\mathrm{n_d} = 3.0$

MPM89 Water Vapor Absorption Model

MPM89 is described in *Liebe* [1989] and follows the general line of the MPM model to devide the total water vapor absorption, $\alpha_{\mathrm{tot}}^{\mathrm{MPM89}}$, into a spectral line term, $\alpha_{\ell}^{\mathrm{MPM89}}$, and a continuum term not attributed to spectral lines, $\alpha_c^{\mathrm{MPM89}}$:

$$\alpha_{\mathrm{tot}}^{\mathrm{MPM89}} = \alpha_{\ell}^{\mathrm{MPM89}} + \alpha_c^{\mathrm{MPM89}} \qquad \mathrm{dB/km} \tag{3.52}$$

All the absorption coefficients are calculated in units of dB/km.

Water Vapor Line Absorption: The MPM89 water vapor line catalog consists of the same 30 lines like MPM87 from 22 GHz up to 988 GHz. The center frequencies and parameter values are listed in Table 3.9. To describe the line absorption, a set of six parameters ($b_{1,k}$ and $b_{6,k}$) per line are used: two for the line strength and four for the line width. The total line absorption coefficient (in units of dB/km) is the sum over all individual line absorption coefficients[4]:

$$\alpha_\ell^{\text{MPM89}} = 0.1820 \cdot \nu_\text{k} \cdot P_\text{H2O} \cdot \sum_k S_\text{k}(T) \cdot F(\nu, \nu_\text{k}) \qquad \text{dB/km} \tag{3.53}$$

where $S_\text{k}(T)$ is the line intensity described by the parameterization

$$S_\text{k}(T) = \text{b}_{1,\text{k}} \cdot \text{P}_\text{H2O} \cdot \Theta^{3.5} \cdot \exp\left(\text{b}_{2,\text{k}} \cdot [1 - \Theta]\right) \qquad \text{kHz} \tag{3.54}$$

whit ν_k as the line center frequency, P_H2O the water vapor partial pressure and $\Theta = 300\,\text{K}/T$.
The line shape function, $F(\nu, \nu_\text{k})$, in Eq. (3.53) is the standard Van Vleck-Weisskopf (VVW) function, given by

$$F(\nu, \nu_\text{k}) = \left(\frac{\nu}{\nu_\text{k}}\right) \cdot \left[\frac{\gamma_\text{k}}{(\nu - \nu_\text{k})^2 + \gamma_\text{k}^2} + \frac{\gamma_\text{k}}{(\nu + \nu_\text{k})^2 + \gamma_\text{k}^2}\right] \tag{3.55}$$

where the pressure broadened line width, γ_k, is calculated as

$$\gamma_\text{k} = \text{b}_{3,\text{k}} \cdot (\text{b}_{5,\text{k}} \cdot \text{P}_\text{H2O} \cdot \Theta^{\text{b}_{6,\text{k}}} + \text{P}_\text{d} \cdot \Theta^{\text{b}_{4,\text{k}}}) \cdot 10^{-3} \qquad \text{GHz} \tag{3.56}$$

with $P_\text{d} = P_\text{tot} - P_\text{H2O}$ as the dry air partial pressure. The only difference between MPM87 and MPM89 with respect to the line absorption is the parameterization of the pressure broadened line width, γ_k, which is calculated with the four parameters $\text{b}_{3,\text{k}}$ to $\text{b}_{6,\text{k}}$ in the case of MPM89 whereas in MPM87 a single parameter ($\text{b}_{3,\text{k}}$) is used (see Eq. (3.50)).

k	ν_k [GHz]	$\text{b}_{1,\text{k}}$ $[\frac{\text{kHz}}{\text{kPa}}]$	$\text{b}_{2,\text{k}}$ [1]	$\text{b}_{3,\text{k}}$ $[\frac{\text{MHz}}{\text{kPa}}]$	$\text{b}_{4,\text{k}}$ [1]	$\text{b}_{5,\text{k}}$ [1]	$\text{b}_{6,\text{k}}$ [1]
1	22.235080	0.1090	2.143	28.11	0.69	4.80	1.00
2	67.813960	0.0011	8.735	28.58	0.69	4.93	0.82
3	119.995940	0.0007	8.356	29.48	0.70	4.78	0.79
4	183.310074	2.3000	0.668	28.13	0.64	5.30	0.85
5	321.225644	0.0464	6.181	23.03	0.67	4.69	0.54
6	325.152919	1.5400	1.540	27.83	0.68	4.85	0.74
7	336.187000	0.0010	9.829	26.93	0.69	4.74	0.61
8	380.197372	11.9000	1.048	28.73	0.69	5.38	0.84
9	390.134508	0.0044	7.350	21.52	0.63	4.81	0.55
10	437.346667	0.0637	5.050	18.45	0.60	4.23	0.48
11	439.150812	0.9210	3.596	21.00	0.63	4.29	0.52
12	443.018295	0.1940	5.050	18.60	0.60	4.23	0.50
13	448.001075	10.6000	1.405	26.32	0.66	4.84	0.67
14	470.888947	0.3300	3.599	21.52	0.66	4.57	0.65
15	474.689127	1.2800	2.381	23.55	0.65	4.65	0.64
16	488.491133	0.2530	2.853	26.02	0.69	5.04	0.72
17	503.568532	0.0374	6.733	16.12	0.61	3.98	0.43
18	504.482692	0.0125	6.733	16.12	0.61	4.01	0.45

Table 3.9: (continued)

[4] see footnote for MPM97 line absorption

k	ν_k	$\mathrm{b}_{1,\mathrm{k}}$	$\mathrm{b}_{2,\mathrm{k}}$	$\mathrm{b}_{3,\mathrm{k}}$	$\mathrm{b}_{4,\mathrm{k}}$	$\mathrm{b}_{5,\mathrm{k}}$	$\mathrm{b}_{6,\mathrm{k}}$
19	556.936002	510.0000	0.159	32.10	0.69	4.11	1.00
20	620.700807	5.0900	2.200	24.38	0.71	4.68	0.68
21	658.006500	0.2740	7.820	32.10	0.69	4.14	1.00
22	752.033227	250.0000	0.396	30.60	0.68	4.09	0.84
23	841.073593	0.0130	8.180	15.90	0.33	5.76	0.45
24	859.865000	0.1330	7.989	30.60	0.68	4.09	0.84
25	899.407000	0.0550	7.917	29.85	0.68	4.53	0.90
26	902.555000	0.0380	8.432	28.65	0.70	5.10	0.95
27	906.205524	0.1830	5.111	24.08	0.70	4.70	0.53
28	916.171582	8.5600	1.442	26.70	0.70	4.78	0.78
29	970.315022	9.1600	1.920	25.50	0.64	4.94	0.67
30	987.926764	138.0000	0.258	29.85	0.68	4.55	0.90

Table 3.9: List of H_2O spectral lines and their spectroscopic parameters (H_2O-air mixture) for the MPM89 model *Liebe* [1989].

Water Vapor Continuum Absorption: The MPM89 continuum absorption coefficients in, $\alpha_{\mathrm{c}}^{\mathrm{MPM89}}$, are identical as those in MPM87 (see Sec. 3.3.1 for details):

$$\alpha_{\mathrm{c}}^{\mathrm{MPM89}} = \nu^2 \cdot P_{\mathrm{H2O}} \cdot (C^{\mathrm{o}}_{\mathrm{H_2O}} \cdot P_{\mathrm{H2O}} \cdot \Theta^{\mathrm{ns}} + C^{\mathrm{o}}_{\mathrm{d}} \cdot P_{\mathrm{d}} \cdot \Theta^{\mathrm{n}}\mathrm{f}), \tag{3.57}$$

with

$C^{\mathrm{o}}_{\mathrm{H_2O}}$ = $6.496 \cdot 10^{-6}$ (dB/km) / (hPa·GHz)^2

$\mathrm{n_s}$ = 10.5

$C^{\mathrm{o}}_{\mathrm{d}}$ = $0.206 \cdot 10^{-6}$ (dB/km) / (hPa·GHz)^2

$\mathrm{n_d}$ = 3.0

MPM93 Water Vapor Absorption Model

This version, which is described in *Liebe et al.* [1993] and follows the general line of the MPM model to devide the total water vapor absorption, $\alpha_{\mathrm{tot}}^{\mathrm{MPM93}}$, into a spectral line term, $\alpha_{\ell}^{\mathrm{MPM93}}$, and a continuum term not attributed to spectral lines, $\alpha_{\mathrm{c}}^{\mathrm{MPM93}}$:

$$\alpha_{\mathrm{tot}}^{\mathrm{MPM93}} = \alpha_{\ell}^{\mathrm{MPM93}} + \alpha_{\mathrm{c}}^{\mathrm{MPM93}} \qquad \mathrm{dB/km} \tag{3.58}$$

The continuum absorption is parameterized like a resonant spectral line of H_2O, a so-called pseudo-line. This is a fundamental change in the parameterization of the water vapor continuum in respect to all older versions of MPM, which makes it quite complicate to compare the different versions, especially to distinguish a self- and foreign broadening term in the continuum.

Water Vapor Line Absorption: The water vapor line spectrum of MPM93 *Liebe et al.* [1993] consists of 34 lines below 1 THz (four more than in MPM89 and MPM87). To describe the MPM93 water vapor line absorption, a set of six parameters ($\mathrm{b_{1,k}}$ and $\mathrm{b_{3,k}}$) per line are used: two for the line strength and four for the line width. The total line absorption coefficient (in units of dB/km) is the sum over all individual line absorption coefficients[5]:

$$\alpha_\ell^{\mathrm{MPM93}} = 0.1820 \cdot \nu_\mathrm{k} \cdot P_\mathrm{H2O} \cdot \sum_k S_\mathrm{k}(T) \cdot F(\nu, \nu_\mathrm{k}) \qquad \mathrm{dB/km} \tag{3.59}$$

where $S_\mathrm{k}(T)$ is the line intensity described by the parameterization

$$S_\mathrm{k}(T) = \mathrm{b_{1,k}} \cdot \mathrm{P_{H2O}} \cdot \Theta^{3.5} \cdot \exp\left(\mathrm{b_{2,k}} \cdot [1 - \Theta]\right) \qquad \mathrm{kHz} \tag{3.60}$$

with ν_k as the line center frequency, P_H2O the water vapor partial pressure and $\Theta = 300\,\mathrm{K}/T$.
The line shape function, $F(\nu, \nu_\mathrm{k})$, in Eq. (3.46) is the standard Van Vleck-Weisskopf (VVW) function, given by:

$$F(\nu, \nu_\mathrm{k}) = \left(\frac{\nu}{\nu_\mathrm{k}}\right) \cdot \left[\frac{\gamma_\mathrm{k}}{(\nu - \nu_\mathrm{k})^2 + \gamma_\mathrm{k}^2} + \frac{\gamma_\mathrm{k}}{(\nu + \nu_\mathrm{k})^2 + \gamma_\mathrm{k}^2}\right] \tag{3.61}$$

(3.62)

The pressure broadened line width, γ_k, is calculated with the single parameter $\mathrm{b_{3,k}}$ in the following way:

$$\gamma_\mathrm{k} = \mathrm{b_{3,k}} \cdot (4.80 \cdot \mathrm{P_{H2O}} \cdot \Theta^{1.1} + \mathrm{P_d} \cdot \Theta^{0.6}) \qquad \mathrm{GHz} \tag{3.63}$$

where P_d is the partial pressure of dry air ($P_\mathrm{d} = P_\mathrm{tot} - P_\mathrm{H2O}$).

The parameterizations of $S_\mathrm{k}(T)$ was already in use for the early version of MPM81 *Liebe* [1981]. The expression for γ_k is the same as in MPM89. The main difference between MPM93 and MPM89 concerning the water vapor line absorption is the updated line catalog.

k	ν_k [GHz]	$\mathrm{b_{1,k}}$ [$\frac{\mathrm{kHz}}{\mathrm{hPa}}$]	$\mathrm{b_{2,k}}$ [1]	$\mathrm{b_{3,k}}$ [$\frac{\mathrm{MHz}}{\mathrm{hPa}}$]	$\mathrm{b_{4,k}}$ [1]	$\mathrm{b_{5,k}}$ [1]	$\mathrm{b_{6,k}}$ [1]
1	22.235080	0.01130	2.143	2.811	4.80	0.69	1.00
2	67.803960	0.00012	8.735	2.858	4.93	0.69	0.82
3	119.995940	0.00008	8.356	2.948	4.78	0.70	0.79
4	183.310091	0.24200	0.668	3.050	5.30	0.64	0.85
5	321.225644	0.00483	6.181	2.303	4.69	0.67	0.54
6	325.152919	0.14990	1.540	2.783	4.85	0.68	0.74
7	336.222601	0.00011	9.829	2.693	4.74	0.69	0.61
8	380.197372	1.15200	1.048	2.873	5.38	0.54	0.89
9	390.134508	0.00046	7.350	2.152	4.81	0.63	0.55
10	437.346667	0.00650	5.050	1.845	4.23	0.60	0.48
11	439.150812	0.09218	3.596	2.100	4.29	0.63	0.52
12	443.018295	0.01976	5.050	1.860	4.23	0.60	0.50
13	448.001075	1.03200	1.405	2.632	4.84	0.66	0.67

Table 3.10: (continued)

[5] see footnote for MPM97 line absorption

	ν_k	$b_{1,k}$	$b_{2,k}$	$b_{3,k}$	$b_{4,k}$	$b_{5,k}$	$b_{6,k}$
14	470.888947	0.03297	3.599	2.152	4.57	0.66	0.65
15	474.689127	0.12620	2.381	2.355	4.65	0.65	0.64
16	488.491133	0.02520	2.853	2.602	5.04	0.69	0.72
17	503.568532	0.00390	6.733	1.612	3.98	0.61	0.43
18	504.482692	0.00130	6.733	1.612	4.01	0.61	0.45
19^+	547.676440	0.97010	0.114	2.600	4.50	0.70	1.00
20^+	552.020960	1.47700	0.114	2.600	4.50	0.70	1.00
21	556.936002	48.74000	0.159	3.210	4.11	0.69	1.00
22	620.700807	0.50120	2.200	2.438	4.68	0.71	0.68
23^+	645.866155	0.00713	8.580	1.800	4.00	0.60	0.50
24	658.005280	0.03022	7.820	3.210	4.14	0.69	1.00
25	752.033227	23.96000	0.396	3.060	4.09	0.68	0.84
26	841.053973	0.00140	8.180	1.590	5.76	0.33	0.45
27	859.962313	0.01472	7.989	3.060	4.09	0.68	0.84
28	899.306675	0.00605	7.917	2.985	4.53	0.68	0.90
29	902.616173	0.00426	8.432	2.865	5.10	0.70	0.95
30	906.207325	0.01876	5.111	2.408	4.70	0.70	0.53
31	916.171582	0.83400	1.442	2.670	4.78	0.70	0.78
32^+	923.118427	0.00869	10.220	2.900	5.00	0.70	0.80
33	970.315022	0.89720	1.920	2.550	4.94	0.64	0.67
34	987.926764	13.21000	0.258	2.985	4.55	0.68	0.90
	ν^*	b_1^*	b_2^*	b_3^*	b_4^*	b_5^*	b_6^*
	[GHz]	$[\frac{kHz}{hPa}]$	[1]	$[\frac{MHz}{hPa}]$	[1]	[1]	[1]
	1780.000000	2230.00000	0.952	17.620	30.50	2.00	5.00

Table 3.10: List of used H_2O spectral lines and their spectroscopic coefficients of H_2O in air for the MPM93 model [*Liebe et al.*, 1993]. The last separated line is the unphysical pseudo-line used in MPM93. The lines which are marked with a "+" were not in the MPM87/MPM89 line catalog.

The MPM93 Continuum Parameterization: In the MPM93 version the water vapor continuum is parameterized as an ordinary spectral line (Eqs. (3.60, 3.61)). The parameters of this continuum "pseudo-line" (ν^*, b_1^*, b_2^*, b_3^*, b_4^*, b_5^*, b_6^*) are given in Table 3.10. More details about this continuum parameterization and its microwave approximation can be found in Section 3.2.1 of this guide.

CP98 Water Vapor Absorption Model

Line Absorption component [*Cruz Pol et al.*, 1998] for the water vapor line absorption is based on MPM87 with the main difference that the line catalog consists of only a single line at $\nu_o = 22\,\text{GHz}$. The contributions from the other lines is put into the water vapor continuum module. The line absorption is therefore very quickly calculated (in units of Np/km) according to the formula

$$\alpha_\ell^{\text{CP98}} = 0.0419 \cdot S_0(T) \cdot F(\nu, \nu_k) \tag{3.64}$$

with

$$
\begin{aligned}
S_0(T) &= 0.0109 \cdot C_L \cdot P_{\mathrm{H2O}} \cdot \nu_0 \cdot \Theta^{3.5} \cdot \exp\left(2.143 \cdot [1-\Theta]\right) \\
\gamma &= 0.002784 \cdot C_W \cdot (P_{\mathrm{d}} \cdot \Theta^{0.6} + 4.8 \cdot P_{\mathrm{H2O}} \cdot \Theta^{1.1})
\end{aligned}
\tag{3.65}
$$

where P_{H2O} and P_{d} are the partial pressure of water vapor and dry air in units of hPa, respectively and the Van Vleck-Weisskopf line shape, $F(\nu, \nu_{\mathrm{k}})$. The numbers correspond to the line parameters form MPM87 for this special line and the factors C_L and C_W are adjustable scaling factors to match the model with the measurements. Setting the scaling factors to C_L=1.00 and C_W=1.00 leads to the same results as for MPM87. According to the parameter estimation of Cruz–Pol et al. best agreement between data and model is obtained with $C_L = 1.0639 \pm 0.016$ and $C_W = 1.0658 \pm 0.0096$. The correlation between these two scaling factors was found to be negligible, as can be seen from Table 3.11.

	C_L	C_W	C_C	C_X
value	1.0639	1.0658	1.2369	1.0739
std. dev.	0.016	0.0096	0.155	0.252
correlation				
C_L	1	-0.085	0.045	-0.048
C_W	-0.085	1	-0.513	0.485
C_C	0.045	-0.513	1	-0.989
C_X	-0.048	0.485	-0.989	1

Table 3.11: Scaling parameter values with standard deviation and correlation coefficients according to [*Cruz Pol et al.*, 1998]. The scaling parameters are C_L:22 GHz line strength, C_W:22 GHz line width , C_C:H_2O-continuum, and C_X:O_2-absorption. C_X scales the entire oxygen absorption, the continuum as well as the line absorption. The Cruz-Pol et al. model uses the *Rosenkranz* [1993] oxygen absorption model.

The main reason why the Cruz-Pol model (CP98) considers only one line lies in the fact that CP98 is especially designed for the data analysis in the 20-31.4 GHz region. The determination of the scaling factors was performed with ground based radiometer data in the frequency range of from different locations[6] in the USA.

Water Vapor Continuum Absorption: The CP98 model uses the same water vapor continuum parameterization as MPM87, just scaled with an empirical factor, CC, determined from the above mentioned data:

$$\alpha_{\mathrm{c}}^{\mathrm{CP98}} = C_C \cdot \alpha_{\mathrm{c}}^{\mathrm{MPM87}} \tag{3.66}$$

The scaling factor C_C, as given in Table 3.11, gives a 23.69 % increased continuum absorption compared with MPM87 (see Table 3.4 for a comparison of the parameter values). But one has to keep in mind that C_C has a high correlation with the scaling factor of the oxygen absorption, C_X, since these two components could not be completely distinguished

[6]The data were recorded at San Diego, California (11. December 1991) and West Palm Beach, Florida (8.-21. March 1992)

in the data. Therefore the value of 23.69 % has a standard deviation of 15.5 % and is not so reliable than C_L and C_W.

PWR98 Water Vapor Absorption Model

The water vapor continuum formulation of *Rosenkranz* [1998] is a re-investigation of the existing models MPM87/MPM89, MPM93, and CKD_2.1 especially for the frequency region below 1-1000 GHz. in the context of the available laboratory and atmospheric data [*Bauer et al.*, 1989, 1993, 1995; *Becker and Autler*, 1946; *English et al.*, 1994; *Godon et al.*, 1992; *Liebe*, 1984; *Liebe and Layton*, 1987; *Westwater et al.*, 1980].

Rosenkranz adopted the structure of MPM89 for his improved model (R98). However, some important differences exist compared with MPM89:

- the water vapor line catalogs are different
- the R98 uses the Van Vleck–Weisskopf line shape function with cutoff and MPM89 without cutoff

Water Vapor Line Absorption: The local line absorption is defined as

$$\begin{aligned} \alpha_\ell^{\mathrm{R98}} &= N_{H_2O} \cdot \sum_k S_{\mathrm{k}}(T) \cdot F_c(\nu, \nu_{\mathrm{k}}) \\ &= N_{H_2O} \cdot \sum_k S_{\mathrm{k}}(T) \cdot \left(\frac{\nu}{\nu_{\mathrm{k}}}\right)^2 \cdot [f_c(\nu, +\nu_{\mathrm{k}}) + f_c(\nu, -\nu_{\mathrm{k}})] \quad \text{Np/km} \end{aligned} \tag{3.67}$$

where N_{H_2O} is the number density of water molecules, ν the frequency and S the line intensity, calculated from the HITRAN92 data base *Rothman et al.* [1992]. Considered for this re-investigation are 15 lines with a frequency lower than 1 THz as listed in Table 3.12.

The line shape function $F_c(\nu, \nu_{\mathrm{k}})$ has a cutoff frequency, ν_{cutoff}, and a baseline subtraction similar to the CKD model *Clough et al.* [1989]. The introduction of a cutoff frequency has two advantages: (1) the cutoff avoids applying the line shape to distant frequencies where the line form is theoretically not well understood and (2) the cutoff also establishes a limit to the summation in Eq. (3.67) where lines far away from the cutoff limit do not contribute to the sum. The Rosenkranz formulation uses the same value for the cutoff frequency as the CKD model:

$$\nu_{\mathrm{cutoff}} = 750 \text{ GHz} \tag{3.68}$$

The explicit mathematical form of the line shape function is defined in such a way that in the limit $\nu_{\mathrm{cutoff}} \to \infty$ the combination of Eq. (3.67) with the line shape function would be equivalent to a Van Vleck–Weisskopf [*Van Vleck and Weisskopf*, 1945] line shape:

$$f_c(\nu, \pm\nu_{\mathrm{k}}) = \begin{cases} \dfrac{\gamma_{\mathrm{k}}}{\pi} \left\{ \dfrac{1}{(\nu \mp \nu_{\mathrm{k}})^2 + \gamma_{\mathrm{k}}^2} - \dfrac{1}{\nu_{\mathrm{cutoff}}^2 + \gamma_{\mathrm{k}}^2} \right\} & : \quad |\nu \pm \nu_{\mathrm{k}}| < \nu_{\mathrm{cutoff}} \\ 0 & : \quad |\nu \pm \nu_{\mathrm{k}}| \geq \nu_{\mathrm{cutoff}} \end{cases} \tag{3.69}$$

ν_{k} is the line center frequency and γ_{k} the line half width, which is calculated according to

$$\gamma_{\mathrm{k}} = w_{\mathrm{s,k}} \cdot P_{\mathrm{H2O}} \cdot \Theta^{\mathrm{ns}} + w_{\mathrm{f,k}} \cdot P_{\mathrm{d}} \cdot \Theta^{\mathrm{nf}} \qquad \text{GHz} \tag{3.70}$$

with P_{H2O} and P_{d} as the partial pressure of water vapor and of dry air, respectively. The line depending parameters $w_{\mathrm{s,k}}$, $\mathrm{n_s}$, $w_{\mathrm{f,k}}$, and $\mathrm{n_f}$ are listed in Table 3.12 and the dimensionless parameter Θ is defined as $\Theta = 300\,\mathrm{K}/T$.

Because of the structural similarity to MPM89, the line broadening parameters differ only in minor respects from the values used therein (only the parameters $x_{\mathrm{s},1}$, $w_{\mathrm{f},2}$ and $\mathrm{w_{s,2}}$ are significantly different).

index k	ν_{k} [GHz]	$w_{\mathrm{f,k}}$ [GHz/kPa]	$\mathrm{n_f}$ [1]	$w_{\mathrm{s,k}}$ [GHz/kPa]	$\mathrm{n_s}$ [1]
1	22.2351	0.00281	0.69	0.01349	0.61
2	183.3101	0.00281	0.64	0.01491	0.85
3	321.2256	0.00230	0.67	0.01080	0.54
4	325.1529	0.00278	0.68	0.01350	0.74
5	380.1974	0.00287	0.54	0.01541	0.89
6	439.1508	0.00210	0.63	0.00900	0.52
7	443.0183	0.00186	0.60	0.00788	0.50
8	448.0011	0.00263	0.66	0.01275	0.67
9	470.8890	0.00215	0.66	0.00983	0.65
10	474.6891	0.00236	0.65	0.01095	0.64
11	488.4911	0.00260	0.69	0.01313	0.72
12	556.9360	0.00321	0.69	0.01320	1.00
13	620.7008	0.00244	0.71	0.01140	0.68
14	752.0332	0.00306	0.68	0.01253	0.84
15	916.1712	0.00267	0.70	0.01275	0.78

Table 3.12: Line parameters of the Rosenkranz absorption model (R98) (values taken from *Rosenkranz* [1998]).

Water Vapor Continuum Absorption: The continuum absorption in R98 has the same functional dependence on frequency, pressure, and temperature like in MPM87/MPM89 (see Sec. 3.3.1 for details):

$$\alpha_{\mathrm{c}}^{\mathrm{R98}} = \nu^2 \cdot P_{\mathrm{H2O}} \cdot (C^{\mathrm{o}}_{\mathrm{H_2O}} \cdot P_{\mathrm{H2O}} \cdot \Theta^{\mathrm{ns}} + C^{\mathrm{o}}_{\mathrm{d}} \cdot P_{\mathrm{d}} \cdot \Theta^{\mathrm{nf}}) \tag{3.71}$$

with

$C^{\mathrm{o}}_{\mathrm{H_2O}} = 7.80 \cdot 10^{-8}$ (dB/km) / (hPa·GHz)^2

$\mathrm{n_s} = 7.5$

$C^{\mathrm{o}}_{\mathrm{d}} = 0.236 \cdot 10^{-8}$ (dB/km) / (hPa·GHz)^2

$\mathrm{n_d} = 3.0$

The main difference to the MPM versions are the values of these parameters, since Rosenkranz used additional data to fit his set of parameters. A second point is the cutoff in the line shape of the line absorption calculation. Since this cutoff decreases the line absorption in the window regions, the continuum absorption tends to compensate this decrease to get the same total absorption as withouot cutoff. This effects mainly the parameters $C^{\mathrm{o}}_{\mathrm{H_2O}}$ and $C^{\mathrm{o}}_{\mathrm{d}}$ but has also an influence in the temperature dependence and therefore on $\mathrm{n_s}$ and $\mathrm{n_d}$.

3.3.2 Complete Oxygen Models

Since the Maxwell equations are symmetric in the electric and magnetic fields, electric as well as magnetic dipole transitions are both possible although magnetic dipoles are in general some orders of magnitudes weaker and therefore not relevant in atmospheric radiative transfer models. An exception to this is the complex around 60 GHz of the paramagnetic oxygen magnetic dipole transitions. This bulk of lines arise due to the fact that for rotational quantum numbers $K > 1$ the allowed transitions $\Delta J = \pm 1$ have an energy gap of approximately 60 GHz.
The most frequently used absorption model for this absorption effect is that of Liebe, Rosenkranz, and Hufford *Liebe et al.* [1992] (also reported in *Rosenkranz* [1993] with a slightly different parameterization).

For oxygen – like for water vapor – the total absorption (α_{tot}) is modelled as the line absorption (α_ℓ) plus a continuum absorption (α_{c}):

$$\alpha_{\mathrm{tot}} = \alpha_\ell + \alpha_{\mathrm{c}} \tag{3.72}$$

It has to be emphasized that, α_ℓ and α_{c} of different models are not necessarily compatible and should therefore not be interchanged.

PWR93 Oxygen Absorption Model

Resonant Oxygen Absorption The oxygen absorption model of Rosenkranz is described in *Rosenkranz* [1993]. It is based on the investigations made by Liebe, Rosenkranz, and Hufford *Liebe et al.* [1992]. The FORTRAN77 computer program of Rosenkranz for the O_2 absorption calculation can be downloaded via anonymous ftp from mesa.mit.edu/phil/lbl_rt.

The oxygen line catalog has 40 lines from which 33 lines build the complex around 60 GHz. The parameterization of the line absorption, $\alpha_\ell^{\mathrm{R98}}$, is:

$$\begin{aligned}
\alpha_\ell^{\mathrm{R98}} &= \frac{n_{\mathrm{O_2}}}{\pi} \cdot \sum_{k=1}^{40} S_k(T) \cdot F(\nu, \nu_k) && (3.73)\\
& \quad \text{line intensity:} \\
S_k(T) &= S_k(300\,\mathrm{K}) \;/\; \exp\left(b_k \cdot \Theta\right) && (3.74)\\
& \quad \text{line shape function:} \\
F(\nu,\nu_k) &= \left(\frac{\nu}{\nu_{\mathrm{k}}}\right)^2 \cdot \left[\frac{\Gamma_k + (\nu-\nu_{\mathrm{k}})\cdot Y_k}{(\nu-\nu_k)^2+\Gamma_k^2} + \frac{\Gamma_k - (\nu+\nu_{\mathrm{k}})\cdot Y_k}{(\nu+\nu_k)^2+\Gamma_k^2}\right] \\
& \quad \text{line width:} \\
\Gamma_k &= w_k \cdot \left(P_{\mathrm{d}} \cdot \Theta^{0.8} + 1.1 \cdot P_{\mathrm{H2O}} \cdot \Theta\right) && (3.75)\\
& \quad \text{line coupling:} \\
Y_k &= P_{\mathrm{air}} \cdot \Theta^{0.8} \cdot \left[y_k + (\Theta - 1)\cdot v_k\right] \\
& \quad \text{number density of } \mathrm{O_2}\text{:} \\
n_{\mathrm{O_2}} &= (0.20946 \cdot P_{\mathrm{air}})/(k_B \cdot T)
\end{aligned}$$

where $S_k(300\,\mathrm{K})$ denotes the reference line intensity at T=300 K ant the exponential term approximates the exact partition function. All model parameters (see Refs. *Rosenkranz* [1993] and *Liebe et al.* [1992] for the laboratory measurements and the fitting parameters) are tabulated in Table 3.13.

index k	ν_k [GHz]	$S_k(300\,\mathrm{K})$ [cm^2 Hz]	b_k [1]	w_k [$\frac{\mathrm{MHz}}{\mathrm{hPa}}$]	y_k [$\frac{10^{-3}}{\mathrm{hPa}}$]	v_k [$\frac{10^{-3}}{\mathrm{hPa}}$]
1	118.7503	$.2936\cdot 10^{-14}$	.009	1.63	-0.0233	0.0079
2	56.2648	$.8079\cdot 10^{-15}$	.015	1.646	0.2408	-0.0978
3	62.4863	$.2480\cdot 10^{-14}$	.083	1.468	-0.3486	0.0844
4	58.4466	$.2228\cdot 10^{-14}$	.084	1.449	0.5227	-0.1273
5	60.3061	$.3351\cdot 10^{-14}$	.212	1.382	-0.5430	0.0699
6	59.5910	$.3292\cdot 10^{-14}$	.212	1.360	0.5877	-0.0776
7	59.1642	$.3721\cdot 10^{-14}$	.391	1.319	-0.3970	0.2309
8	60.4348	$.3891\cdot 10^{-14}$	.391	1.297	0.3237	-0.2825
9	58.3239	$.3640\cdot 10^{-14}$	.626	1.266	-0.1348	0.0436
10	61.1506	$.4005\cdot 10^{-14}$	.626	1.248	0.0311	-0.0584
11	57.6125	$.3227\cdot 10^{-14}$	.915	1.221	0.0725	0.6056
12	61.8002	$.3715\cdot 10^{-14}$	.915	1.207	-0.1663	-0.6619
13	56.9682	$.2627\cdot 10^{-14}$	1.260	1.181	0.2832	0.6451
14	62.4112	$.3156\cdot 10^{-14}$	1.260	1.171	-0.3629	-0.6759
15	56.3634	$.1982\cdot 10^{-14}$	1.660	1.144	0.3970	0.6547
16	62.9980	$.2477\cdot 10^{-14}$	1.665	1.139	-0.4599	-0.6675
17	55.7838	$.1391\cdot 10^{-14}$	2.119	1.110	0.4695	0.6135
18	63.5685	$.1808\cdot 10^{-14}$	2.115	1.108	-0.5199	-0.6139
19	55.2214	$.9124\cdot 10^{-15}$	2.624	1.079	0.5187	0.2952
20	64.1278	$.1230\cdot 10^{-14}$	2.625	1.078	-0.5597	-0.2895
21	54.6712	$.5603\cdot 10^{-15}$	3.194	1.05	0.5903	0.2654
22	64.6789	$.7842\cdot 10^{-15}$	3.194	1.05	-0.6246	-0.2590
23	54.1300	$.3228\cdot 10^{-15}$	3.814	1.02	0.6656	0.3750
24	65.2241	$.4689\cdot 10^{-15}$	3.814	1.02	-0.6942	-0.3680
25	53.5957	$.1748\cdot 10^{-15}$	4.484	1.00	0.7086	0.5085
26	65.7648	$.2632\cdot 10^{-15}$	4.484	1.00	-0.7325	-0.5002
27	53.0669	$.8898\cdot 10^{-16}$	5.224	.97	0.7348	0.6206
28	66.3021	$.1389\cdot 10^{-15}$	5.224	.97	-0.7546	-0.6091
29	52.5424	$.4264\cdot 10^{-16}$	6.004	.94	0.7702	0.6526
30	66.8368	$.6899\cdot 10^{-16}$	6.004	.94	-0.7864	-0.6393
31	52.0214	$.1924\cdot 10^{-16}$	6.844	.92	0.8083	0.6640
32	67.3696	$.3229\cdot 10^{-16}$	6.844	.92	-0.8210	-0.6475
33	51.5034	$.8191\cdot 10^{-17}$	7.744	.89	0.8439	0.6729
34	67.9009	$.1423\cdot 10^{-16}$	7.744	.89	-0.8529	-0.6545
35	368.4984	$.6460\cdot 10^{-15}$	.048	1.92	0.0000	0.0000
36	424.7631	$.7047\cdot 10^{-14}$	.044	1.92	0.0000	0.0000
37	487.2494	$.3011\cdot 10^{-14}$	.049	1.92	0.0000	0.0000

Table 3.13: (continued)

index	ν_k	$S_k(300\,\mathrm{K})$	b_k	w_k	y_k	v_k
38	715.3932	$.1826\cdot 10^{-14}$	.145	1.81	0.0000	0.0000
39	773.8397	$.1152\cdot 10^{-13}$	.141	1.81	0.0000	0.0000
40	834.1453	$.3971\cdot 10^{-14}$	.145	1.81	0.0000	0.0000

Table 3.13: List of O_2 spectral lines of the Rosenkranz absorption model *Rosenkranz* [1993].

Oxygen Continuum Absorption: As pointed out by Van Vleck *Van Vleck* [1987], the standard theory for non-resonant absorption is that of Debye (see also Ref. *Townes and Schawlow* [1955]). The Debye line shape is obtained from the VVW line shape function by the limiting case $\nu_\mathrm{k} \to 0$. Rosenkranz *Rosenkranz* [1993] adopt the Debye theory for his models:

$$\alpha_\mathrm{c} = C \cdot P_\mathrm{d} \cdot \Theta^2 \cdot \frac{\nu^2 \cdot \gamma}{\nu^2 + \gamma^2} \tag{3.76}$$

$$\gamma = w \cdot (P_\mathrm{d} \cdot \Theta^{0.8} + 1.1 \cdot P_\mathrm{H2O} \cdot \Theta) \tag{3.77}$$

The values for the parameters are $C = 1.11 \cdot 10^{-5}$ dB/km/(hPa GHz) and $w = 5.6 \cdot 10^{-4}$ GHz/hPa, respectively. This absorption term is proportional to the collision frequency of a single oxygen molecule and thus proportional to the dry air pressure[7].

MPM93 Oxygen Absorption Model

Oxygen Line Absorption: The oxygen line catalog has 44 lines from which 37 lines build the complex around 60 GHz [*Liebe et al.*, 1993]. The parameterization of the line absorption, α_ℓ^MPM, is (in units of dB/km):

$$\alpha_\ell^\mathrm{MPM} = 0.1820 \cdot \nu^2 \cdot \sum_{k=1}^{44} S_k(T) \cdot F(\nu, \nu_k) \quad \text{dB/km} \tag{3.78}$$

with

line intensity:

$$S_k(T) = \frac{a_{1,k}}{\nu_\mathrm{k}} \cdot P_\mathrm{d} \cdot \Theta^3 \cdot \exp\left[a_{2,k} \cdot (1 - \Theta)\right] \tag{3.79}$$

line shape function:

$$F(\nu, \nu_k) = \left[\frac{\gamma_k + (\nu - \nu_\mathrm{k}) \cdot \delta_k}{(\nu - \nu_k)^2 + \gamma_k^2} + \frac{\gamma_k - (\nu + \nu_\mathrm{k}) \cdot \delta_k}{(\nu + \nu_k)^2 + \gamma_k^2} \right]$$

line width:

$$\gamma_k = a_{3,k} \cdot 10^{-3} \cdot (P_\mathrm{d} \cdot \Theta^{a_{4,k}} + 1.10 \cdot P_\mathrm{H2O} \cdot \Theta) \tag{3.80}$$

line coupling:

$$\delta_k = P_\mathrm{air} \cdot \Theta^{0.8} \cdot [a_{5,k} + \Theta \cdot a_{6,k}]$$

[7] The absorption due to weakly bound complexes of O_2–X with $X = H_2O$, N_2 is treated separately and therefore not included in this Debye formula.

where $a_{1-5,k}$ are the fitted parameters due to laboratory measurements *Liebe et al.* [1992]. All model parameters are tabulated in Table 3.14. One has to note that in the MPM93 code is a threshold value for $\alpha_\ell^{\mathrm{MPM}}$ implemented:

$$\alpha_\ell^{\mathrm{MPM}} = \begin{cases} \alpha_\ell^{\mathrm{MPM}} & : \quad \alpha_\ell^{\mathrm{MPM}} > 0 \\ 0 & : \quad \alpha_\ell^{\mathrm{MPM}} < 0 \end{cases} \tag{3.81}$$

Therefore the oxygen absorption in the wings of the strong O_2-lines is remarkably higher than in the R93 model.

index	ν_k	$a_{1,k}$	$a_{2,k}$	$a_{3,k}$	$a_{4,k}$	$a_{5,k}$	$a_{6,k}$
k	[GHz]	[$\frac{\mathrm{kHz}}{\mathrm{hPa}}$]	[1]	[$\frac{\mathrm{MHz}}{\mathrm{hPa}}$]	[1]	[$\frac{10^3}{\mathrm{hPa}}$]	[$\frac{10^3}{\mathrm{hPa}}$]
1	50.474238	0.094	9.694	0.890	0.0	0.240	0.790
2	50.987749	0.246	8.694	0.910	0.0	0.220	0.780
3	51.503350	0.608	7.744	0.940	0.0	0.197	0.774
4	52.021410	1.414	6.844	0.970	0.0	0.166	0.764
5	52.542394	3.102	6.004	0.990	0.0	0.136	0.751
6	53.066907	6.410	5.224	1.020	0.0	0.131	0.714
7	53.595749	12.470	4.484	1.050	0.0	0.230	0.584
8	54.130000	22.800	3.814	1.070	0.0	0.335	0.431
9	54.671159	39.180	3.194	1.100	0.0	0.374	0.305
10	55.221367	63.160	2.624	1.130	0.0	0.258	0.339
11	55.783802	95.350	2.119	1.170	0.0	-0.166	0.705
12	56.264775	54.890	0.015	1.730	0.0	0.390	-0.113
13	56.363389	134.400	1.660	1.200	0.0	-0.297	0.753
14	56.968206	176.300	1.260	1.240	0.0	-0.416	0.742
15	57.612484	214.100	0.915	1.280	0.0	-0.613	0.697
16	58.323877	238.600	0.626	1.330	0.0	-0.205	0.051
17	58.446590	145.700	0.084	1.520	0.0	0.748	-0.146
18	59.164207	240.400	0.391	1.390	0.0	-0.722	0.266
19	59.590983	211.200	0.212	1.430	0.0	0.765	-0.090
20	60.306061	212.400	0.212	1.450	0.0	-0.705	0.081
21	60.434776	246.100	0.391	1.360	0.0	0.697	-0.324
22	61.150560	250.400	0.626	1.310	0.0	0.104	-0.067
23	61.800154	229.800	0.915	1.270	0.0	0.570	-0.761
24	62.411215	193.300	1.260	1.230	0.0	0.360	-0.777
25	62.486260	151.700	0.083	1.540	0.0	-0.498	0.097
26	62.997977	150.300	1.665	1.200	0.0	0.239	-0.768
27	63.568518	108.700	2.115	1.170	0.0	0.108	-0.706
28	64.127767	73.350	2.620	1.130	0.0	-0.311	-0.332
29	64.678903	46.350	3.195	1.100	0.0	-0.421	-0.298
30	65.224071	27.480	3.815	1.070	0.0	-0.375	-0.423
31	65.764772	15.300	4.485	1.050	0.0	-0.267	-0.575
32	66.302091	8.009	5.225	1.020	0.0	-0.168	-0.700
33	66.836830	3.946	6.005	0.990	0.0	-0.169	-0.735
34	67.369598	1.832	6.845	0.970	0.0	-0.200	-0.744

Table 3.14: (continued)

index	ν_k	$a_{1,k}$	$a_{2,k}$	$a_{3,k}$	$a_{4,k}$	$a_{5,k}$	$a_{6,k}$
35	67.900867	0.801	7.745	0.940	0.0	-0.228	-0.753
36	68.431005	0.330	8.695	0.920	0.0	-0.240	-0.760
37	68.960311	0.128	9.695	0.900	0.0	-0.250	-0.765
38	118.750343	94.500	0.009	1.630	0.0	-0.036	0.009
39	368.498350	6.790	0.049	1.920	0.6	0.000	0.000
40	424.763124	63.800	0.044	1.930	0.6	0.000	0.000
41	487.249370	23.500	0.049	1.920	0.6	0.000	0.000
42	715.393150	9.960	0.145	1.810	0.6	0.000	0.000
43	773.839675	67.100	0.130	1.820	0.6	0.000	0.000
44	834.145330	18.000	0.147	1.810	0.6	0.000	0.000

Table 3.14: List of O_2 spectral lines of the MPM93 absorption model *Liebe et al.* [1993].

Oxygen Continuum Absorption: As pointed out by Van Vleck *Van Vleck* [1987], the standard theory for non-resonant absorption is that of Debye (see also Ref. *Townes and Schawlow* [1955]). The Debye line shape is obtained from the VVW line shape function by the limiting case $\nu_{\rm k} \to 0$. *Liebe et al.* [1993] adopt the Debye theory for his model:

$$\begin{aligned} \alpha_{\rm c} &= C \cdot P_{\rm d} \cdot \Theta^2 \cdot \frac{\nu^2 \cdot \gamma}{\nu^2 + \gamma^2} \\ \gamma &= w \cdot P_{\rm tot} \cdot \Theta^{0.8} \end{aligned} \tag{3.82}$$

The values for the parameters are $C = 1.11 \cdot 10^{-5}$ dB/km/(hPa GHz) and $w = 5.6 \cdot 10^{-4}$ GHz/hPa, respectively. This absorption term is proportional to the collision frequency of a single oxygen molecule and thus proportional to the dry air pressure[8].

3.3.3 ARTS Workspace Variables and Methods

This section explains how the above described full models (continuum+lines) are represented in the structure of the arts source code and how one can invoke them in the arts control file.

The full model tags need more input specification than normal trace gas tags. Why this is so can be seen from Eq. 3.27 and Table 3.4. For a single function for the water vapor continuum we find several different function parameters in the literature. To solve this ambiguity arts has two methods implemented which helps the user to select a single set of parameters in an easy way. In connection with this input parameters we distinguish generally two types, the referenced models which are taken from the literature (e. g. *Liebe et al.* [1993] or *Rosenkranz* [1993]) and the user model, for which the arts user is providing the necessary parameter values.

After selecting the continuum tag with the `tagDefine` method, the arts user has to setup the arts internal structure (i. e. the workspace variables *cont_description_names,*

[8] The absorption due to weakly bound complexes of O_2–X with $X = H_2O$, N_2 is treated separately and therefore not included in this Debye formula.

cont_description_models, and cont_description_parameters) for the selected continuum tags, which can simply be done by putting the following line into the arts control file:

```
cont_descriptionInit{}
```

After this initialization, the continuum tag specific information has to be transfered to arts. This is possible with the arts method *cont_descriptionAppend*, which has itself three input variables: *tagname*, *model*, and *userparameters*. The user has to specify these input variables in the arts control file for each selected continuum tag. Below is a list of all the implemented continuum tags and the associated valid range of the input variables for *cont_descriptionAppend*. For a condensed overview of the possible continuum tags and their referenced models see Table 3.15 and the online documentation can be found under *arts/doc/doxygen/html/continua_cc.html*.

One has to note at this place that the two input variables *model* and *userparameters* are to some extend redundant. Therefore one can also produce an ambiguity by giving contradicting values for these two input variables. To avoid such ambiguities the arts user should keep in mind the general rule that only the user model (*model ="user"*) needs input parameters via the input variable *userparameters*. All the referenced models need no input via *userparameters*. If you try to run the arts control file with a referenced model and input parameters you will get an error message. Below in the detailed description of *cont_descriptionAppend* you can find correct examples for all the continuum tags.

- The water vapor model of MPM87 [*Liebe and Layton*, 1987] has the arts tag name `"H2O-MPM87"`. The details about this water vapor absorption model are described in Section 3.3.1. The standard way to use the full (=continuum+lines) MPM87 water vapor absorption model is to set the input variable *model* to "MPM87" and leaving the input parameter *userparameters* empty. It might be necessary in some cases to use only the line or the continuum absorption part of MPM87. This can be easily done by setting *model* to "MPM87Lines" or "MPM87Continuum", respectively (leaving the input parameter *userparameters* empty too).
  To have a minimum possibility of variation for MPM87, arts allows to run MPM87 also with *model*="user". In this case the user has to provide three scaling factors, CC, CL, and CW, with the input variable *userparameters*, *userparameters* = $[CC$, CL, $CW]$. Each line intensity $S_{\mathrm{k}}(T)$ (see Eq. (3.47)) is multiplied with the scaling factor CL, while CW scales each line width, γ_{k}, (see Eq. (3.50)). The continuum absorption, $\alpha_{\mathrm{c}}^{\mathrm{MPM87}}$, (see Eq. (3.51)) also scales with CC.
  In the following all the valid possibilities for the tag `"H2O-MPM87"` are listed (the values for the model user are just example values):

  ```
  cont_descriptionAppend{
      name           = "H2O-MPM87"
      model          = "MPM87"
      userparameters = [ ]
  }
  cont_descriptionAppend{
      name           = "H2O-MPM87"
      model          = "MPM87Lines"
  ```

```
    userparameters = [ ]
}
cont_descriptionAppend{
    name           = "H2O-MPM87"
    model          = "MPM87Continuum"
    userparameters = [ ]
}
cont_descriptionAppend{
    name           = "H2O-MPM87"
    model          = "user"
    userparameters = [ 1.0, 1.0, 1.0 ]
}
```

- The full water vapor absorption model MPM89 *Liebe* [1989] has the arts tag name `"H2O-MPM89"`. The details about this water vapor absorption model are described in Section 3.3.1. The standard way to use the full (=continuum+lines) MPM87 water vapor absorption model is to set the input variable *model* to "MPM89" and leaving the input parameter *userparameters* empty. It might be necessary in some cases to use only the line or the continuum absorption part of MPM89. This can be easily done by setting *model* to "MPM89Lines" or "MPM89Continuum", respectively (leaving the input parameter *userparameters* empty too).
  To have a minimum possibility of variation for MPM89, arts allows to run MPM89 also with *model* = "user". In this case the user has to provide three scaling factors, CC, CL, and CW, with the input variable *userparameters*, *userparameters* = $[CC$, CL, $CW]$. Each line intensity $S_{\mathrm{k}}(T)$ (see Eq. (3.54)) is multiplied with the scaling factor CL, while CW scales each line width, γ_{k}, (see Eq. (3.56)). The continuum absorption, $\alpha_{\mathrm{c}}^{\mathrm{MPM89}}$, (see Eq. (3.57)) also scales with CC.
  In the following all the valid possibilities for the tag `"H2O-MPM89"` are listed (the values for the model user are just example values):

```
cont_descriptionAppend{
    name           = "H2O-MPM89"
    model          = "MPM89"
    userparameters = [ ]
}
cont_descriptionAppend{
    name           = "H2O-MPM89"
    model          = "MPM89Lines"
    userparameters = [ ]
}
cont_descriptionAppend{
    name           = "H2O-MPM89"
    model          = "MPM89Continuum"
    userparameters = [ ]
}
cont_descriptionAppend{
```

```
    name           = "H2O-MPM89"
    model          = "user"
    userparameters = [ 1.0, 1.0, 1.0 ]
}
```

- The water vapor model of MPM93 *Liebe et al.* [1993] has the arts tag name H2O-MPM93. The details about this water vapor absorption model are described in Section 3.3.1. The standard way to use the full (=continuum+lines) MPM93 water vapor absorption model is to set the input variable *model* to "MPM93" and leaving the input parameter *userparameters* empty. It might be necessary in some cases to use only the line or the continuum absorption part of MPM93. This can be easily done by setting *model* to "MPM93Lines" or "MPM93Continuum", respectively (leaving the input parameter *userparameters* empty too).
  To have a minimum possibility of variation for MPM93, arts allows to run MPM93 also with *model* = "user". In this case the user has to provide three scaling factors, CC, CL, and CW, with the input variable *userparameters*, *userparameters* = $[CC$, CL, $CW]$. Each line intensity $S_{\text{k}}(T)$ (see Eq. (3.60)) is multiplied with the scaling factor CL, while CW scales each line width, γ_{k}, (see Eq. (3.63)). The continuum absorption, $\alpha_{\text{c}}^{\text{MPM93}}$, (see Eq. (3.60)) also scales with CC.
  In the following all the valid possibilities for the tag "H2O-MPM93" are listed (the values for the model user are just example values):

```
cont_descriptionAppend{
    name           = "H2O-MPM93"
    model          = "MPM93"
    userparameters = [ ]
}
cont_descriptionAppend{
    name           = "H2O-MPM93"
    model          = "MPM93Lines"
    userparameters = [ ]
}
cont_descriptionAppend{
    name           = "H2O-MPM93"
    model          = "MPM93Continuum"
    userparameters = [ ]
}
cont_descriptionAppend{
    name           = "H2O-MPM93"
    model          = "user"
    userparameters = [ 1.0, 1.0, 1.0 ]
}
```

- The water vapor model of CP98 [*Cruz Pol et al.*, 1998] has the arts tag name "H2O-CP98". The details about this water vapor absorption model are described in Section 3.3.1. The standard way to use the full (=continuum+lines) CP98 water

vapor absorption model is to set the input variable *model* to "CP98" and leaving the input parameter *userparameters* empty. It might be necessary in some cases to use only the line or the continuum absorption part of CP98. This can be easily done by setting *model* to "CruzPolLines" or "CruzPolContinuum", respectively (leaving the input parameter *userparameters* empty too).
To have a minimum possibility of variation for CP98, arts allows to run CP98 also with *model* = "user". In this case the user has to provide three scaling factors, CC, CL, and CW, with the input variable *userparameters*, *userparameters* = $[CC$, CL, $CW]$. Each line intensity $S_{\mathrm{k}}(T)$ (see Eq. (3.65)) is multiplied with the scaling factor CL, while CW scales each line width, γ_{k}, (see Eq. (3.65)). The continuum absorption, $\alpha_{\mathrm{c}}^{\mathrm{CP98}}$, (see Eq. (3.66)) also scales with CC.
In the following all the valid possibilities for the tag `"H2O-CP98"` are listed (the values for the model user are just example values):

```
cont_descriptionAppend{
    name           = "H2O-CP98"
    model          = "CruzPol"
    userparameters = [ ]
}
cont_descriptionAppend{
    name           = "H2O-CP98"
    model          = "CruzPolLines"
    userparameters = [ ]
}
cont_descriptionAppend{
    name           = "H2O-CP98"
    model          = "CruzPolContinuum"
    userparameters = [ ]
}
cont_descriptionAppend{
    name           = "H2O-CP98"
    model          = "user"
    userparameters = [ 1.0, 1.0, 1.0 ]
}
```

- The water vapor model of PWR98 [*Rosenkranz*, 1998] has the arts tag name `"H2O-PWR98"`. The details about this water vapor absorption model are described in Section 3.3.1. The standard way to use the full (=continuum+lines) CP98 water vapor absorption model is to set the input variable *model* to "Rosenkranz" and leaving the input parameter *userparameters* empty. It might be necessary in some cases to use only the line or the continuum absorption part of PWR98. This can be easily done by setting *model* to "RosenkranzLines" or "RosenkranzContinuum", respectively (leaving the input parameter *userparameters* empty too).
  To have a minimum possibility of variation for CP98, arts allows to run PWR98 also with *model* = "user". In this case the user has to provide three scaling factors, CC, CL, and CW, with the input variable *userparameters*, *userparameters* = $[CC$, CL,

CW]. Each line intensity $S_{\mathrm{k}}(T)$ (see Eq. (3.67)) is multiplied with the scaling factor CL, while CW scales each line width, γ_{k}, (see Eq. (3.70)). The continuum absorption, $\alpha_{\mathrm{c}}^{\mathrm{R98}}$, (see Eq. (3.71)) also scales with CC.
In the following all the valid possibilities for the tag "H2O-PWR98" are listed (the values for the model user are just example values):

```
cont_descriptionAppend{
    name           = "H2O-PWR98"
    model          = "Rosenkranz"
    userparameters = [ ]
}
cont_descriptionAppend{
    name           = "H2O-PWR98"
    model          = "RosenkranzLines"
    userparameters = [ ]
}
cont_descriptionAppend{
    name           = "H2O-PWR98"
    model          = "RosenkranzContinuum"
    userparameters = [ ]
}
cont_descriptionAppend{
    name           = "H2O-PWR98"
    model          = "user"
    userparameters = [ 1.0, 1.0, 1.0 ]
}
```

- The MPM93 full absorption model for oxygen [*Liebe et al.*, 1993] has the arts tag name "O2-MPM93". The details about this oxygen absorption model are described in Section 3.3.2. The standard way to use the full (=continuum+lines) MPM93 oxygen absorption model is to set the input variable *model* to "MPM93" and leaving the input parameter *userparameters* empty. It might be necessary in some cases to use only the line or the continuum absorption part of MPM93. This can be easily done by setting *model* to "MPM93Lines" or "MPM93Continuum", respectively (leaving the input parameter *userparameters* empty too).
  To have a minimum possibility of variation for MPM93, arts allows to run MPM93 also with *model*="user". In this case the user has to provide four scaling factors, CC, CL, CW, and CO, with the input variable *userparameters*, e. g. *userparameters*=$[CC, CL, CW, CO]$. Each line intensity $S_{\mathrm{k}}(T)$ (see Eq. (3.79)) is multiplied with the scaling factor CL, while CW scales each line width, γ_{k}, (see Eq. (3.80)) and CO the line coupling parameter (see Eq. (3.81)). The continuum absorption, (see Eq. (3.82)) also scales with CC.
  In the following all the valid possibilities for the tag "O2-MPM93" are listed (the values for the model user are just example values):

```
cont_descriptionAppend{
```

```
    name           = "O2-MPM93"
    model          = "MPM93"
    userparameters = [ ]
}
cont_descriptionAppend{
    name           = "O2-MPM93"
    model          = "MPM93Lines"
    userparameters = [ ]
}
cont_descriptionAppend{
    name           = "O2-MPM93"
    model          = "MPM93Continuum"
    userparameters = [ ]
}
cont_descriptionAppend{
    name           = "O2-MPM93"
    model          = "user"
    userparameters = [ 1.0, 1.0, 1.0, 1.0 ]
}
```

- The PWR93 full absorption model for oxygen [*Rosenkranz*, 1993] has the arts tag name `"O2-PWR93"`. The details about this oxygen absorption model are described in Section 3.3.2. The standard way to use the full (=continuum+lines) PWR93 oxygen absorption model is to set the input variable *model* to "Rosenkranz" and leaving the input parameter *userparameters* empty. It might be necessary in some cases to use only the line or the continuum absorption part of PWR93. This can be easily done by setting *model* to "RosenkranzLines" or "RosenkranzContinuum", respectively (leaving the input parameter *userparameters* empty too).
  To have a minimum possibility of variation for PWR93, arts allows to run PWR93 also with *model*="user". In this case the user has to provide four scaling factors, CC, CL, CW, and CO, with the input variable *userparameters*, e. g. *userparameters*=$[CC, CL, CW, CO]$. Each line intensity $S_{\mathrm{k}}(T)$ (see Eq. (3.74)) is multiplied with the scaling factor CL, while CW scales each line width, γ_{k}, (see Eq. (3.75)) and CO the line coupling parameter (see Eq. 3.76)). The continuum absorption, (see Eq. (3.76)) also scales with CC.
  In the following all the valid possibilities for the tag `"O2-PWR93"` are listed (the values for the model user are just example values):

```
cont_descriptionAppend{
    name           = "O2-PWR93"
    model          = "Rosenkranz"
    userparameters = [ ]
}
cont_descriptionAppend{
    name           = "O2-PWR93"
    model          = "RosenkranzLines"
```

```
    userparameters = [ ]
}
cont_descriptionAppend{
    name           = "O2-PWR93"
    model          = "RosenkranzContinuum"
    userparameters = [ ]
}
cont_descriptionAppend{
    name           = "O2-PWR93"
    model          = "user"
    userparameters = [ 1.0, 1.0, 1.0, 1.0]
}
```

continuum	*cont_descriptionAppend* input input parameter			reference/ arts uguide	arts source code function
			water vapor (H_2O)		
Rosenkranz	tagname	=	`"H2O-PWR98"`	*Rosenkranz* [1998]	PWR98H2OAbsModel
	model	=	"Rosenkranz"		
	userparameters	=	[ ]		
Cruz-Pol	tagname	=	`"H2O-CP98"`	*Cruz Pol et al.* [1998]	CP98H2OAbsModel
	model	=	"CruzPol"		
	userparameters	=	[ ]		
MPM97	tagname	=	`"H2O-MPM87"`	*Liebe and Layton* [1987]	MPM87H2OAbsModel
	model	=	"MPM93"		
	userparameters	=	[ ]		
MPM89	tagname	=	`"H2O-MPM89"`	*Liebe* [1989]	MPM89H2OAbsModel
	model	=	"MPM93"		
	userparameters	=	[ ]		
MPM93	tagname	=	`"H2O-MPM93"`	*Liebe et al.* [1993]	MPM93H2OAbsModel
	model	=	"MPM93"		
	userparameters	=	[ ]		
			oxygen (O_2)		
Rosenkranz	tagname	=	`"O2-SelfContPWR93"`	*Rosenkranz* [1993]	PWR93O2AbsModel
	model	=	"Rosenkranz"		
	userparameters	=	[ ]		
MPM93	tagname	=	`"O2-SelfContMPM93"`	*Liebe et al.* [1993]	MPM93O2AbsModel
	model	=	"MPM93"		
	userparameters	=	[ ]		

Table 3.15: This table gives an overview of the implemented referenced full (continua+line) absorption models and how they are specified in the arts method *cont_descriptionAppend*. Additionally the reference and the arts source code function names (see file *arts/src/continua.cc* are provided. The detailed online documentation can be found under *arts/doc/doxygen/html/continua_cc.html*).

ARTS Example Control File for the Full Model Tags

Below you will find an example of a control file for all the implemented fixed full models to calculate line+continuum absorption of water vapor and oxygen. Please note that to run this example control file you have to specify user specific paths and input file names to run it properly. You can find this example in the arts directory *arts/doc/examples/fullmodels_example.arts*

```
# ######## EXAMPLE FULL MODEL TAG CONTROL FILE #######
# define the arts full model tags of arts
tgsDefine{
      [
        "N2-SelfContStandardType",
        "H2O-CP98",
        "H2O-MPM87",
        "H2O-MPM89",
        "H2O-MPM93",
        "H2O-PWR98",
        "O2-PWR93",
        "O2-MPM93"
      ]
}
#
#
#-----------------------------------------------------
#
#
cont_descriptionInit{}
#
#
# ----- H2O full models (line+continuum) ------------
#
# Cruz-Pol 1998 H2O absorption model (line + continuum)
cont_descriptionAppend{
    tagname        = "H2O-CP98"
    model         = "CruzPol"
    userparameters = [ ]
}
#
# Rosenkranz 1998 H2O absorption model (lines + continuum)
cont_descriptionAppend{
    tagname        = "H2O-PWR98"
    model         = "Rosenkranz"
    userparameters = [ ]
}
#
# MPM87 H2O absorption model (lines + continuum)
cont_descriptionAppend{
    tagname        = "H2O-MPM87"
    model         = "MPM87"
    userparameters = [ ]
}
#
# MPM89 H2O absorption model (lines + continuum)
cont_descriptionAppend{
    tagname        = "H2O-MPM89"
    model          = "MPM89"
    userparameters = [ ]
}
#
# MPM93 H2O absorption model (lines + continuum)
cont_descriptionAppend{
    tagname        = "H2O-MPM93"
    model          = "MPM93"
    userparameters = [ ]
}
#
#
#
# ----- O2 full models (line+continuum) -------------
#
# Rosenkranz O2 absorption model (line + continuum):
cont_descriptionAppend{
    tagname        = "O2-PWR93"
    model         = "RosenkranzContinuum"
    userparameters = [ ]
}
#
```

full model tag selection for calculation.

initialize the continua tag description structure in arts.
This is essential for the later use of the method cont_descriptionAppend.

description of every full model tag also mentioned in the tagDefine methode above. Each description has three input variables:

* tag name
* model to select a referenced model or the user model
* user given input parameters (only valid for model "user" otherwise leave it blank)

Only in the case where the model "user" is selected, the user given input parameters are considered. All other models neglect these input parameters.

```
# MPM93 O2 absorption model (lines + continuum)
cont_descriptionAppend{
    tagname        = "O2-MPM93"
    model         = "MPM93Continuum"
    userparameters = [ ]
}
#
# ----- N2 continuum --------------------------------
# One has to provide a N2 tag for method absCalc that's
# the only reason why we specify here the N2 continuum tag
# Rosenkranz N2-N2 continuum (only N2-N2 broadening):
cont_descriptionAppend{
    tagname        = "N2-SelfContStandardType"
    model          = "Rosenkranz"
    userparameters = [ ]
}
#
#-----------------------------------------------------
#
# Read the pressure, temperature, and altitude
# profiles and create the workspace variable 'raw_ptz'.
# ATTENTION! THE PATH AND FILE NAMES ARE USER SPECIFIC!
MatrixReadAscii (raw_ptz)
        {"@ac_arts_data@/atmosphere/fascod/midlatitude-summer.tz.aa"}
#
# The same for the input VMR profiles
# ATTENTION! THE PATH AND FILE NAMES ARE USER SPECIFIC!
raw_vmrsReadFromScenario
         {"@ac_arts_data@/atmosphere/fascod/midlatitude-summer"}
#
# Create the pressure grid 'p_abs' (just an example)
VectorNLogSpace(p_abs){
        start = 100000.000
        stop  =   1000.000
        n     = 100
}
# reads the input profiles
AtmFromRaw{}
#
#-----------------------------------------------------
#
# Set the H2O profile
h2o_absSet{}
#
# Set the N2 profile
n2_absSet{}
#
#-----------------------------------------------------
#
# Read spectral line data from HITRAN96 catalogue for
# the frequency range from 1 to 2 GHz.
# This in not essential for the continuum tags but
# bust be given as input for absCalc below.
# ATTENTION! THE PATH AND FILE NAMES ARE USER SPECIFIC!
lines_per_tgReadFromCatalogues{
  filenames = [ "@ac_arts_data@/spectroscopy/hitran96/hitran96_lowfreq.par" ]
  formats   = [ "HITRAN96" ]
  fmin      = [ 1.0e9 ]
  fmax      = [ 2.0e9 ]
}
# Create an example frequency grid 'f_mono'
VectorNLinSpace(f_mono){
        start =    100.0e9
        stop  =    200.0e9
        n     =    100
}
#
#-----------------------------------------------------
#
#
```

Information about the model atmosphere. Also the VMR profiles H2O and N2 have to be given seperately.

Spectral line data is also necessary for the method absCalc.

Input frequency grid on which the calculation is performed.

```
# Set the lineshape function for each continuum tag
lineshape_per_tgDefine{
        shape               = [ "no_shape",
                                "no_shape",
                                "no_shape",
                                "no_shape",
                                "no_shape",
                                "no_shape",
                                "no_shape",
                                "no_shape"]
        normalizationfactor = [ "no_norm",
                                "no_norm",
                                "no_norm",
                                "no_norm",
                                "no_norm",
                                "no_norm",
                                "no_norm",
                                "no_norm"]
        cutoff              = [ -1,
                                -1,
                                -1,
                                -1,
                                -1,
                                -1,
                                -1,
                                -1]
}
#
#----------------------------------------------------
#
# calculate the absorption coefficients, unit=1/meter
absCalc{}
#
#----------------------------------------------------
#
# These we definitely want to write to files:
# 1. absorption coefficient per continuum tag
ArrayOfMatrixWriteAscii (abs_per_tg) {""}
# 2. temperature profile
VectorWriteAscii (t_abs)  {""}
# 3. altitude grid
VectorWriteAscii (z_abs)  {""}
# 4. pressure grid
VectorWriteAscii(p_abs)   {""}
# 5. frequency grid
VectorWriteAscii (f_mono) {""}
# 6. cont_descriptionAppend continuum tagnames
ArrayOfStringWriteAscii (cont_description_names) {""}
# 7. cont_descriptionAppend model selections
ArrayOfStringWriteAscii (cont_description_models) {""}
# 8. cont_descriptionAppend user given input parameters
ArrayOfVectorWriteAscii (cont_description_parameters) {""}
###########################################################
```

The line shape of the full model tags are all internally set. Therefore The user has not to specify the line shape here.

This is the method which calculates the absorption coefficients in units of 1/meters.

Here the output is written into the output files.

Chapter 4

Cloud Absorption

4.1 Liquid water and ice particle absorption

So far only absorption due to air was described. However hydrometeors[1] can have a noticeable effect on the radiative transfer through the atmosphere in the 10-30 GHz frequency range.

The MPM93 model provides beside the absorption model of air also an absorption model for suspended liquid water droplets and ice particles [*Liebe et al.*, 1989, 1991; *Hufford*, 1991; *Liebe et al.*, 1993]. The model is applicable for the Rayleigh regime, for which the relation $r < 0.05 \cdot \lambda$ holds where r is the particle radius and λ is the wavelength[2], e. g. for a frequency of around 22 GHz this means $r < 500\,\mu$m. Considering *Salby* [1996], this criterium is – except for cirrus – nearly for every aerosol and cloud class satisfied. But one has to bear in mind that these values have a wide range of variability, for example, *Salby* [1996] states that the mean particle radius for stratus, cumulus, and nimbus clouds can be in the range of 10-1000 μm and that the particle radius distribution is highly unsymmetric.

With respect to the imaginary part of the complex refractivity, a unified parameterization of liquid and ice particle absorption is formulated in MPM93:

$$\begin{aligned} \alpha &= 0.1820 \cdot \nu \cdot N'' \qquad\qquad \text{dB/km} \\ N'' &= \frac{3}{2} \cdot \frac{w}{m} \cdot \Im[(\epsilon_r - 1)/(\epsilon_r + 2)] \\ N'' &= \frac{3}{2} \cdot \frac{w}{m} \cdot \left[\frac{3 \cdot \epsilon_r^{''}}{(\epsilon_r^{'} + 2)^2 \; + \; (\epsilon_r^{''})^2} \right] \end{aligned} \tag{4.1}$$

where w is the liquid water (0.0$< LWC <$5.0 g/m^3) or ice mass (0.0 IWC 1.0 g/m^3) content and m is the water or ice bulk density ($\rho_{\rm l,i}$=1.0 g/cm^3 and 0.916 g/cm^3, respectively). The difference between liquid water and ice absorption is put in the expressions for the complex permittivities (i. e. the relative dielectric constant), $\epsilon_r = \epsilon_r^{'} + i \cdot \epsilon_r^{''}$, which depend on frequency and temperature.

• Complex permittivity for suspended liquid water droplets:

$$\epsilon_r^{'} \;=\; \epsilon_o \;-\; \nu^2 \cdot \left[\frac{\epsilon_o - \epsilon_1}{\nu^2 + \gamma_1^2} \;+\; \frac{\epsilon_1 - \epsilon_2}{\nu^2 + \gamma_2^2} \right]$$

[1] We denote liquid water and ice particles, either suspended or precipitating, in the air as hydrometeors.

[2] See *Brussaard and Watson* [1995], page 81, for details.

$$\begin{aligned}
\epsilon_r'' &= \nu \cdot \left[\gamma_1 \cdot \frac{\epsilon_o - \epsilon_1}{\nu^2 + \gamma_1^2} + \gamma_2 \cdot \frac{\epsilon_1 - \epsilon_2}{\nu^2 + \gamma_2^2}\right] \\
\epsilon_o &= 77.66 + 103.3 \cdot (\Theta - 1) \qquad (4.2) \\
\epsilon_1 &= 0.0671 \cdot \epsilon_o \\
\epsilon_2 &= 3.52 \\
\gamma_1 &= 20.20 - 146 \cdot (\Theta - 1) + 316 \cdot (\Theta - 1)^2 \,\mathrm{GHz} \qquad (4.3) \\
\gamma_2 &= 39.8 \cdot \gamma_1 \,\mathrm{GHz} \\
\Theta &= 300\,\mathrm{K} \,/\, T
\end{aligned}$$

• Complex permittivity for ice crystals:

$$\begin{aligned}
\epsilon_r' &= 3.15 \\
\epsilon_r'' &= \frac{a}{\nu} + b \cdot \nu \\
a &= (\Theta - 0.1871) \cdot \exp(17.0 - 22.1 \cdot \Theta) \qquad (4.4) \\
b &= \left[\left(\frac{0.233}{1 - 0.993/\Theta}\right)^2 + \frac{6.33}{\Theta} - 1.31\right] \cdot 10^{-5} \qquad (4.5) \\
\Theta &= 300\,\mathrm{K} \,/\, T
\end{aligned}$$

The absorption is directly proportional to the liquid or ice water content LWC/IWC and inversely proportional to the density of a single liquid ice particle $\rho_{\mathrm{l,i}}$. Like the mean particle radius, the liquid and ice water content have a high variability. Table 4.1 reflects this variability by summarizing different literature values for several cloud types. Additional uncertainty of this absorption term comes from two sides: (1) the difference to the Rayleigh approximation of the order of 1-6% as reported in *Li et al.* [1997] and (2) from the fit of the complex permittivity. Since $\epsilon(\nu, T)$ was fitted to measurements which were mostly performed above 0°C, the extrapolated values for T <0°C for super-cooled clouds are not well established. For example in *Liebe et al.* [1991] itself two different parameterizations for the so called primary relaxation frequency (γ_1 in Equation 4.2) are given, one polynomial in Θ as presented in Equation 4.2) and an exponential function derived from theory. Although the polynomial describes the selected data better than the exponential function, this might not be true for temperatures well below 0°C. The difference in γ_1 according to these two approaches can be more than 2 GHz for very low temperatures [*Lipton et al.*, 1999]. The resulting consequences from this discrepancy for the absorption calculation at three microwave frequencies are shown in Figure 4.2. A more detailed discussion about this source of uncertainty is given in Section 4.2.

4.2 Variability and Uncertainty in Cloud Absorption

In the case of clouds three sources of uncertainties can be considered at first sight: (1) validity of the Rayleigh approximation (2) the parameterization of the relative dielectric constants (ϵ_r) of water and ice in the microwave region, and (3) the statistical and climatological variability of the cloud liquid water and ice content.

As it was stated above (Section 4.1) the Rayleigh approximation is valid for particle sizes $< 500\,\mu$m. Figure 4.1 shows a particle size distribution for water clouds and ice

		liquid water content (LWC)	
cloud	class	(g/m^3)	reference
stratus	St	0.15	*Salby* [1996]
		0.09-0.9	*Seinfeld and Pandis* [1998]
		0.28-0.3	*Hess et al.* [1998]
		0.29	*Kneizys et al.* [1996]
nimbostratus	Ns	0.4	*Salby* [1996]
		0.65	*Kneizys et al.* [1996]
		0.05-0.3	*Berton* [2000]
altostratus	As	<0.01-0.2	*Seinfeld and Pandis* [1998]
		0.41	*Kneizys et al.* [1996]
		0.1-1	*Berton* [2000]
stratocumulus	Sc	0.3	*Salby* [1996]
		<0.1-0.7	*Seinfeld and Pandis* [1998]
		0.15	*Kneizys et al.* [1996]
		<0.5	*Pawlowska et al.* [2000]
		0.05-1	*Berton* [2000]
cumulus	Cu	0.5	*Salby* [1996]
		0.26-0.44	*Hess et al.* [1998]
		1.00	*Kneizys et al.* [1996]
cumulonimbus	Cb	2.5	*Salby* [1996]
		0.1-2	*Berton* [2000]
cumulus congestus	Cg	0.1-3.2	*Berton* [2000]
FIRE-ACE	-	<0.7	*Shupe et al.* [2000]

		ice water content (IWC)	
cloud	class	(g/m^3)	reference
cirrus	Ci	0.025	*Salby* [1996]
		0.00193-0.0260	*Hess et al.* [1998]
		$3.128\cdot10^{-4}$-0.06405	*Kneizys et al.* [1996]
		0.15-0.3	*Larsen et al.* [1998]
		<0.1	*Berton* [2000]
cirrostratus	Cs	0.2	*Salby* [1996]
		0.05-2	*Berton* [2000]

Table 4.1: Stated values for the liquid and ice water content of several cloud classes from different sources.

clouds (cirrus) from the OPAC model [*Hess et al.*, 1998]. According to this model only cirrus clouds will have particles of size larger than 500 μm. Nevertheless one has to keep in mind that the variability of the particle size can be very high so that at certain conditions some cloud types (most probable is the cumulonimbus) a non-negligible large particle concentration can occur.

The uncertainty in the relative dielectric constant of water (see e. g. *Lipton et al.* [1999]) is largest below the freezing temperature, since only a few measurements at -4^{o}C contributed to the parameterization of ϵ_r in *Liebe et al.* [1991], which in turn is used in the cloud liquid water absorption model of MPM93. Figure 4.2 shows a comparison of *Liebe et al.* [1991] and *Ray* [1972][3] parameterizations for the temperature dependence of the expression $\Im[(\epsilon_r - 1)/(\epsilon_r + 2)]$, which is in the Rayleigh approximation one of the relevant terms in the absorption calculation (see Equation 4.1). Additionally the same calculations with the alternative expression of the first relaxation frequency, γ_1, as stated in Equation 2b of *Liebe et al.* [1991] is shown. The three versions give comparable results for temperatures warmer than 260 K but show significant differences for temperatures below 240 K. However, an uncertainty estimation of $\Im[(\epsilon_r - 1)/(\epsilon_r + 2)]$ is due to the lack of measurements not easy, but it will certainly increase with decreasing temperature.

The largest variability of the involved quantities of cloud absorption is the liquid and ice water content (LWC and IWC) of the clouds (see Table 4.1). Even within a single cloud the LWC (IWC) changes with altitude and the distance from the cloud center as can be seen for example in Figure 10 of *Ludlam and Mason* [1957] and in the model study of *Costa et al.* [2000].

4.3 Water Vapor Saturation Adjustment in the Cloud

The arts method `WaterVaporSaturationInClouds` assures that the water vapor partial pressure is automatically set to saturation pressure (100 % relative humidity) in the cloud vertical range. This method sets the water vapor partial pressure to the saturation pressure over liquid water in case where liquid clouds are present and to the saturation pressure over ice where ice or water/ice clouds are present. The calculation of the saturation pressure is calculated according to the Goff-Gratch approximation [*Liebe et al.*, 1993]:

$$\theta = (373.16\,\mathrm{K}/T) \tag{4.6}$$

$$x = A \cdot (\theta - 1) + B \cdot \log(\theta) + C \cdot \left(10^{d\cdot(1-\theta^{-1})} - 1\right) + E \cdot \left(10^{g\cdot(\theta-1)} - 1\right) \tag{4.7}$$

$$e_s^w = 101324.6 \cdot 10.0^x\ \mathrm{Pa} \tag{4.8}$$

with

$$\begin{aligned} A &= -7.90298 \\ B &= 5.02808 \\ C &= -1.3816 \cdot 10^{-7} \\ d &= 11.344 \\ E &= 8.1328 \cdot 10^{-3} \\ g &= -3.49149 \end{aligned}$$

[3] The calculations for this parameterizastion are performed with the computer code of W. Wiscombe, NASA, GSFC (ftp://climate.gsfc.nasa.gov/pub/wiscombe/Refrac_Index/WATER/) For the microwave frequency range this program uses the *Ray* [1972] temperature parameterization.

The H_2O saturation pressure over ice the Goff-Gratch approximation [*Liebe et al.*, 1993] is as follows:

$$\theta = (273.16\,\mathrm{K}/T) \tag{4.9}$$

$$x = A \cdot (\theta - 1) + B \cdot \log(\theta) + C \cdot \left(1 - \theta^{-1}\right) \tag{4.10}$$

$$e_s^i = 610.71 \cdot 10.0^x\,\mathrm{Pa} \tag{4.11}$$

with

$$A = -9.09718$$
$$B = -3.56654$$
$$C = 0.876793$$

4.4 ARTS Workspace Variables and Methods

This section explains how the above described cloud absorption models are represented in the structure of the arts source code and how one can invoke them in the arts control file.

The cloud tags needs not necessarily more input information than normal trace gas tags, since both need only a profile. But to have more flexibility one can run these absorption models as black boxes or with some user given input. In connection with this input parameters we distinguish generally two types, the referenced models which are taken from the literature (e. g. *Liebe et al.* [1993]) and the user model, for which the arts user is providing the necessary parameter values.

Formally the cloud tags are like the continuum or full model tags implemented. Therefore after selecting the cloud tag with the `tagDefine` method, the arts user has to setup the arts internal structure (i. e. the workspace variables *cont_description_names, cont_description_models, and cont_description_parameters*) for the selected continuum tags, which can simply be done by putting the following line into the arts control file:

```
cont_descriptionInit{}
```

After this initialization, the cloud tag specific information has to be transfered to arts. This is possible with the arts method *cont_descriptionAppend*, which has itself three input variables: *tagname*, *model*, and *userparameters*. The user has to specify these input variables in the arts control file for each selected cloud tag. Below is a list of all the implemented continuum tags and the associated valid range of the input variables for *cont_descriptionAppend*. For an overview of the possible continuum tags and their referenced models see Table 4.2 and the online documentation can be found under *arts/doc/doxygen/html/continua_cc.html*.

One has to note at this place that the two input variables *model* and *userparameters* are to some extend redundant. Therefore one can also produce an ambiguity by giving contradicting values for these two input variables. To avoid such ambiguities the arts user should keep in mind the general rule that only the user model (*model* =*"user"*) needs input parameters via the input variable *userparameters*. The referenced models need no input via *userparameters*. If you try to run the arts control file with a referenced model and input parameters you will get an error message. Below in the detailed description of *cont_descriptionAppend* you can find correct examples for all the cloud tags.

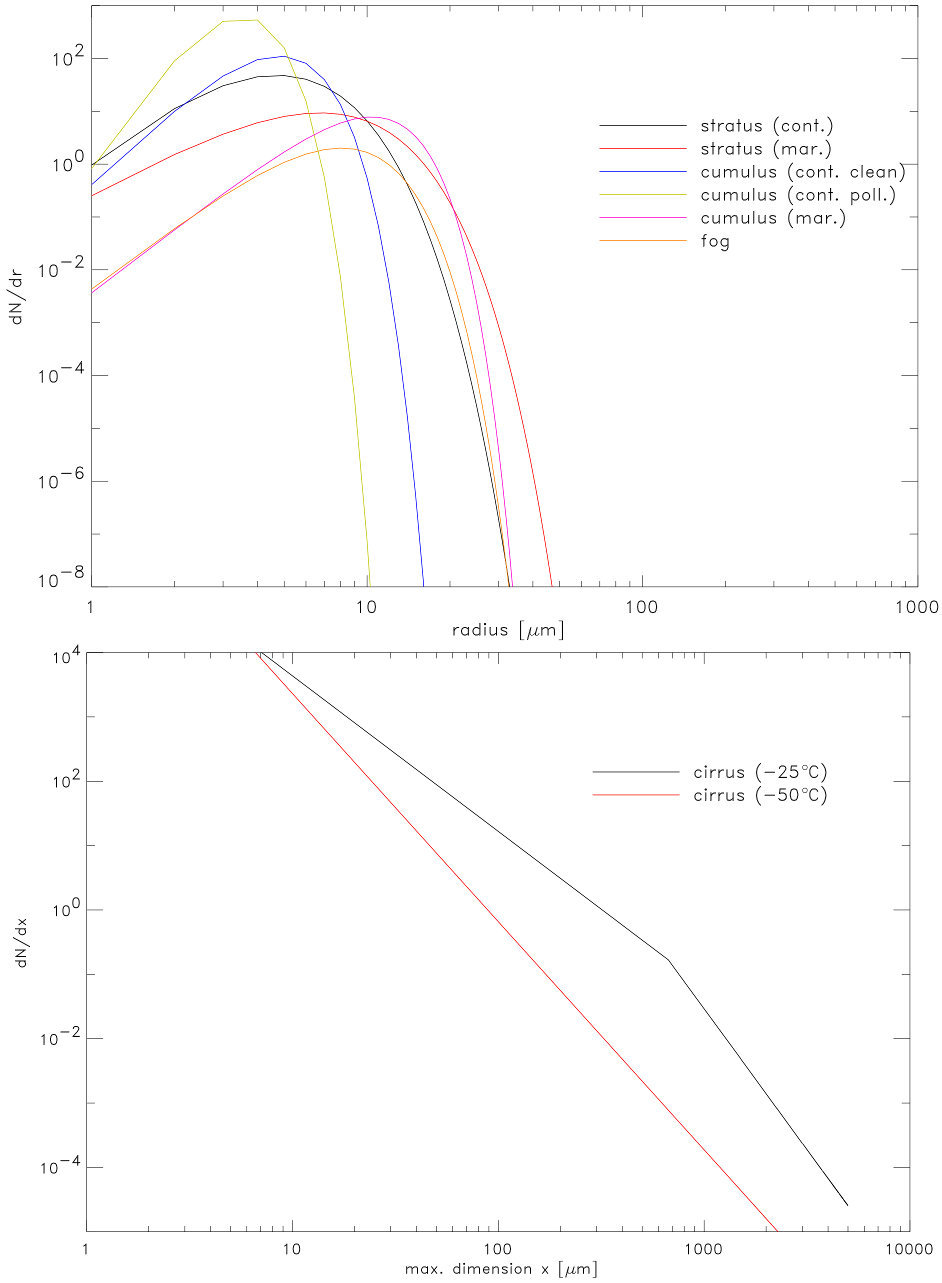

Figure 4.1: Cloud particle size distributions according to Equations 3a and 3c and the microphysical properties are from the Tables 1a and 1b of the OPAC model *Hess et al.* [1998]. For the liquid water clouds (upper plot) a modified gamma distribution is assumed whereas for the ice clouds (lower plot) exponential functions are taken.

- The liquid water cloud absorption model of MPM93 [*Liebe et al.*, 1993] has the arts tag name `"liquidcloud-MPM93"`. The details about this absorption model are described in Section 4. The standard way to use the MPM93 liquid water cloud ab-

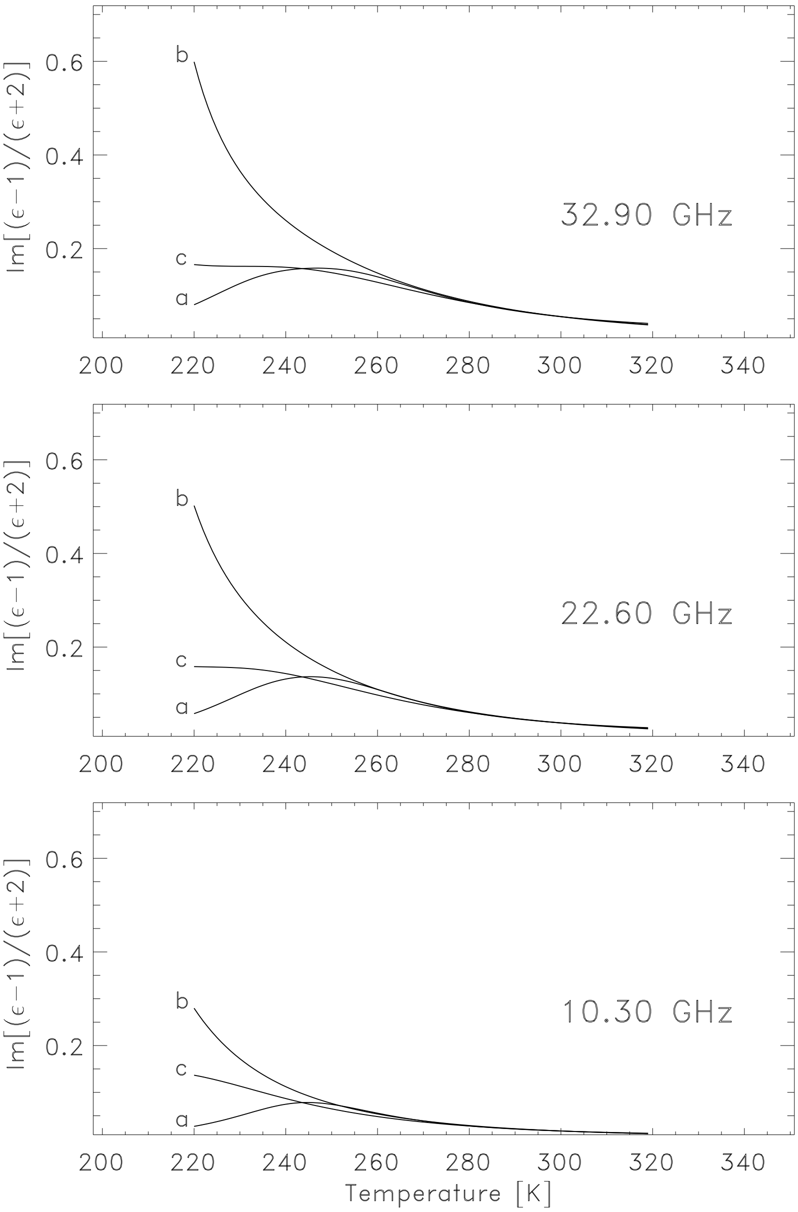

Figure 4.2: Comparison of the imaginary part of the expression $(\epsilon_r - 1)/(\epsilon_r + 2)$ for liquid water at the three frequencies of 32.9, 22.6, and 10,3 GHz. Plotted are the two common models of *Liebe et al.* [1991] (a) and *Ray* [1972] (b). The Ray parameterization is calculated with the F77 program of W. Wiscombe, NASA, GSFC, take from ftp://climate.gsfc.nasa.gov/pub/wiscombe/Refrac_Index/WATER/. Additionally the *Liebe et al.* [1991] parameterization (c) with the alternative expression for the first relaxation frequency, $\gamma_1 = 20.1 \cdot \exp\left[7.88 \cdot (1 - \Theta)\right]$, is plotted.

sorption model is to set the input variable *model* to "MPM93" and leaving the input parameter *userparameters* empty.

To have a minimum possibility of variation one can also run this tag with *model* = "user". In this case one has to provide three input parameters via *userparameters*, e. g. *userparameters* = $[CC, CG, CE]$. The first input parameter (CC) scales the total liquid water cloud absorption (see Eq. 4.1) while CG scales the first relaxation frequency, γ_1, (see Eq. (4.3)) and CE scales ϵ_o (see Eq. 4.2)

```
cont_descriptionAppend{
    tagname        = "liquidcloud-MPM93"
    model          = "MPM93"
    userparameters = [ ]
}
cont_descriptionAppend{
    tagname        = "liquidcloud-MPM93"
    model          = "user"
    userparameters = [ 1.0, 1.0, 1.0 ]
}
```

- The ice water cloud absorption model of MPM93 [*Liebe et al.*, 1993] has the arts tag name `"icecloud-MPM93"`. The details about this absorption model are described in Section 4. The standard way to use the MPM93 ice water cloud absorption model is to set the input variable *model* to "MPM93" and leaving the input parameter *userparameters* empty.
  To have a minimum possibility of variation one can also run this tag with *model* = "user". In this case one has to provide three input parameters via *userparameters*, e. g. *userparameters* = $[CC, CA, CB]$. The first input parameter (CC) scales the total ice water cloud absorption (see Eq. 4.1) while CA scales a, (see Eq. (4.4)) and CB scales b (see Eq. 4.5)

```
# MPM93 model for ice water particle absorption:
cont_descriptionAppend{
    tagname        = "icecloud-MPM93"
    model          = "MPM93"
    userparameters = [ ]
}
# MPM93 model for ice water particle absorption:
cont_descriptionAppend{
    tagname        = "icecloud-MPM93"
    model          = "user"
    userparameters = [ 1.0, 1.0, 1.0 ]
}
```

Another important point is to state in the arts control file which cloud profile should be read in. Becasue for a single climate zone one can have several different cloud types, it is not enough to state e. g. just *midlatitude-summer* or *tropical* as for trace gases. Therefore the method `raw_vmrsReadFromFiles` has to be used to read in the trace gas and clooud profiles. For example the following lines in the control file will read in trace gas profiles from a midlatitude-summer scenario (this information is provided with the input variable *basename* which states the basic scenario) and cumulonimbus and cirrus cloud profiles for liquid water and ice water clouds, respectively:

```
# ATTENTION! THE PATH AND FILE NAMES ARE USER SPECIFIC!
raw_vmrsReadFromFiles
  {seltags   = [ "liquidcloud-MPM93",
                 "icecloud-MPM93" ]
   filenames = [ "cumulonimbus.MPM93droplet.aa",
                 "cirrus.MPM93ice.aa" ]
   basename  =   "midlatitude-summer"
  }
#
```

The method `raw_vmrsReadFromFiles`can be used for any tag and is not specific for cloud tags. In general one has to state in the input variable *seltags* the tags which take their profile information from the files stated in the input variable *filenames*, while all the tags which are not stated in *seltags* take their profile information, i. e. the atmospheric scenario, from the input variable *basename*.

To set the water vapor pressure in the cloud range to the saturation pressure you can use the method `WaterVaporSaturationInClouds` after the call of the method `AtmFromRaw`. The saturation pressure is calculated over liquid water in liquid water clouds and over ice in ice water clouds. If both cloud types are present the saturation over ice is taken.

continuum	*cont_descriptionAppend* input input parameter			reference/ arts uguide	arts source code function
			water vapor (H_2O)		
MPM93	tagname	=	`"liquidcloud-MPM93"`	*Liebe et al.* [1993]	MPM93WaterDropletAbs
	model	=	”MPM93”		
	userparameters	=	[ ]		
MPM93	tagname	=	`"icecloud-MPM93"`	*Liebe et al.* [1993]	MPM93IceCrystalAbs
	model	=	”MPM93”		
	userparameters	=	[ ]		

Table 4.2: This table gives an overview of the implemented referenced cloud absorption models and how they are specified in the arts method *cont_descriptionAppend*. Additionally the reference and the arts source code function names (see file *arts/src/continua.cc* are provided. The detailed online documentation can be found under *arts/doc/doxygen/html/continua_cc.html*).

ARTS Example Control File for the Full Model Tags

Below you will find an example of a control file for all the implemented cloud absorption models. At the moment only the MPM93 model for water and ice clouds is implemented. Please note that to run this example control file you have to specify user specific paths and input file names to run it properly. You can find this example in the arts directory *arts/doc/examples/cloud_example.arts*

```
# ########## EXAMPLE CLOUD TAG CONTROL FILE #########
#--------------------------------------------------
# define the arts cloud and additional tags of arts
tgsDefine{
      [
        "H2O-MPM93",
        "O2-MPM93",
        "N2-SelfContStandardType",
        "liquidcloud-MPM93",
        "icecloud-MPM93"
      ]
}
#
#--------------------------------------------------
# initialize the continua tag structures
cont_descriptionInit{}
#
#--------------------------------------------------
#
#
# ----- H2O full models (line+continuum) ------------
#
# MPM93 H2O absorption model (lines + continuum)
cont_descriptionAppend{
    tagname        = "H2O-MPM93"
    model          = "MPM93"
    userparameters = [ ]
}
#
# ----- N2 continuum -------------------------------
#
cont_descriptionAppend{
    tagname        = "N2-SelfContStandardType"
    model          = "Rosenkranz"
    userparameters = [ ]
}
#
# ----- O2 full models (line+continuum) -------------
#
# MPM93 O2 absorption model (lines + continuum)
cont_descriptionAppend{
    tagname        = "O2-MPM93"
    model         = "MPM93Continuum"
    userparameters = [ ]
}
#
# ----- liquid water particle -----------------------
#
# MPM93 model for liquid water particle absorption:
cont_descriptionAppend{
    tagname        = "liquidcloud-MPM93"
    model          = "MPM93"
    userparameters = [ ]
}
# ----- ice water particle --------------------------
# MPM93 model for ice water particle absorption:
cont_descriptionAppend{
    tagname        = "icecloud-MPM93"
    model          = "MPM93"
    userparameters = [ ]
}
#
#--------------------------------------------------
#
# Read the pressure, temperature, and altitude
# profiles and create the workspace variable 'raw_ptz'.
# ATTENTION! THE PATH AND FILE NAMES ARE USER SPECIFIC!
MatrixReadAscii (raw_ptz)
  {"@ac_arts_data@/atmosphere/fascod/midlatitude-summer.tz.aa"}
#
```

cloud tag selection for calculation.

initialize the cloud tag description structure in arts. This is essential for the later use of the method cont_descriptionAppend.

description of every cloud tag also mentioned in the tagDefine methode above. Each description has three input variables:

* tag name
* model to select a referenced model or the user model
* user given input parameters (only valid for model "user", otherwise leave it blank)

Only in the case where the model "user" is selected, the user given input parameters are considered. All other models neglect these input parameters.

```
# ATTENTION! THE PATH AND FILE NAMES ARE USER SPECIFIC!
raw_vmrsReadFromFiles
  {seltags   = ["liquidcloud-MPM93", "icecloud-MPM93"]
   filenames = ["@ac_arts_data@/atmosphere/particles/midlatitude-summer.cumuloni
mbus.MPM93droplet.aa",
                "@ac_arts_data@/atmosphere/particles/midlatitude-summer.cirrus.M
PM93ice.aa"]
   basename  =  "@ac_arts_data@/atmosphere/fascod/midlatitude-summer"
  }
#
# Create the pressure grid 'p_abs' (just an example)
VectorNLogSpace(p_abs){
        start = 100000.000
        stop  =   1000.000
        n     = 10
}
# Now interpolate all the raw atmospheric input onto the pressure
# grid and create the atmospheric variables 't_abs','z_abs','vmrs'
AtmFromRaw{}
#
# set the H2O VMR in clouds to saturation level
# (must be called after AtmFromRaw)
WaterVaporSaturationInClouds{}
#
#----------------------------------------------------
#
# Set the physical H2O profile from the H2O profile in vmrs:
h2o_absSet{}
#
# Set the physical N2 profile from the N2 profile in vmrs:
n2_absSet{}
#
#----------------------------------------------------
#
# Read spectral line data from HITRAN96 catalogue for
# the frequency range from 1 to 2 GHz.
# This in not essential for the continuum tags but
# bust be given as input for absCalc below.
# ATTENTION! THE PATH AND FILE NAMES ARE USER SPECIFIC!
#
lines_per_tgReadFromCatalogues{
  filenames = [ "@ac_arts_data@/spectroscopy/hitran96/hitran96_lowfreq.par" ]
  formats   = [ "HITRAN96" ]
  fmin      = [ 1.0e9 ]
  fmax      = [ 2.0e9 ]
}
#
# Create an example frequency grid 'f_mono'
VectorNLinSpace(f_mono){
        start =    100.0e9
        stop  =    200.0e9
        n     =    100
}
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#----------------------------------------------------
#
```

Information about the model atmosphere. Also the VMR profiles H2O and N2 have to be given seperately.

Water vapor saturation in the cloud range

Spectral line data is also necessary for the method absCalc.

Input frequency grid on which the calculation is performed.

```
# Set the lineshape function for each continuum tag
lineshape_per_tgDefine{
        shape               = [ "no_shape",
                                "no_shape",
                                "no_shape",
                                "no_shape",
                                "no_shape"]
        normalizationfactor = [ "no_norm",
                                "no_norm",
                                "no_norm",
                                "no_norm",
                                "no_norm"]
        cutoff              = [ -1,
                                -1,
                                -1,
                                -1,
                                -1]
}
#
#----------------------------------------------------
#
# calculate the absorption coefficients, unit=1/meter
absCalc{}
#
#----------------------------------------------------
#
# These we definitely want to write to files:
# 1. absorption coefficient per continuum tag
ArrayOfMatrixWriteAscii (abs_per_tg) {""}
# 2. temperature profile
VectorWriteAscii (t_abs)  {""}
# 3. altitude grid
VectorWriteAscii (z_abs)  {""}
# 4. pressure grid
VectorWriteAscii(p_abs)   {""}
# 5. frequency grid
VectorWriteAscii (f_mono) {""}
# 6. cont_descriptionAppend continuum tagnames
ArrayOfStringWriteAscii (cont_description_names) {""}
# 7. cont_descriptionAppend model selections
ArrayOfStringWriteAscii (cont_description_models) {""}
# 8. cont_descriptionAppend user given input parameters
ArrayOfVectorWriteAscii (cont_description_parameters) {""}
#####################################################
```

The line shape of the cloud tags are all internally set. Therefore The user has not to specify the line shape here.

This is the method which calculates the absorption coefficients in units of 1/meters.

Here the output is written into the output files.

Chapter 5

Basic radiative transfer

This section discusses the solution of the atmospheric radiative transfer equation (RTE). A non-scattering atmosphere in local thermodynamic equilibrium is assumed. The radiative transfer equation gives the monochromatic (infinite frequency resolution) pencil beam (infinite spatial resolution) spectrum. The main problem here is how to practically and accurately estimate the (continuous) integral in the discrete forward model.

The discussion treats mainly measurements of atmospheric emission. The forward model can also handle pure absorption measurements (that is, emission is neglected) and such observations are discussed last in the section.

The equations of this section are valid for monochromatic pencil beam spectra, no effects of the sensor are considered. How to incorporate sensor effects in the spectra is discussed separately (Sec. 7).

5.1 Introduction

Atmospheric radiative transfer can be expressed generally as

$$I = I_1 e^{-\int_{l_1}^{l_2} \kappa(l) dl} + \int_{l_1}^{l_2} \kappa(l)\sigma(l) e^{-\int_{l}^{l_2} \kappa(l') dl'} dl \tag{5.1}$$

where I is the monochromatic pencil beam intensity, l distance along the line of sight (LOS), l_1 the point of the considered part of the LOS furthest away from the sensor, l_2 the closest point of the LOS, I_1 the intensity at l_1, κ the total absorption along the LOS and σ the source function.[1]

Equation 5.1 is of general validity if σ and κ consider the relevant effects, for example, scattering. However, below in this section it is assumed that there is no scattering and the atmosphere is in local thermodynamic equilibrium.

[1] The symbols κ and σ are used here for the absorption and the source function *along* the LOS. The more commonly used symbols, k and S, respectively, are used below to express the variables as functions of altitude.

History	
000307	Started by Patrick Eriksson.
000908	First version finished by Patrick Eriksson.
031205	Cooling rates added by Patrick Eriksson.

Note that Eq. 5.1 is valid both for the case when the LOS is determined by geometrical calculations and when refraction is considered (the refraction changes however the LOS).

With the assumptions of no scattering and local thermodynamic equilibrium, κ is the summed gaseous absorption, and the source function equals the Planck function, B:

$$\sigma = B(\nu, T) = \frac{2h\nu^3}{c^2} \frac{1}{e^{h\nu/k_B T} - 1} \tag{5.2}$$

giving the blackbody radiation for a temperature T and frequency ν.

If σ is constant along the considered part of the LOS, that is, the temperature is constant for the case $\sigma = B$, the RTE can be solved analytically to give

$$I = I_1 e^{-\tau} + \sigma \left(1 - e^{-\tau}\right) \tag{5.3}$$

where τ is the optical thickness

$$\tau = \int_{l_1}^{l_2} \kappa(l) dl \tag{5.4}$$

The transmission corresponding to τ is

$$\zeta = e^{-\tau} \tag{5.5}$$

5.2 Practical considerations

The LOS can be divided into parts in several ways. As absorption and temperature most likely are avaliable at some vertical grid, the most natural choice would be to define the LOS using this vertical grid. This solution is problematic for limb sounding as the ratio between the distance along LOS and the corresponding vertical distance becomes infinite at the tangent point. Another solution would be to base the division on τ, but such a division does not guarantee that T is close to constant inside the slabs as the vertical extension in some cases could be very large, and each combination of frequency and viewing angle should require a specific division.

As a practical compromise, it was here decided to divide LOS into equal long geometrical steps. With this scheme the division is identical for all frequency components, but changes between the viewing angles, and should give relatively fast and straightforward calculations, maintaining a good accuracy. This approach has been applied successfully in the Odin sub-mm forward model [*Eriksson and Merino*, 1997; *Eriksson et al.*, 2000].

The next question is when and how to calculate LOS and the associated variables. As the determination of weighting functions associated with the absorption needs basically the same quantities as RTE, it is most efficient to do this procedure only once and in such way that the values are suitable for both RTE and the weighting functions. Hence, the LOS calculations shall be a separate part, not included in the RTE functions. The standard use of the forward model should then be:

1. Calculation of absorption coefficients.

2. Determination of LOS.

3. Calculation of the source function and transmissions along LOS.

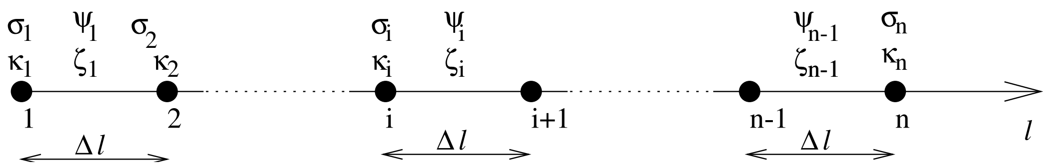

Figure 5.1: Schematic description of the LOS and associated variables. The absorption and the source function at the LOS points are denoted κ_i and σ_i, respectively, while ζ_i is the transmission between the points and Ψ_i is the mean of neighbouring source function values. Only ζ and Ψ are stored for the later calculations. All the points are separated by the distance Δl (along the LOS). The distance between point i and $i+1$ is denoted as step i of the LOS.

4. Iteration to solve RTE.
5. Calculation of weighting functions.
6. Saving etc.

The determination of LOS is described separately in Section 6.

5.3 Practical solution

The LOS is here assumed to be defined with n points where the distance between the points is constant (see Fig 5.1). There are at least two definition points of the LOS ($n \geq 1$). The absorption and the source function are determined at the points of the LOS, and these values are used to calculate the transmission and a mean source function value for the distances between the LOS points. Only the later two quantities are stored.

5.3.1 Absorption and transmission

The absorption is treated to vary linearly between the LOS points. As mentioned above, the transmission values shall be valid between the LOS points. With these definitions, the optical thickness associated with step i is

$$\tau_i = \frac{\Delta l}{2}\left(\kappa_i + \kappa_{i+1}\right), \quad 1 \leq i < n \tag{5.6}$$

The relationship between the optical thicknesses and the transmission is

$$\zeta_i = e^{-\tau_i} \tag{5.7}$$

Note that

$$e^{-(\tau_1+\tau_2\ldots\tau_n)} = \zeta_1\zeta_2\ldots\zeta_n \tag{5.8}$$

The absorption at the LOS points is determined from the absorption matrix provided by the absorption module by linear interpolation, using the logarithm of the pressure as altitude coordinate[2].

[2]The logarithm of the pressure is throughout the basic altitude coordinate in ARTS.

5.3.2 The source function

The source function is also basically assumed to vary linearly between the LOS points, but for simplicity reasons, a single source function value is assigned to the LOS steps:

$$\Psi_i = \frac{\sigma_i + \sigma_{i+1}}{2}, \quad 1 \leq i < n \tag{5.9}$$

The source function at the LOS points (σ) is simply by interpolating linearly the temperature profile, and calculating the Planck function (Eq. 5.2) for the obtained temperatures.

To fully model that the absorption and the source function have a simultanous linear variation between the LOS points would give much more complicated analytical expressions than presented here (if even possible to derive?). However, the simplified approach used here should not influence the accuracy in any important way. This as the source function has, compared to the absorption, a relatively low variation and it can be treated to be piecewise constant when solving the raqdiative transfer.

If long wavelengths are assumed and the source function equals the Planck function (Eq. 5.2), σ should maximally vary with about a factor of 2 as the minimum and the maximum temperature in the atmosphere are about 150 and 300 K, respectively, and the relationship between σ and temperature is close to linear. This should be compared to the absorption that, even for a single frequency, often varies with many orders of magnitude.

5.3.3 Solving the radiative transfer equation

With the definitions given above, the intensity at point n can be expressed as

$$I = I_1 \prod_{j=1}^{n-1} \zeta_j + \sum_{i=1}^{n-1} \left[\Psi_i (1 - \zeta_i) \prod_{j=i+1}^{n-1} \zeta_j \right] \tag{5.10}$$

However, an alternative approach, requiring less computer memory, is to follow the radiation from one slab of the atmosphere to next, and is the method of choice here. Following Equation 5.3, the following iterative expression can be determined [*Eriksson and Merino*, 1997]

$$I_{i+1} = I_i \zeta_i + \Psi_i (1 - \zeta_i) \qquad i = 1, 2, ..., n-1 \tag{5.11}$$

where I_i is the intensity reaching point i. The iteration is started by setting I_1 to the intensity at the atmospheric limit, that is, cosmic background radiation or correspondingly.

5.3.4 Considering ground reflection

The effect of a ground reflection is modeled as

$$I^{after} = I^{before}(1 - e) + eB(\nu, T_{ground}) \tag{5.12}$$

where e is the ground emission factor and I^{before} and I^{after} is the intensity before and after the reflection, respectively. See further Section 6.6.

5.4 Optical thicknesses

The atmospheric emission can be neglected if the observation is performed towards a sufficiently strong source, such as the Sun, and the measurement gives then the total atmospheric transmission, ζ^{tot}. When inverting such observations, the standard approach is to invert the optical thicknesses (τ) to obtain a more linear inversion problem. For this reason, the output from ARTS when neglecting emission was selected to be optical thicknesses instead of transmission values. However, as the transmission for each step along the LOS is stored for the emission calculations, ARTS calculates internally transmission spectra that are converted to optical thicknesses.

This transmission is

$$\zeta^{tot} = e^{-\int_{l_1}^{l_2} \kappa(l)dl} \tag{5.13}$$

The corresponding iterative formula used in the forward model is simply (cf. Eq. 5.11)

$$\zeta^{tot} = \prod_{i=1}^{n-1} \zeta_i \tag{5.14}$$

It is noteworthy that the multiplication order is of no importance, a fact that can be used for 1D limb sounding where the conditions are assumed to be symmetrical around the tangent point and only one half of the line of sight is stored.

If there is a ground reflection, it is considered as

$$\zeta^{tot} = (1-e)\prod_{i=1}^{n-1} \zeta_i \tag{5.15}$$

where e is the ground emission factor.

The optical thicknesses are finally calculated as

$$\tau^{tot} = -\ln(\zeta^{tot}) \tag{5.16}$$

5.5 Cooling rates

Cooling rates is an important concept for climate models and studies. The cooling rate gives the change in temperature due to exchange of radiation, keeping all variables constant. The typical unit is K/day. There are two balancing (more or less) effects. Absorption of shortwave (UV – near IR) results in a heating of the air mass. For thermal IR, emission and absorption are coupled phenomena but exchange of thermal and far IR radiation normally results in a cooling effect. These two effects give together the heating rate, but are normally calculated seperately and are then denoted as the heating and cooling rate. In ARTS only the IR cooling rate is handled, and that only for cloud free conditions. For more details on heating / cooling rates, see any text book on atmospheric physics.

Cooling rates are calculated by the WSM `CoolingRates`. The method returns the spectral cooling rates for the frequencies of `f_mono` and the pressure levels of `p_coolrates`. A positive value here means a cooling effect (that is, the heating rate is not returned). The unit of values in `CoolingRates` is K/day/Hz. Directly below follows a derivation of the expression used in `CoolingRates`.

The heating rate can be expressed as

$$\frac{\mathrm{d}T}{\mathrm{d}t} = \frac{-1}{\rho c_P}\frac{\mathrm{d}F}{\mathrm{d}z}, \tag{5.17}$$

where T is the temperature, t is the time, ρ is the air density, c_P is the heat capacitivity (for pressure work), F is radiative flux and z is vertical altitude. This is the expression normally used to determine cooling rates.

The idea here is to find an expression that gives the heating rate if the spectral radiance as a function of zenith angle is known. As a first step, Equation 5.17 is rewritten to include the spectral flux, F_ν:

$$\frac{\mathrm{d}T}{\mathrm{d}t} = \frac{-1}{\rho c_P}\int_0^\infty \left[\lim_{\Delta z \leftarrow 0} \frac{F_\nu(z+\Delta z) - F_\nu(z)}{\Delta z}\right] \mathrm{d}\nu. \tag{5.18}$$

The relationship between F_ν and spectral radiance, I, is

$$F_\nu(z) = 2\pi \int_0^\pi I(z,\phi)\cos\phi\sin\phi \mathrm{d}\theta. \tag{5.19}$$

The assumption below is that the upwelling part of I is known at z and the downwelling part I is known at $z + \Delta z$.

The upwelling spectral radiance at altitude $z + \Delta z$ can be expressed as

$$I(z+\Delta z,\phi) = I(z,\phi)e^{-\tau(\phi)} + B_\nu(1 - e^{-\tau(\phi)}), \qquad 0 \le \phi \le \pi/2, \tag{5.20}$$

where B_ν is the Planck function for blackbody radiation. The fact that Δz will approach zero has been used, which mean that B_ν and ϕ can be assumed to be constant between z and $z + \Delta z$. The downwelling spectral radiance at z is in similar way

$$I(z,\phi) = I(z+\Delta z,\phi)e^{-\tau(\phi)} + B_\nu(1 - e^{-\tau(\phi)}), \qquad \pi/2 < \phi \le \pi. \tag{5.21}$$

Again using the fact that $\Delta z \approx 0$, the optical depth is

$$\tau(\phi) = \frac{\Delta z}{|\cos\phi|}\alpha \tag{5.22}$$

and the transmission is

$$e^{-\tau(\phi)} = 1 - \frac{\Delta z}{|\cos\phi|}\alpha. \tag{5.23}$$

We have then that

$$\begin{aligned} \tfrac{F_\nu(z+\Delta z)-F_\nu(z)}{\Delta z} &= \tfrac{2\pi}{\Delta z}\Big\{ \textstyle\int_0^{\pi/2} \left[I(z,\phi)(1-\tau(\phi)) + B_\nu\tau(\phi)\right]\cos\phi\sin\phi \mathrm{d}\phi + \\ &+ \textstyle\int_{\pi/2}^{\pi} I(z+\Delta z,\phi)\cos\phi\sin\phi\mathrm{d}\phi - \int_0^{\pi/2} I(z,\phi)\cos\phi\sin\phi\mathrm{d}\phi - \\ &- \textstyle\int_{\pi/2}^{\pi} \left[I(z+\Delta z,\phi)(1-\tau(\phi)) + B_\nu\tau(\phi)\right]\cos\phi\sin\phi\mathrm{d}\phi\Big\}. \end{aligned} \tag{5.24}$$

Many of the terms above cancel out and the expression above can be shortened to

$$\begin{aligned} \tfrac{F_\nu(z+\Delta z)-F_\nu(z)}{\Delta z} &= 2\pi\Big\{ \textstyle\int_0^{\pi/2} \left[B_\nu - I(z,\phi)\right]\alpha\sin\phi\mathrm{d}\phi + \\ &+ \textstyle\int_{\pi/2}^{\pi} \left[B_\nu - I(z+\Delta z,\phi)\right]\alpha\sin\phi\mathrm{d}\phi\Big\}. \end{aligned} \tag{5.25}$$

Putting Equation 5.25 into Equation 5.18, and noting that $I(z,\phi) = I(z+\Delta z,\phi)$ for $\Delta z = 0$, gives finally

$$\frac{\mathrm{d}T}{\mathrm{d}t} = \frac{-2\pi}{\rho c_P} \int_0^\infty \int_0^\pi \left[B_\nu - I(z,\phi)\right] \alpha \sin\phi \mathrm{d}\phi \mathrm{d}\nu. \tag{5.26}$$

We can already here note that the contribution to the heating rate will be zero for frequencies where $\alpha = 0$.

To obtain correct results it is crucial that I converges to B_ν when it is expected that $I = B_\nu$, which is the case when the absorption is very high. Considering that radiative transfer applies a mean value of the Planck function at the end points of the integration step (Eq. 5.9), it is not a good idea to compare I with the B_ν for the position of interest. This would require an extremly short radiative transfer step length (`l_step`), a statement verified by practical calculations. A better solution is to replace $(B_\nu - I(z,\phi))$ in Equation 5.26 by $(\Psi - I(z,\phi))$, where Ψ (defined in Eq. 5.9) is the effective source function for radiative transfer step closest to the point of interest. This modification will balance the radiation budget perfectly for high values of α and has a small impact on the accuracy. However, as for all radiative transfer calculations in ARTS, reducing the step length will improve the calculation accuracy.

5.6 Control file examples

See Section 6.8.

Chapter 6

Line of sight, 1D

This section describes how the line of sight (LOS) is determined for situations where the atmosphere is assumed to be horizontally stratified, a 1D atmosphere. Expressions are given both for pure geometrical calculations and when considering refraction.

6.1 Definitions

Vertical (geometrical) altitudes are denoted as z, pressures as p and distances along the LOS are denoted as l. Vertical distances are measured from the geoid and l is the distance from the lowest point of the LOS.

As a 1D atmosphere is assumed here, the conditions are symmetrical around tangent points and points of ground reflection, and, for such cases, only one half of the LOS is stored for efficiency reasons. The points of the LOS are stored by increasing vertical altitude point. Index 1 corresponds accordingly to either the platform, the tangent point or the ground. The internal description of the LOS is further described in the file `los.h`.

The line of sight is defined by two variables, the platform altitude, z_p, and the zenith angle, ϕ, (see Fig. 6.1):

The platform altitude is the altitude above the geoid of the sensor used to detect the spectrum simulated.

The zenith angle is the angle between the zenith direction and the direction of observation. As an 1D atmosphere is assumed, there is no difference between positive and negative zenith angles.

The lower limit of the atmosphere is given by the ground altitude, z_g. The practical upper limit of the atmosphere is denoted z_{lim} and is in the forward model determined by the highest point of the absorption grid. The absorption grid can extend below z_g. On the other hand, it is not allowed that any part of the LOS is between the lowest absorption altitude and the ground.

History

000307	Started by Patrick Eriksson.
010219	First version finished by Patrick Eriksson.

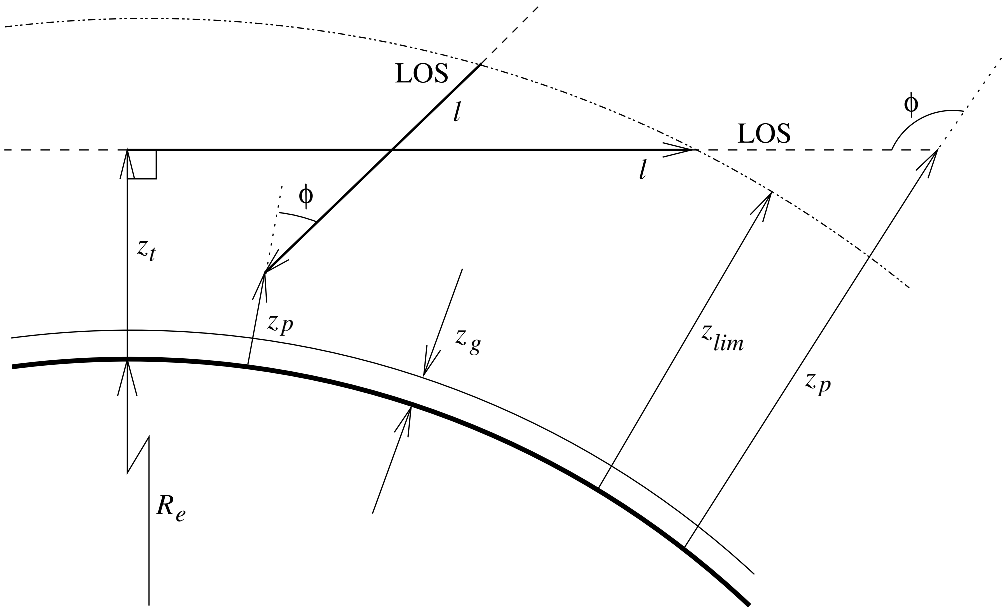

Figure 6.1: Schematic description of the main variables of the observation geometry and the LOS. R_e is the Earth radius. Other variables are defined in the text.

If $\phi > 90°$ the lowest point of the LOS is not the platform altitude, and this point is denoted as the tangent point, z_t. The angle between the LOS and the vector to the Earth center is at the tangent point 90°. If the tangent point is below ground level, z_t is determined by an imaginary geometric prolonging of the LOS inside the Earth.

The forward model uses internally three main observation geometries:

Limb sounding covers here all observations from a point outside the atmosphere ($z_p \geq z_{lim}$). All zenith angles are covered, and, for example, nadir looking observations ($\phi = 180$) are treated as limb sounding in the forward model. If the LOS does not pass the atmosphere ($z_{tan} \geq z_{lim}$), cosmic background radiation, or correspondingly, is returned.

Upward looking signifies observation from within the atmosphere in an upward direction ($z_p < z_{lim}$ and $\phi \leq 90°$).

Downward looking is observation from within the atmosphere in a downward direction ($z_p < z_{lim}$ and $\phi > 90°$).

6.2 Outlook towards 2D

So far ARTS is only capable of calculating spectra for 1D cases. It is planned to also handle satellite measurements with atmospheric horizontal variations, but limited to observations in the orbit plane, here denoted as 2D observations.

For 2D observations there is no symmetry to be used, each point of the LOS is unique. This is also the case for 1D upward looking observations, and it is planned that 2D and 1D upward calculations of radiative transfer and weighting functions shall be performed with the same general functions. The 2D case exhibits however one difference compared to the 1D upward case. For 2D cases there could be a ground reflection along the LOS, which is never the case for 1D upward looking observations by definition. Note that if the 1D upward functions are used for 2D simulations, the point of LOS closest to the sensor will throughout have index 1.

As a first preparation for the 2D calculations, the angular distances between the sensor and the points of the LOS, ψ, are stored beside the pressure and vertical altitudes of the points. The variable ψ is defined to be the angle between the vectors going from the Earth's center to the sensor and the LOS point, respectively. For cases with symmetry, the angles are valid for the part of the LOS furthest away from the sensor.

6.3 The step length

As described in Section 5, the LOS is divided into equal long geometrical steps, Δl. The user gives an upper limit for this step length. A point of the LOS is always placed at the sensor (if inside the atmosphere), tangent points and points of ground reflection, but no adjustment to the upper atmospheric limit is made. This gives a single fixed point for limb sounding and upward looking observation and Δl is set to the value given by the user if the LOS has at least two definition points. If the LOS gets only one point with the user defined value, for example when the tangent point is just below the atmospheric limit, the step length is adjusted to the length from the fixed point of the LOS (the sensor or the tangent point) and the atmospheric limit.

In contrast to upward and limb sounding observations, for downward observations there are two fixed points inside the atmosphere (the platform and the tangent point, or the point of ground reflection) and Δl is here adjusted according to the the distance between these two points. See further Section 6.4.4.

6.4 Geometrical calculations

6.4.1 General expressions

The relationship between vertical altitude (z) and distance along LOS (l) can be found be the law of cosines, giving

$$(R_e + z)^2 = (R_e + z_0)^2 + l^2 + 2l(R_e + z_0)\cos(\phi) \tag{6.1}$$

where z_0 is the lowest point of the LOS (where $l = 0$) and ϕ is the angle between the LOS and zenith at z_0. This equation gives

$$z = \sqrt{(R_e + z_0)^2 + l^2 + 2l(R_e + z_0)\cos(\phi)} - R_e \tag{6.2}$$

The distance between the sensor and the limit of the atmosphere is

$$l_{lim} = \sqrt{(R_e + z_{lim})^2 - (R_e + z_0)^2 \sin^2(\phi)} - (R_e + z_0)\cos(\phi) \tag{6.3}$$

The angle ψ between the point corresponding to z_0 and some altitude z is

$$\psi = \cos^{-1}\left(\frac{(R_e + z_0)^2 + (R_e + z)^2 - l^2}{2(R_e + z_0)(R_e + z)}\right) \tag{6.4}$$

6.4.2 Limb sounding

For limb sounding the lowest point of the LOS is (by definition) the tangent point, and it is given by the expression

$$z_t = (R_e + z_p)\sin(\phi) - R_e \qquad \phi \geq 90^\circ \tag{6.5}$$

This relationship holds even if $z_t < z_g$. Note that $\sin(180^\circ - \phi) = \sin(\phi)$ and it must be checked that $\phi \geq 90^\circ$. Zenith angles $< 90^\circ$ correspond to an imaginary tangent point behind the sensor, and are treated as observations into the space.

The LOS starting at the tangent point is then calculated by Equations 6.2 – 6.4 with $z_0 = z_t$ and $\phi = 90^\circ$. The angle between the vectors going from the Earth's center and the sensor and the tangent point, respectively, is

$$\psi_0 = \phi - 90^\circ \tag{6.6}$$

The value of ψ_0 is added to the angles given by Equation 6.4 as the equation in this case gives the angles from the tangent point instead from the sensor.

If the tangent point is below ground, z_0 is set to z_g and ϕ to ϕ_g where

$$\phi_g = \sin^{-1}\left(\frac{R_e + z_t}{R_e + z_g}\right) \tag{6.7}$$

The correction term for ψ is here

$$\psi_0 = \phi + \phi_g - 180^\circ \tag{6.8}$$

6.4.3 Upward looking

The LOS for upward looking observations is given by Equations 6.2 – 6.4 where z_0 is set to the platform altitude and ϕ to the observation zenith angle.

6.4.4 Downward looking

The altitude of the tangent point is given by Equation 6.5. As both the sensor and the tangent point (or the ground) are treated to be fixed points of the LOS, the step length must be adjusted. The distance between the sensor and a tangent point is

$$l_p = \sqrt{(R_e + z_p)^2 - (R_e + z_t)^2} \qquad z_t \geq z_g \tag{6.9}$$

and the distance between the sensor and a point of ground reflection is

$$l_p = \sqrt{(R_e + z_p)^2 - (R_e + z_t)^2} - \sqrt{(R_e + z_g)^2 - (R_e + z_t)^2} \qquad z_t < z_g \tag{6.10}$$

The part of the LOS between the sensor and the tangent or ground point gets the following number of points:

$$m = 1 + \mathbf{ceil}(l_{lim}/\Delta l_{max}) \tag{6.11}$$

where Δl_{max} is the upper limit for Δl specified by the user, and **ceil** is a function giving the first integer larger than the argument. The step length is accordingly

$$\Delta l = \frac{l_{lim}}{m-1} \tag{6.12}$$

The LOS is determined in the same manner as for limb sounding described above, but with the adjusted value for Δl. The angular distance between the the tangent point, or the ground. and the sensor (ψ_0) is value m of the angle vector given by Equation 6.4.

6.5 With refraction

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

6.5.1 General theory

When determining the LOS through the atmosphere geometrical optics can be applied because the change of the refractive index over a wavelength can be neglected. Applying Snell's law to the geometry shown in Figure 6.2 gives

$$n_i \sin(\theta_i) = n_{i+1} \sin(\theta'_i) \tag{6.13}$$

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

$$\frac{\sin(\theta_{i+1})}{R_e + z_i} = \frac{\sin(180^\circ - \theta'_{i+1})}{R_e + z_{i+1}} = \frac{\sin(\theta'_i)}{R_e + z_{i+1}} \tag{6.14}$$

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

$$c = (R_e + z_i)n_i \sin(\theta_i) = (R_e + z_{i+1})n_{i+1} \sin(\theta_{i+1}) \tag{6.15}$$

where c is a constant. With other words, the Snell's law for spherical atmospheres states that the product of n, $(R_e + z)$ and $\sin(\theta)$ is constant along the LOS.

The radiative transfer is evaluated along the LOS, while Equation 6.15 is expressed for vertical altitudes. The relationship between a change in vertical altitude and the corresponding change along the LOS is here denoted as the geometrical term and it is

$$g(z) = \frac{1}{\cos(\theta)} \tag{6.16}$$

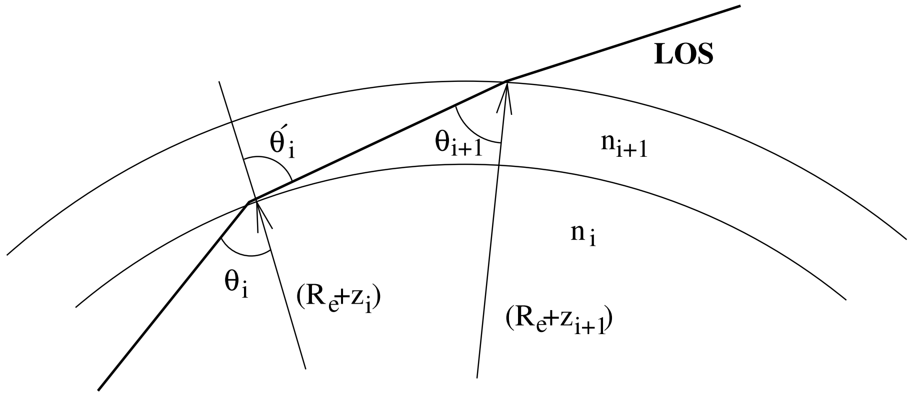

Figure 6.2: Geometry to derive Snell's law for a spherical atmosphere. The Earth radius is R_e, the vertical altitude z, the refractive index n and the angle between the LOS and the vector to the Earth center θ.

which can be rewritten using trigonometric identities and Equation 6.15:

$$g(z) = \frac{(R_e + z)n(z)}{\sqrt{(R_e + z)^2 n^2(z) - c^2}} \tag{6.17}$$

6.5.2 Practical solution

A possible solution for calculating the LOS with refraction would be to integrate numerically the geometrical term [*Eriksson et al.*, 2000] but this approach is problematic for limb sounding as the geometric factor is singular at the tangent point (Figure 6.3). Further, Equation 6.17 cannot be solved analytically for the simple reason that no general analytical expression for n exists. A possible solution would be to assume that n is a piecewise linear function but the solution of Equation 6.17 is then unfortunately a very lengthy expression (at least the one provided by Mathematica!). However, for a piecewise constant n it is very simple to derive a solution of the integral, and thus avoiding the problem with singularities:

$$\Delta l = \sqrt{(R_e + z_2)^2 - \left(\frac{c}{\bar{n}}\right)^2} - \sqrt{(R_e + z_1)^2 - \left(\frac{c}{\bar{n}}\right)^2} \tag{6.18}$$

where z_1 and z_2 are two vertical altitudes, Δl the length along the LOS between these two altitudes and $\bar{n}$ a mean value of the refractive index between z_1 and z_2. The calculations are performed along the LOS and the follwing expression is used in practice

$$z_2 = \sqrt{\left(\Delta l + \sqrt{(R_e + z_1)^2 - \left(\frac{c}{\bar{n}}\right)^2}\right)^2 + \left(\frac{c}{\bar{n}}\right)^2} - R_e \tag{6.19}$$

The angular distance between the points corresponding to z_1 and z_2 is

$$\Delta\psi = \cos^{-1}\left(\frac{(R_e + z_1)^2 + (R_e + z_2)^2 - l^2}{2(R_e + z_1)(R_e + z_2)}\right) \tag{6.20}$$

The practical calculations are performed as follows:

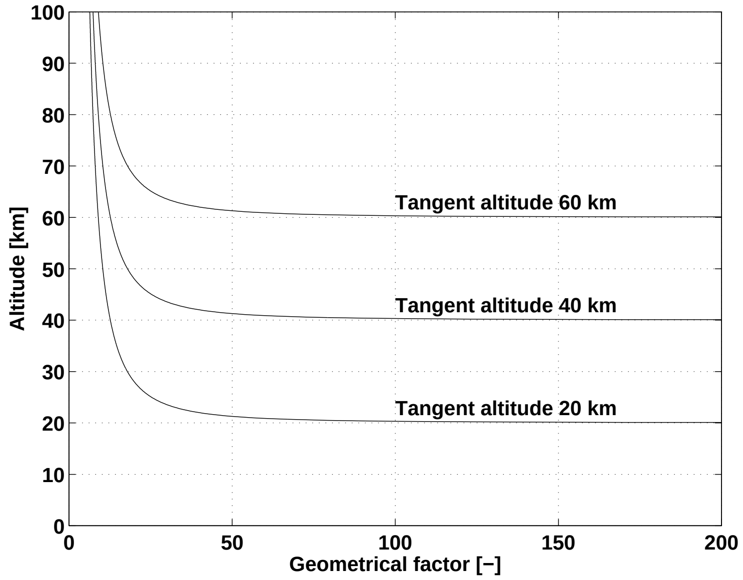

Figure 6.3: The geometrical factor, as a function of altitude, for limb sounding and three tangent altitudes. Taken from *Eriksson and Merino* [1997]

.

1. The lowest point of the LOS is determined and the "zenith angle" at this point.
2. The ray tracing step length is set to the LOS step length divided by the factor given by the user (`refr_lfac`).
3. The ray tracing is performed from the lowest altitude of the LOS until the upper limit of the atmosphere is reached.

Each ray tracing step is performed as

1. The refractive index $(\bar{n})$ is set to the value at z_1.
2. The altitude of the other end of the ray tracing step is calculated by Equation 6.19.
3. The refractive index at z_2 is determined by an interpolation and $(\bar{n})$ is set to the mean value of the refractive index at z_1 and z_2.
4. Step 2 and 3 are repeated two times.
5. The change in the angle ψ is calculated by Equation 6.20.

The number of iterations of step 2 and 3 is hard coded. A practical test showed a clear improvement when going from 1 to 2 iterations, a small improvement when going from 2 to 3 iterations and no practical improvement when going from 3 to 4 iterations. Accordingly, 3 iterations are needed to reach convergence, but as the estimated accuracy for 2 iterations was judged to be sufficient 2 iterations was selected as a compremise between speed and

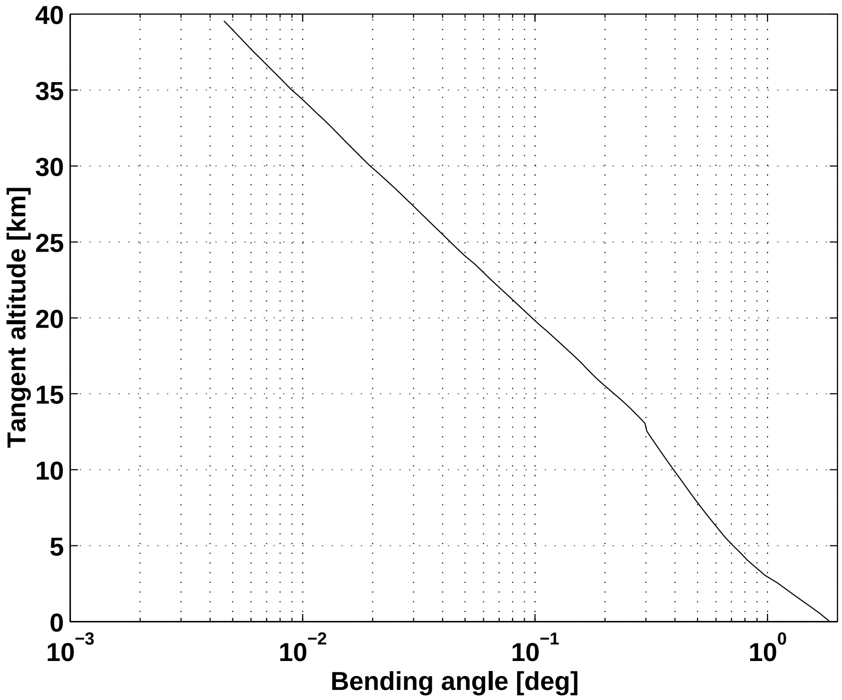

Figure 6.4: Bending angle as a function of tangent altitude. The bending angle is the angle between the line from the tangent point and the sensor and the LOS tangent at the tangent point (see also *Kursinski et al.* [1997, Figure 1]). Calculated for the FASCODE mid-latitude summer atmosphere. The figure can be compared to *Kursinski et al.* [1997, Figure 3] and the agreeemnt is as good as expected.

accuracy. However, if the best accuracy possible is wanted, the number of iterations can easily be changed in the code.

This calculation scheme has the advantage of always starting from the lowest point of the LOS which should be beneficial for the calculation accuracy. How the tangent altitude is determined for limb sounding is described below.

6.5.3 Limb sounding

The most important factor for limb sounding is to get a correct tangent altitude. Fortunately, there is a way to determine the tangent altitude directly for 1D cases, without following the LOS from the top of the atmosphere.

The tangent altitude is given by the relationship

$$(R_e + z_t)n(z_t) = (R_e + z_p)\sin(\phi) = c \tag{6.21}$$

as $\sin(\theta) = 1$ at tangent points, the refractive index in space is 1 and $sin(180° - \phi) = sin(\phi)$. The tangent altitude is practically determined by finding the highest altitude where $(R_e + z)n(z)$ exceeds the value of c, followed by an interpolation of the product $(R_e + z)n(z)$ between the found altitude and the altitude above to find the altitude fulfilling Equation 6.21.

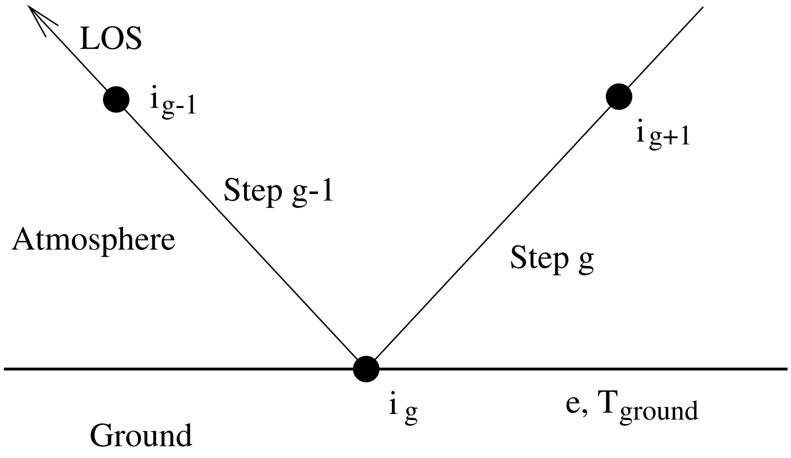

Figure 6.5: Schematic of ground reflections for 2D cases. The index of the point corresponding to the ground is i_g. Point 1 of the LOS is the point closest to the sensor.

For cases with ground reflections, a similar relationship,

$$(R_e + z_g)n(z_g)sin(\theta_g) = (R_e + z_p)\sin(\phi) = c, \tag{6.22}$$

gives the angle between the LOS and the ground normal.

The angular distance between the tangent point and the sensor (ψ_0) is calculated as

$$\psi_0 = \theta_{z_{max}} + \phi - 180^\circ + \psi_{z_{max}} \tag{6.23}$$

where $\theta_{z_{max}}$ and $\psi_{z_{max}}$ are the angles for the highest point of the LOS (θ defined in Figure 6.2).

Figure 6.4 gives a good confirmation of the implemented refraction ray tracing scheme.

6.6 Ground intersections

Ground reflections are indicated by a special flag. This flag is zero when there is no ground intersection or gives the index of the LOS point corresponding to the ground, i_g (for 1-based indexing). For 1D calculations, i_g is either 0 or 1, as index 1 is here defined to always be the lowest altitude of the LOS. However, to pave the way for 2D calculations, cases where the ground is placed at other positions than index 1 are handled.

For 1D cases, where only half of the total LOS is stored and the ground can only have index 1 ($i_g = 1$), the effect of a ground reflection (Eq. 5.12) is put in when reversing the loop order. Accordingly, the calculation order is: ... step2, step 1, ground, step 1, step 2, ... Ground reflections for 1D cases are treated internally in ARTS by the limb sounding functions.

6.7 Inclusion of hydrostatic equilibrium

The pressure in the atmosphere changes as

$$\Delta P = -\rho g \Delta z \tag{6.24}$$

where ΔP is the change in pressure for an altitude change of Δz, ρ is the air density and g the gravitational acceleration. If this expression is combined by the ideal gas law, the hypsometric equation is obtained:

$$z_2 - z_1 = \frac{R_d \bar{T}_v}{g} ln\Big(\frac{P_1}{P_2}\Big) \tag{6.25}$$

where the indeces 1 and 2 indicate two close altitudes, R_d is the gas constant for dry air (287.053 $\mathrm{JK^{-1}kg^{-1}}$) and $\bar{T}_v$ the average virtual temperature between the altitudes z_1 and z_2. The virtual temperature is introduced to include effects of the variable amount of water vapor. If no liquid water is present, the virtual temperature can be calculated as

$$T_v = T\Big(1 + 0.379\frac{x_{H2O}}{1 - x_{H2O}}\Big) \tag{6.26}$$

where x_{H2O} is the volume mixing ratio of water vapor.

The practical calculations take into account that the gravitational acceleration and the average molecular weight changes with altitude. As the exalt altitudes not are known when starting the calculations, and thus there exists some uncertainty for the gravitational acceleration and the average molecular weight, the calculations can be iterated to improve the accuracy. Type `arts -d hseCalc` for some more information.

6.8 Control file examples

The practical calculations are performed by a set of functions `refrCalc`, `losCalc`, `sourceCalc`, `transCalc` and `yCalc`. All these functions have no global input/output or keyword arguments, and the main task is to define the input for the functions. The sequence

```
refrCalc{}
losCalc{}
sourceCalc{}
transCalc{}
yCalc{}
```

must always be used as, for example, the variable `refr_index` must be set when calling `losCalc` and `source` must be set when calculating spectra by `yCalc`. If refraction is not considered, `refrCalc` sets `refr_index` to be empty and `sourceCalc` does the same with `source` for transmission calculations (`emission` = 0).

6.8.1 Ground-based observation

The following control file excerpt shows a typical example for simulating a ground-based observation:

```
# Set the radius of the geoid to a standard value
r_geoidStd{
}

# Set the platform altitude to 50 m
NumericSet( z_plat ) {
   value = 50
```

```
}

# Measurement in the zenith direction
VectorSet( za_pencil ) {
   length = 1
   value = 0
}

# A step length for LOS of 500 m
NumericSet( l_step ) {
   500
}

# Here we don't need to care about the ground and refraction
groundOff{
}
refrOff{
}

# Cosmic radiation
y_spaceStd{ "cbgr" }

# An emission measurement
emissionOn {}

# Do the actual calculations
refrCalc{
}
losCalc{
}
sourceCalc{
}
transCalc{
}
yCalc{
}

# Convert to Rayleigh-Jean temperature
yTRJ{}

# Save the spectra
VectorWriteBinary( y ) {
   ""
}
```

6.8.2 Limb sounding

The following control file excerpt shows a typical example for limb sounding:

```
# Set the geoid radius for observation in the S-N direction
# at latitude 45 degrees
r_geoidWGS84{
   latitude     = 45
   obsdirection = 0
```

```
}

# Set the platform altitude to 620 km
NumericSet( z_plat ) {
   value = 620e3
}

# Five zenith angles between 113.5 and 114.0
VectorNLinSpace (za_pencil) {
   start = 113.5
   stop  = 114.0
   n     = 5
}

# A step length for LOS of 10 km
NumericSet( l_step ) {
   10e3
}

# A blackbody ground at 200 m
groundSet{
   z = 200
   e = 1
}

# Turn on refraction, select parameterization for refractive
# index and set ray tracing step length to 2.5 km
refrSet{
  on    = 1
  model = "Boudouris"
  lfac  = 4
}

# An emission measurement
emissionOn {}

# Cosmic radiation
y_spaceStd{ "cbgr" }

# Do the actual calculations
refrCalc{
}
losCalc{
}
sourceCalc{
}
transCalc{
}
yCalc{
}

# Convert to Rayleigh-Jean temperature
yTRJ{}

# Save the spectra
VectorWriteBinary( y ) {
```

```
   ""
}
```

6.8.3 Limb transmission calculations

Simulation of transmission measurements is performed in the same way as emission observations. Compared to the example above, beside that converion to brightness temperatures shall not be done, the only changes are:

```
...
# Turn off emission
emissionOff {}

# We don't need y_space here, set to be empty
VectorSet( y_space ) {
   length = 0
   value = 0
}
...
```

Chapter 7

Sensor modeling

Modeling of the sensor is not yet part of ARTS. Sensor modeling is so far covered by Qpack but this chapter is included here for completness. On the other hand, conversion of radiances to brightness temperatures is part ARTS and this issue is also discussed here.

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 (that is $\mathbf{y} \neq \mathbf{i}$ always). For a radio (heterodyne) instrument, the most influential sensor parts are the antenna, the mixer, the sideband filter and the spectrometer. Limb sounding observations are also affected by Doppler shifts, but this effect is not considered here, it is assumed to be treated separately. Conversion of radiances to brightness temperatures is also treated here.

7.1 Implementation strategy

7.1.1 The sensor transfer matrix

The modeling of a sensor part is either a summation of different frequency components (mixer), or a weighting of the spectra as a function of frequency (spectrometer) or viewing direction (antenna) with the instrument response of concern. In all cases it is possible to describe the sensor influence by an analytical expression. See for example *Eriksson and Merino* [1997] for more details. These analytical expressions can be implemented and solved for each run of the sensor model, but this would be relatively computationally demanding for cases when the settings are kept constant, as the calculations are duplicated in an unnecessary manner, and we want to find a better implementation strategy.

Summation and weighting of the spectral components are both linear operations, and thus it is possible to model the effect of the different sensor parts as subsequent matrix multiplications of the monochromatic pencil beam spectrum, as suggested in *Eriksson et al.* [2000]:

$$\mathbf{y} = \mathbf{H}_n \ldots \mathbf{H}_2\mathbf{H}_1\mathbf{i} + \varepsilon \tag{7.1}$$

History

000321	Started by Patrick Eriksson.
000826	First version finished by Patrick Eriksson.

where n is the number of sensor parts to consider, and this results in that the sensor model can be expressed as a single matrix multiplication (Eq. 2.9)

$$\mathbf{y} = \mathbf{Hi} + \varepsilon$$

Applying Equation 2.9 for the sensor model will clearly give very rapid calculations, and we must find ways to calculate $\mathbf{H}$.

7.1.2 Normalization of H

It is important that the transfer matrix for each sensor part is normalized in such way that a unit response is obtained. A unit response signifies here that a constant intensity (as a function of frequency or zenith angle) is preserved, that is

$$\mathbf{u}_2 = \mathbf{H}\mathbf{u}_1 \tag{7.2}$$

where $\mathbf{u}_1$ and $\mathbf{u}_2$ are vectors of appropriate length where each element is 1. This criterion equals that the sum of the elements of each row of $\mathbf{H}$ is 1.

7.2 Integration as vector multiplication

The effect of both the antenna and the spectrometer can be expressed as an integral [e.g. *Eriksson and Merino*, 1997, Eq. 86 and 94], and the question is how to transform these integrals into matrix operations.

The problem at hand is that the antenna and spectrometer responses and the zenith angle and frequency grids are known, while the spectral values are unknown. This problem corresponds to determine a (row) vector $\mathbf{h}$ that multiplied with an unknown (column) vector, $\mathbf{g}$, approximates the integral of the product between the functions g and f:

$$\mathbf{hg} = \int f(x)g(x)\mathrm{d}x \tag{7.3}$$

where $\mathbf{g}$ contains values of g at some discrete points. The functions f is here the response for some sensor part, and g holds the spectral values. The shape of f and g between the grid points must be known to solve this problem.

7.2.1 Piecewise linear functions

In this section the problem of Equation 7.3 is solved analytically when both functions are piecewise linear. The practical solution used Qpack is discussed in next section.

Following Figure 7.1, the function g can between the points x_1 and x_4 be expressed as a sum of the two unknown values g_1 and g_2:

$$g(x) = g_1 + (g_2 - g_1)\frac{x - x_1}{x_4 - x_1} = g_1\frac{x_4 - x}{x_4 - x_1} + g_2\frac{x - x_1}{x_4 - x_1} \tag{7.4}$$

which can be rewritten as

$$g(x) = g_1(a + bx) + g_2(c - bx), \qquad x_1 \leq x \leq x_4 \tag{7.5}$$

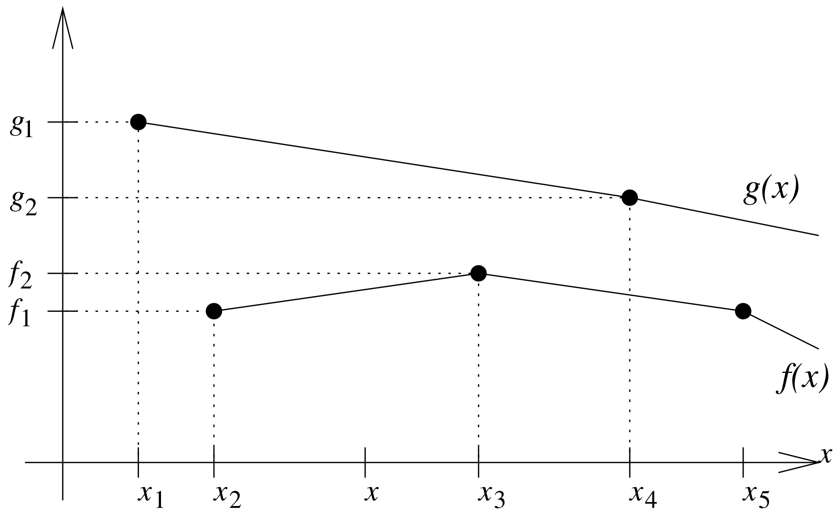

Figure 7.1: The quantities used in Section 7.2.

where

$$a = \frac{x_4}{x_4 - x_1}, \qquad b = \frac{-1}{x_4 - x_1}, \qquad c = \frac{-x_1}{x_4 - x_1}$$

A shorter expression can be obtained for the function f as the values f_1 and f_2 are known:

$$f(x) = (d + ex), \qquad x_2 \leq x \leq x_3 \tag{7.6}$$

where

$$d = f_1 - x_2 \frac{f_2 - f1}{x_3 - x_2} \qquad e = \frac{f_2 - f_1}{x_3 - x_2}$$

The integral in Equation 7.3 can now for ranges between x_2 and x_3 be calculated analytically in a straightforward manner:

$$\int_{x_a}^{x_b} f(x)g(x)\mathrm{d}x = \int_{x_a}^{x_b} (d + ex)(g_1(a + bx) + g_2(c - bx))\mathrm{d}x = \ldots =$$
$$\left[g_1 x\Big(ad + \frac{x}{2}(bd + ae) + \frac{x^2}{3}be\Big) + g_2 x\Big(cd + \frac{x}{2}(ce - bd) - \frac{x^2}{3}be\Big)\right]_{x_a}^{x_b} \tag{7.7}$$

For the practical calculations, the integral is solved from one grid point to next, of either $\mathbf{f}$ or $\mathbf{g}$. The functions are assumed to be zero outside their defined ranges (for example, $f = 0$ for $x < x_2$). For the case shown in Figure 7.1, the integration order would be $(x_a, x_b) = (x_2, x_3)$, $(x_a, x_b) = (x_3, x_4)$, $(x_a, x_b) = (x_4, x_5) \ldots$

Using Equation 7.7, we can now determine how to calculate $\mathbf{h}$. For each integration step, $\mathbf{h}_i$ and $\mathbf{h_{i+1}}$ are increased as

$$\begin{aligned}
\mathbf{h}_i &= \mathbf{h}_i + x_b\Big(ad + \frac{x_b}{2}(bd + ae) + \frac{x_b^2}{3}be\Big) - x_a\Big(ad + \frac{x_a}{2}(bd + ae) + \frac{x_a^2}{3}be\Big) \\
\mathbf{h}_{i+1} &= \mathbf{h}_{i+1} + x_b\Big(cd + \frac{x_b}{2}(ce - bd) - \frac{x_b^2}{3}be\Big) - x_a\Big(cd + \frac{x_a}{2}(ce - bd) - \frac{x_a^2}{3}be\Big)
\end{aligned}$$

where i is the index for which $\mathbf{x}^i \leq x_a$ and $x_b \leq \mathbf{x}^{i+1}$. The vector $\mathbf{h}$ is initialized with zeros before the calculation starts.

7.2.2 Practical solution

The functions f and g can in Qpack be treated to be piecewice linear or cubic functions. The polynomial order of the two functions is set individually. When a function is assumed to be piecewise cubic, two points on each side of the range of interest (that is, in total 4 points) are used to determine the polynomial. For the end ranges, a quadratic polynomial is used as there exists only a single point on one of the sides.

Accordingly, Equation 7.3 must be handled in Qpack for combinations of piecewise linear, quadratic and cubic functions. Instead of repeating the calculations in Section 7.2.1 for all possible polynomial combinations, a more general solution was implemented. The polynomial coefficents for f are simply obtained by doing a polynomial fit to the considered points (by the Matlab function `polyfit`). The polynomial basis for g (a, b and c in Equation 7.5) is obtained by Lagrange's formula (Equation 9.54), which expresses the polynomial that passes a fixed set of points. The Lagrange's formula can be written as:

$$
\begin{aligned}
g(x) \quad = \quad & (a_{11} + a_{12}x + \ldots + a_{1N}x^N) * g_1 + \\
& (a_{21} + a_{22}x + \ldots + a_{2N}x^N) * g_2 + \\
& \ldots \\
& (a_{N1} + a_{N2}x + \ldots + a_{NN}x^N) * g_N
\end{aligned}
\tag{7.8}
$$

With the obtained coefficients for f and g, Equation 7.7 can easily be solved analytically in a general manner. The polynomial pasis is determined by the AMI function `pbasis`, the both set of coefficients are multiplicated in the function `pbasis_x_pol` and the integral is solved by the function `pbasis_integrate`.

7.3 Summation as vector multiplication

The influence of the mixer and sideband filter of the sensor correspond to a summation of pairs of frequency components. The two frequencies of the pair are related as

$$\nu' = 2\nu_{LO} - \nu \tag{7.9}$$

where ν_{LO} is the frequence of the local oscillator signal, and ν' is denoted as the image frequency.

The intensity correspondence after the mixer and the sideband filter can be written as

$$I_{IF}(\nu) = \frac{f_s(\nu)I(\nu) + f_s(\nu')I(\nu')}{f_s(\nu) + f_s(\nu')} \tag{7.10}$$

where $I(\nu)$ is the intensity for frequency ν and f_s the response of the sideband filter as a function of frequency.

The frequency grid after the mixer consists of the frequencies inside the primary band of the grid before the mixer. To include frequencies from the image band (mirrored to the primary band) would need an interpolation in the primary band that could cause unexpected effects.

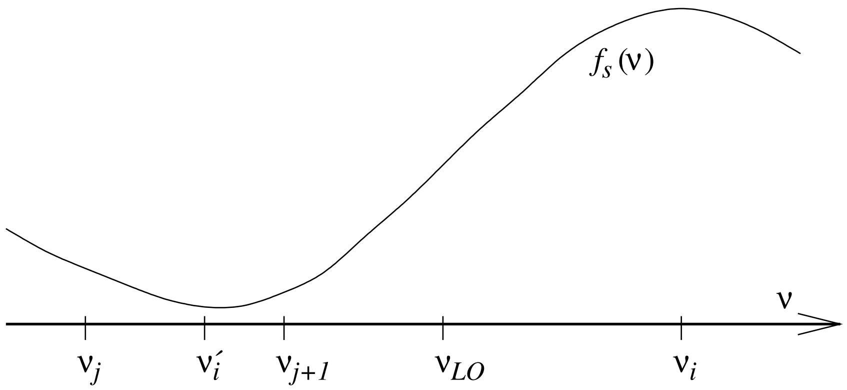

Figure 7.2: Schematic description of image frequency and sideband filtering.

7.3.1 Piecewise linear functions

If the intensity is assumed to vary linearly between the points of the frequency grid, Equation 7.10 can be written as

$$\begin{aligned} I_{IF}(\nu^i) &= \frac{1}{f_s(\nu_i)+f_s(\nu_i')}\Big[f_s(\nu_i)I(\nu_i) + \\ &\quad + \frac{f_s(\nu_i')}{\nu_{j+1}-\nu_j}\Big(I(\nu_j)(\nu_{j+1}-\nu_i') + I(\nu_{j+1})(\nu_i'-\nu_j)\Big)\Big] \end{aligned} \tag{7.11}$$

where f_s for the different frequencies is obtained by linear interpolation, and ν_j and ν_{j+1} are the two points of the frequency grid surrounding the image frequency, ν_i'. The row of the $\mathbf{H}$ matrix corresponding to ν^i is then

$$\begin{aligned} \mathbf{h}^i &= \frac{f_s(\nu_i)}{f_s(\nu_i)+f_s(\nu_i')} \\ \mathbf{h}^j &= \frac{f_s(\nu_i')}{f_s(\nu_i)+f_s(\nu_i')}\frac{\nu_{j+1}-\nu_i'}{\nu_{j+1}-\nu_j} \\ \mathbf{h}^{j+1} &= \frac{f_s(\nu_i')}{f_s(\nu_i)+f_s(\nu_i')}\frac{\nu_i'-\nu_j}{\nu_{j+1}-\nu_j} \end{aligned} \tag{7.12}$$

where $\mathbf{h}^i$ is the value of $\mathbf{h}$ for frequency ν_i etc. Remaining values of $\mathbf{H}$ are zero.

For the special case when the image frequency matches perfectly a frequency grid point, the equations above can be simplified to give

$$\begin{aligned} \mathbf{h}^i &= \frac{f_s(\nu_i)}{f_s(\nu_i)+f_s(\nu_i')} \\ \mathbf{h}^j &= \frac{f_s(\nu_i')}{f_s(\nu_i)+f_s(\nu_i')} \end{aligned}$$

7.3.2 Practical solution

The responses of the sideband filter is determined by linear or cubic interpolation, dependent on the selected order. As the frequency in the primary band always equals one of the points of the monochromatic frequency grid, Equation 7.12 can be used throughout. The weights for the image band are found by evaluating the polynomial basis from Equation 7.8 at ν'_i and multiplicate with $f_s(\nu'_i)/(f_s(\nu_i) + f_s(\nu'_i))$. These calculations are performed in the AMI function `h_matrix`.

7.4 Brightness temperature

Some kind of calibration process, either in absolute or relative units, is always needed. For mm and sub-mm receivers, the calibration normally presents the measured intensity in some temperature scale, and conversion to brightness and Rayleigh-Jeans temperatures is also treated in this section.

7.4.1 Conversion to Planck brightness temperature

The brightness temperature is defined as the temperature a blackbody shall have to give the same intensity magnitude as observed. The brightness temperature is thus calculated as

$$T_b = \frac{h\nu}{k_B} \frac{1}{\ln\left(\frac{2h\nu^3}{c^2 I} + 1\right)} \tag{7.13}$$

where I is the radiative intensity.

It should be noted that the conversion from intensity to brightness temperature is non-linear. This non-linearity has (at least) two important consequences:

- The conversion from intensity to brightness temperature cannot be included in $\mathbf{H}$.
- **Brightness temperature cannot be used for retrievals.**

Accordingly, the main reason to convert a spectrum to brightness temperatures is to display the spectrum in an unit that gives a more intuitive understanding of the emission magnitude.

7.4.2 Conversion to Rayleigh-Jean temperature

For lower frequencies where $h\nu \ll k_B T$ the Planck function can be approximated by the Rayleigh-Jean (RJ) formula:

$$B \approx \frac{2\nu^2 k_B T}{c^2} \tag{7.14}$$

This relationship holds rather well in the microwave region. For example, for $T = 50$ K, $h\nu = k_B T$ at 1.04 THz. The RJ approximation of the Planck function gives a natural definition on a "brightness temperature" with that has a linear relationship to the intensity:

$$T_{rj} = \frac{c^2}{2\nu^2 k_B} I \tag{7.15}$$

This intensity unit is often referred to as the brightness temperature but to avoid confusion it is here denoted as the RJ temperature.

As the intensity from intensity to RJ temperature is linear, this conversion can be included in $\mathbf{H}$ and weighting functions can be converted using 7.15, that is, retrievals are possible using RJ temperatures. On the other hand, the RJ temperature shall not be mistaken for the "physical" brightness temperature (T_b) as the deviation between T_b and T_{rj} is not negligible [*Eriksson and Merino*, 1997].

7.5 Control file examples

The following sequence of ARTS functions can be used to store the spectra in both brightness temperature units:

```
VectorCopy( y0, y ) {
}
yTRJ{
}
VectorWriteAscii( y ) {
   "ytb_rj.aa"
}
VectorCopy( y, y0 ) {
}
yTB{
}
VectorWriteAscii( y ) {
   "ytb_planck.aa"
}
```

A weighting function matrix is converted to Rayleigh-Jean temperature as:

```
MatrixTRJ( kx, kx ) {
}
```

Chapter 8

Data reduction

Many observation scenarios give rise to very large measurement vectors, larger than can be handled practically during the inversions, and some kind of reduction of the data size is needed. This data reduction can be made part of the sensor transfer matrix. In fact, the data reduction can be viewed upon as an imaginary second spectrometer. The transfer matrix to use is then (Eq. 2.10)

$$\mathbf{H} = \mathbf{H}_d \mathbf{H}_s$$

where $\mathbf{H}_d$ is the data reduction matrix and $\mathbf{H}_s$ the sensor matrix. *Data reduction can so far only be performed in Qpack.*

8.1 Averaging of viewing angles

In some cases the spectra from different viewing angles are combined, either as a pure data reduction or internally in the spectrometer. The rows of $\mathbf{H}_d$ for this case have the structure

$$\mathbf{h} = [0, \ldots, 0, \frac{1}{n_v}, 0, \ldots, 0, \frac{1}{n_v}, 0, \ldots, 0, \frac{1}{n_v}, 0, \ldots, 0] \tag{8.1}$$

where n_v is the number of viewing angles to combine.

8.2 Data binning

Data binning means that neighboring channels are combined by weighted averaging. If channels i_1 to i_2 of $\mathbf{y}'$ are combined to give element j of $\mathbf{y}$, the binning can be expressed as

$$\mathbf{y}^j = \frac{1}{\sum_{i=i_1}^{i_2} \Delta\nu^i} \sum_{i=i_1}^{i_2} \Delta\nu^i (\mathbf{y}')^i \tag{8.2}$$

History

000321 Created and written by Patrick Eriksson.

Row j of $\mathbf{H}_d$ is accordingly

$$\mathbf{h}^i = \frac{\Delta\nu^i}{\sum_{i=i_1}^{i_2} \Delta\nu^i}, \qquad i_1 \leq i \leq i_2 \tag{8.3}$$

Other values of $\mathbf{h}$ are zeros. The matrix $\mathbf{H}_d$ is for data binning highly sparse.

8.3 Reduction by eigenvectors

A commonly used approach for reducing data sizes is to base the reduction of the eigenvectors of the covariance matrix expressing the variability of the measurements. These empirical eigenvectors fulfills the relationships

$$\mathbf{S_y} = \mathbf{E}\Lambda\mathbf{E}^T \tag{8.4}$$

where Λ is a diagonal matrix holding the eigenvalues corresponding to the eigenvectors, the columns of $\mathbf{E}$. The eigenvectors form an orthogonal basis:

$$\mathbf{I} = \mathbf{E}_j^T\mathbf{E}_j \tag{8.5}$$

where $\mathbf{E}_j$ signifies the j first columns of the matrix.

The data reduction for this case is performed as

$$\mathbf{y} = \mathbf{E}_j^T\mathbf{y}' \tag{8.6}$$

that is

$$\mathbf{H}_d = \mathbf{E}_j^T \tag{8.7}$$

By basing the data reduction on the covariance matrix eigenvectors, the reduction maintaining the maximum possible fraction of the variability of the spectra, for a given j, is achieved.

Different versions of this scheme are described in *Eriksson et al.* [2001a]. The existing options in Qpack are described in the file `README`.

Chapter 9

Atmospheric weighting functions

This section describes how the calculation of the atmospheric weighting functions (WFs) matrices is performed in the forward model. For several types of variables (such as species profiles and fit of absorption continuum) WFs are obtained by semi-analytical expressions, while for other quantities the WFs are obtained by straightforward perturbation calculations.

9.1 Calculation approaches

9.1.1 Pure numerical calculation

The most straightforward method to determine WFs is by perturbing one parameter at a time. For example, the WF for the state variable p can always be calculated as

$$\mathbf{K}_{\mathbf{x}}^{p} = \frac{\mathcal{F}(\mathbf{x} + \Delta\mathbf{x}^{p}\mathbf{e}^{p}, \mathbf{b}) - \mathcal{F}(\mathbf{x}, \mathbf{b})}{\Delta\mathbf{x}^{p}} \tag{9.1}$$

where $\mathbf{K}_{\mathbf{x}}^{p}$ is column p of $\mathbf{K}_{\mathbf{x}}$, $(\mathbf{x}, \mathbf{b})$ is the linearization state, $\mathbf{e}^{p}$ is a vector of zeros except for the component p that is unity, and $\Delta\mathbf{x}^{p}$ 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 in general either part of the atmospheric or the sensor state, but not both. If $\mathbf{x}^{p}$ is an atmospheric variable, the calculation can be performed as (Eq. 2.14)

$$\mathbf{K}_{\mathbf{x}}^{p} = \mathbf{H}\left[\frac{\mathcal{F}_r(\mathbf{x}_r + \Delta\mathbf{x}^{p}\mathbf{e}^{p}, \mathbf{b}_r) - \mathcal{F}_r(\mathbf{x}_r, \mathbf{b}_r)}{\Delta\mathbf{x}^{p}}\right] \tag{9.2}$$

where $\mathbf{x}_r$ is the atmospheric part of the state vector etc (see further Sec. 2).

9.1.2 Analytical expressions

For some atmospheric variables, such as species abundance, it is possible to derive a semi-analytical expression for the WFs. This is advantageous because it results in faster and more

---

History

000310 Started by Patrick Eriksson.
000911 First version finished by Patrick Eriksson.

accurate calculations. By Equation 2.14,

$$\mathbf{K_x} = \mathbf{H}\frac{\partial \mathbf{i}}{\partial \mathbf{x}},$$

it can be seen that the core problem of finding these analytical expressions is to determine $\partial\mathbf{i}/\partial\mathbf{x}$. If $\mathbf{x}^p$ influences only the conditions at one altitude, the problem can be simplified as [*Eriksson et al.*, 2000, Eq. 43]

$$\mathbf{K}^p_\mathbf{x} = \mathbf{H}\frac{\partial \mathbf{i}}{\partial \mathbf{x}^p} = \mathbf{H}\left[\frac{\partial \mathbf{i}}{\partial \mathbf{S}^p}\frac{\partial \mathbf{S}^p}{\partial \mathbf{x}^p} + \frac{\partial \mathbf{i}}{\partial \mathbf{k}^p}\frac{\partial \mathbf{k}^p}{\partial \mathbf{x}^p}\right] \tag{9.3}$$

where $\mathbf{S}^p$ and $\mathbf{k}^p$ are the source function and the absorption at the (vertical) altitude p, respectively.

It is very important to note that the analytical expressions are derived with the assumption that $\mathbf{x}^p$ influences only the local conditions. For species it is further assumed that the absorption can be expressed as (see Section 9.6 for definitions and details)

$$\mathbf{k}^p = \bar{\mathbf{k}}^p_s\mathbf{x}^p + \sum_{i\neq s}\mathbf{k}^p_i \tag{9.4}$$

These assumption should be of general validity for species above the tropopause. Two examples on when the analytical expressions will be approximative are

- The variable of interest can change the line-of-sight (by the refractive index). This is an example of a non-local effect. This is always valid for temperature.
- The amount of different species must be considered when calculating the pressure broadening, and not only the total absorption.

If the analytical expressions can be used for such cases must be tested numerically. When it is found that the analytical approach cannot be used, the WFs must be calculated by perturbations to include the neglected effects (such a function for species is not yet implemented in ARTS). An important example when these questions must be considered is limb sounding of water vapor in the troposphere where both points above are true. The abundance of water in the troposphere is sufficient high to have a significant influence on both the refractive index and the pressure broadening. These questions are discussed somewhat further in *Eriksson et al.* [2001b].

The absorption and source function in Equation 9.3 are defined in vertical coordinates (as we retrieve atmospheric variables as functions of altitude). For different reasons it is more practical to work with these quantities defined along the LOS. For example, the source function and transmission along the LOS are already determined when calculating the spectra. To solve this problem, Equation 9.3 is expanded one step further

$$\mathbf{K}^p_\mathbf{x} = \mathbf{H}\left[\frac{\partial \mathbf{i}}{\partial \sigma}\frac{\partial \sigma}{\partial \mathbf{S}^p}\frac{\partial \mathbf{S}^p}{\partial \mathbf{x}^p} + \frac{\partial \mathbf{i}}{\partial \kappa}\frac{\partial \kappa}{\partial \mathbf{k}^p}\frac{\partial \mathbf{k}^p}{\partial \mathbf{x}^p}\right] \tag{9.5}$$

where σ and κ are the source function and the absorption along the LOS, respectively.

The term $\partial\mathbf{i}/\partial\sigma$ is here denoted as source line of sight weighting functions (source LOS WFs) and is discussed in Section 9.4. The term $\partial\mathbf{i}/\partial\kappa$ is denoted as absorption LOS WFs

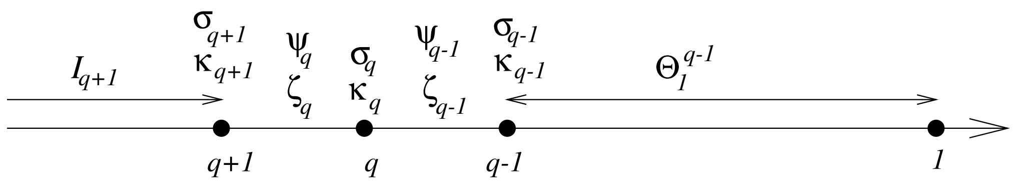

Figure 9.1: The terms used for the derivation of line of sight weighting functions when the individual atmospheric parts are passed a single time. The variables are defined in Figure 5.1.

and is discussed in Sections 9.2 and 9.3. These terms are treated separately as they are common for all variables influencing the source function or the absorption.

The term $\partial \mathbf{S}^p/\partial \mathbf{x}^p$ can often be neglected. When scattering is neglected and local thermodynamic equilibrium is assumed, the only variable of interest affecting the source function is the temperature. See further Section 9.8. For other variables, such as species abundance, $\partial \mathbf{S}^p/\partial \mathbf{x}^p = 0$.

It was decided to allow that the retrieval grids differ between species, temperature etc. This results in that the terms $\partial\sigma/\partial \mathbf{S}^p$ and $\partial\kappa/\partial \mathbf{k}^p$ are not constant, they change according to the selected retrieval grid. Accordingly, it is not suitable to include these terms in the corresponding LOS WFs, they must be treated separately.

9.2 Absorption LOS WFs with emission

The absorption line of sight weighting functions are defined as

$$\mathbf{K}_\kappa^q = \frac{\partial \mathbf{i}}{\partial \kappa^q} \tag{9.6}$$

These weighting functions express how the intensity is affected by changes of the absorption at the points of the line of sight. Note that κ is the total absorption, not the absorption of a single species.

For simplicity, the absorption LOS WFs are below derived without using vector notation. The notation used here is identical to the one used in Section 5. The calculation approach used for the LOS WFs is "inspired" by the corresponding work in *Reburn et al.* [2000].

9.2.1 Single pass

This section derives the absorption LOS WFs for cases when each individual part of the atmosphere is passed only once, as for upward looking measurements, or when each point in the atmosphere is treated separately (2D simulations). With other words, the conditions are not assumed to be symmetrical around some point. Accordingly, 1D limb sounding and 1D downward observations are not treated here, and are instead discussed in Section 9.2.2 and 9.2.3, respectively.

By rewriting Equation 5.10, the monochromatic pencil beam intensity can be expressed in the following ways (see Fig. 9.1)[1]

$$\begin{aligned} I &= I_2\zeta_1 + \psi_1(1-\zeta_1) \quad (q=1) \\ I &= \Big[I_{q+1}\zeta_q\zeta_{q-1} + \psi_q(1-\zeta_q)\zeta_{q-1} + \psi_{q-1}(1-\zeta_{q-1})\Big]\Theta_1^{q-1}, \quad 1<q<n \\ I &= \Big[I_n\zeta_{n-1} + \psi_{n-1}(1-\zeta_{n-1})\Big]\Theta_1^{n-1} \quad (q=n) \end{aligned} \tag{9.7}$$

where it assumed that the LOS has n points, index 1 is the point closest to the sensor,

$$I_q = I_n\Theta_q^n + \sum_{i=q}^{n-1}\psi_i(1-\zeta_i)\Theta_q^i, \quad 1 \le q < n \tag{9.8}$$

is the intensity reaching point q along the LOS, I_n is the radiation at point n (the radiation entering the atmosphere), and

$$\Theta_q^p = \prod_{i=q}^{p-1}\zeta_i \quad \text{for} \quad p>q, \quad \text{and} \quad \Theta_p^p = 1 \tag{9.9}$$

the transmission from point q and p. It should be noted that I_q and Θ_q^p not are calculated as indicated by the equations above. These quantities are instead updated when going from one step of the LOS to the next, as described below. It should also be noted that ground reflections are here neglected and are discussed separately below.

The transmissions ζ_{q-1} and ζ_q are separated in Equation 9.7 as they are the only terms including the absorption at point q. For example

$$\zeta_{q-1} = e^{-\Delta l(\kappa_{q-1}+\kappa_q)/2} \tag{9.10}$$

and we have that

$$\frac{\partial\zeta_q}{\partial\kappa_q} = -\frac{\Delta l}{2}\zeta_q \tag{9.11}$$

$$\frac{\partial\zeta_{q-1}}{\partial\kappa_q} = -\frac{\Delta l}{2}\zeta_{q-1} \tag{9.12}$$

$$\frac{\partial\zeta_{q-1}\zeta_q}{\partial\kappa_q} = -\Delta l\zeta_{q-1}\zeta_q \tag{9.13}$$

The derivate of transmission values beside ζ_q and ζ_{q-1} with respect to κ_q is zero.

The LOS WFs are now easily determined, using the case $1<q<n$ as example

$$\mathbf{K}_\kappa^q = -\frac{\Delta l}{2}\Big[2I_{q+1}\zeta_q\zeta_{q-1} + \psi_q(1-2\zeta_q)\zeta_{q-1} - \psi_{q-1}\zeta_{q-1}\Big]\Theta_1^{q-1}, \; 1<q<n \tag{9.14}$$

which can be rewritten as

$$\begin{aligned} \mathbf{K}_\kappa^1 &= -\frac{\Delta l}{2}[I_2 - \psi_1]\Theta_1^2 \\ \mathbf{K}_\kappa^q &= -\frac{\Delta l}{2}[2(I_{q+1}-\psi_q)\zeta_q + \psi_q - \psi_{q-1}]\Theta_1^q, \quad 1<q<n \\ \mathbf{K}_\kappa^n &= -\frac{\Delta l}{2}[I_n - \psi_{n-1}]\Theta_1^n \end{aligned} \tag{9.15}$$

[1]The indexing used here is 1-based (starts at 1), while inside ARTS 0-based indexing is used.

Note that one ζ_q is incorporated in Θ_q^q, and that $\Theta_1^2 = \zeta_1$.

These equations are used for the practical calculations, but it could be of interest to note that Equation 9.15 can be written

$$\mathbf{K}_\kappa^q = -\frac{\Delta l}{2}[(I_{q+1} - \psi_q)\zeta_q + I_q - \psi_{q-1}]\Theta_1^q, \quad 1 < q < n, \tag{9.16}$$

showing that the expressions for $q = 1$ and $q = n$ are special cases of the general expression where the terms connected to $q - 1$ and q, are neglected, respectively.

The iteration starts here at the end closest to the sensor, that is, at index 1 (reversed order to the RTE part). The iteration is started by setting I_1 to the already calculated spectrum and Θ_1^1 to 1. These two variables are updated as

$$I_{q+1} = \frac{I_q - \psi_q(1 - \zeta_q)}{\zeta_q} \tag{9.17}$$

$$\Theta_1^{q+1} = \Theta_1^q \zeta_q \tag{9.18}$$

For 2D calculations possible ground reflections inside the LOS must be handled. The ground cannot be found at any of the end points of the LOS, and the correspondence to Equation 9.7 for a ground point is (c.f. Equations 5.12 and 5.15)

$$\begin{aligned} I \quad = \quad & \Big[I_{q+1}\zeta_q(1 - e)\zeta_{q-1} + \psi_q(1 - \zeta_q)(1 - e)\zeta_{q-1} + eB\zeta_{q-1} + \\ & +\psi_{q-1}(1 - \zeta_{q-1})\Big]\Theta_1^{q-1}, \quad 1 < q < n \end{aligned} \tag{9.19}$$

and the corresponding absorption LOS WF for this point is (cf. Eq. 9.15)

$$\mathbf{K}_\kappa^q \quad = \quad -\frac{\Delta l}{2}[2(I_{q+1} - \psi_q)\zeta_q(1 - e) + \psi_q(1 - e) + eB - \psi_{q-1}]\Theta_1^q \tag{9.20}$$

The intensity and the transmission are here updated as

$$\begin{aligned} I_{q+1} \quad &= \quad \frac{I_q - \psi_q(1 - \zeta_q)(1 - e) - eB}{\zeta_q(1 - e)} \\ \Theta_1^{q+1} \quad &= \quad \Theta_1^q \zeta_q(1 - e) \end{aligned}$$

It is noteworthy that the effect of a ground intersection is included in I_1 when the iteration starts.

9.2.2 1D limb sounding

For limb sounding and when the atmosphere is assumed to be consist of homogenous layers (horizontally stratified), there is a perfect symmetry around the tangent point. This covers also the case with a ground reflection. For these cases the distance from the sensor is neglected, the important factor is the vertical altitude. All altitudes above the tangent point are passed twice (Fig. 9.2) and both crossings of an atmospheric layer are treated to be identical for the retrievals, and this fact must also be reflected by the WFs.

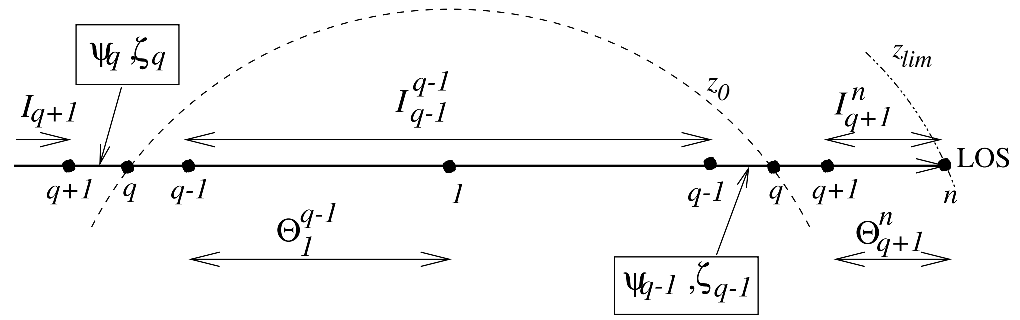

Figure 9.2: The terms used for the derivation of line of sight weighting functions for 1D limb sounding.

Using a nomenclature similar to the one used for Equation 9.7, the intensity of a limb sounding observations can be expressed as (Fig. 9.2)

$$
\begin{aligned}
I &= \Big(I_2\big(\zeta_1\Theta_1^1\big)^2 + \psi_1(1-\zeta_1)\big(\Theta_1^1\big)^2\zeta_1 + I_1^1\zeta_1 + \psi_1(1-\zeta_1)\Big)\Theta_2^n \quad (q=1) \\
I &= \Big[\Big(I_{q+1}\zeta_q\zeta_{q-1} + \psi_q(1-\zeta_q)\zeta_{q-1} + \psi_{q-1}(1-\zeta_{q-1})\Big)\Big(\Theta_1^{q-1}\Big)^2\zeta_{q-1}\zeta_q + \\
&\quad + I_{q-1}^{q-1}\zeta_{q-1}\zeta_q + \psi_{q-1}(1-\zeta_{q-1})\zeta_q + \psi_q(1-\zeta_q)\Big]\Theta_{q+1}^n,\ 1<q<n \qquad (9.21) \\
I &= \Big(I_n\zeta_{n-1} + \psi_{n-1}(1-\zeta_{n-1})\Big)\Big(\Theta_1^{n-1}\Big)^2\zeta_{n-1} + I_{n-1}^{n-1}\zeta_{n-1} + \\
&\quad + \psi_{n-1}(1-\zeta_{n-1}) \quad (q=n)
\end{aligned}
$$

where the expression for $q = 1$ is commented below, index 1 of the LOS is the tangent (or the ground) point, index n corresponds to the highest altitude,

$$
I_q = I_n\Theta_q^n + \sum_{i=q}^{n-1}\psi_i(1-\zeta_i)\Theta_q^{i-1} \qquad (9.22)
$$

is the intensity reaching point q from the part of the atmosphere furthest away from the sensor, I_n the intensity at point n,

$$
I_q^q = \Big[\sum_{i=1}^{q-1}(\psi_i(1-\zeta_i)\Theta_1^{i-1}\Big]\Theta_1^q + \sum_{i=1}^{q-1}\psi_i(1-\zeta_i)\Theta_{i+1}^q, \qquad q>1 \qquad (9.23)
$$

is the intensity generated along the LOS (towards the sensor) between the two crossing with altitude q, $I_1^1 = 0$, Θ is defined by Equation 9.9. The equations defining I_q, I_q^q and Θ neglect ground reflections, but could easily be extended to cover also such cases. However, I_1^1 and Θ_1^1 are included for $q = 1$ to make Equation 9.21 valid for cases with ground reflections. The treatment of ground reflections are discussed separately last in the section.

If the different combinations of ζ_{q-1} and ζ_q are grouped, for example, Equation 9.21 becomes

$$
\begin{aligned}
I &= \Big[\Big((I_{q+1}-\psi_q)\zeta_{q-1}^2\zeta_q^2 + (\psi_q-\psi_{q-1})\zeta_{q-1}^2\zeta_q + \psi_{q-1}\zeta_{q-1}\zeta_q\Big)\Big(\Theta_1^{q-1}\Big)^2 + \\
&\quad + (I_{q-1}^{q-1}-\psi_{q-1})\zeta_{q-1}\zeta_q + (\psi_{q-1}-\psi_q)\zeta_q + \psi_q\Big]\Theta_{q+1}^n \qquad (9.24)
\end{aligned}
$$

This equation has some higher products between ζ_{q-1} and ζ_q than Equation 9.7, and the derivatives, with respect to κ_q, of these product are

$$\frac{\partial\zeta_{q-1}^2\zeta_q}{\partial\kappa_q} = -\frac{3\Delta l}{2}\zeta_{q-1}^2\zeta_q \tag{9.25}$$

$$\frac{\partial\zeta_{q-1}^2\zeta_q^2}{\partial\kappa_q} = -2\Delta l\zeta_{q-1}^2\zeta_q^2 \tag{9.26}$$

Using Equations 9.11, 9.13, 9.25 and 9.26, the LOS WFs for 1D limb sounding can be determined to be

$$\begin{aligned}
\mathbf{K}_\kappa^1 &= -\frac{\Delta l}{2}\Big[\Big(2I_2\zeta_1 + \psi_1(1-2\zeta_1)\Big)\Big(\Theta_1^1\Big)^2 + I_1^1 - \psi_1\Big]\Theta_1^n \\
\mathbf{K}_\kappa^q &= -\frac{\Delta l}{2}\Big[\Big(4(I_{q+1}-\psi_q)\zeta_{q-1}\zeta_q + 3(\psi_q-\psi_{q-1})\zeta_{q-1} + 2\psi_{q-1}\Big)\Big(\Theta_1^{q-1}\Big)^2\zeta_{q-1} \\
&\quad +2(I_{q-1}^{q-1}-\psi_{q-1})\zeta_{q-1} + \psi_{q-1} - \psi_q\Big]\Theta_q^n, \quad 1 < q < n \qquad (9.27) \\
\mathbf{K}_\kappa^n &= -\frac{\Delta l}{2}\Big[\Big(2I_n\zeta_{n-1} + \psi_{n-1}(1-2\zeta_{n-1})\Big)\Big(\Theta_1^{n-1}\Big)^2\zeta_{n-1} + \\
&\quad +I_{n-1}^{n-1} - \psi_{n-1}\Big]\zeta_{n-1}
\end{aligned}$$

The function calculating these LOS WFs takes the total spectrum as input (that is, I_n^n) and it is then most suitable to iterate downwards, starting with point n. For each iteration, the quantities are updated as

$$I_q = I_{q+1}\zeta_q + \psi_q(1-\zeta_q)$$

$$\Theta_1^{q-1} = \frac{\Theta_1^q}{\zeta_{q-1}}$$

$$I_{q-1}^{q-1} = \frac{I_q - \psi_{q-1}(1-\zeta_{q-1})(1+(\Theta_1^{q-1})^2\zeta_{q-1})}{\zeta_{q-1}}$$

The iteration is started by setting I_n to cosmic background radiation, or correspondingly, and setting Θ_1^n to the square root of the total transmission. As mentioned above, I_n^n is an input to the function.

No special attention needs to be given here to possible ground reflections. This as the effects of a ground reflection are already included in I_n^n and Θ_1^n when starting the iteration. The procedure of setting Θ_1^n to the square root of the total transmission maintains the symmetry and makes it possible to treat the ground as an imaginary altitude "below" point 1. If there is a ground reflection, Θ_1^1 and I_1^1 equal $\sqrt{1-e}$ and eB, respectively, at the end of the iteration.

9.2.3 1D downward looking observations

Downward observation from an aircraft or a balloon can mainly be treated as a combination of limb sounding and upward looking observations. The altitudes below the platform altitude are covered by the limb sounding expressions with a suitable choice of I_q for the highest point. The altitudes above the platform altitude are treated by the upward looking equations, but also considering the transmission through the lower altitudes.

If q is the index for platform altitude, the intensity can be expressed as

$$\begin{aligned} I &= \Big(I_{q+1}\zeta_q\zeta_{q-1} + \psi_q(1-\zeta_q)\zeta_{q-1} + \psi_{q-1}(1-\zeta_{q-1})\Big)\Big(\Theta_1^{q-1}\Big)^2\zeta_{q-1} + \\ &\quad + I_{q-1}^{q-1}\zeta_{q-1} + \psi_{q-1}(1-\zeta_{q-1}) \end{aligned} \tag{9.28}$$

and the corresponding WF is

$$\begin{aligned} \mathbf{K}_\kappa^q &= -\frac{\Delta l}{2}\Big[\Big(3(I_{q+1}-\psi_q)\zeta_{q-1}\zeta_q + 2(\psi_q-\psi_{q-1})\zeta_{q-1} + \psi_{q-1}\Big)\Big(\Theta_1^{q-1}\Big)^2 + \\ &\quad + I_{q-1}^{q-1} - \psi_{q-1}\Big]\zeta_{q-1} \end{aligned} \tag{9.29}$$

9.3 Absorption LOS WFs for optical thicknesses

This section treats the absorption LOS WFs for cases when emission can neglected. For such pure absorption calculations the output of ARTS is optical thicknesses (instead of e.g. transmissions) and for these conditions the absorption LOS WFs get very simple.

9.3.1 Single pass

The optical thickness (τ) is for single pass cases (cf. Eq. 5.6)

$$\tau = \Delta l\left(\frac{\kappa_1+\kappa_2}{2} + \frac{\kappa_2+\kappa_3}{2} + \ldots + \frac{\kappa_{n-2}+\kappa_{n-1}}{2} + \frac{\kappa_{n-1}+\kappa_n}{2}\right) \tag{9.30}$$

and we have that

$$\begin{aligned} \mathbf{K}_\kappa^1 &= \Delta l/2 \\ \mathbf{K}_\kappa^q &= \Delta l, \quad 1<q<n \\ \mathbf{K}_\kappa^n &= \Delta l/2 \end{aligned} \tag{9.31}$$

9.3.2 1D limb sounding

For limb sounding each altitude is passed twice and the total optical thickness is double the optical thickness from the tangent point to the atmospheric limit. This fact results in that the absorption LOS WFs for 1D limb sounding are just the single pass ones multiplicated by two:

$$\begin{aligned} \mathbf{K}_\kappa^1 &= \Delta l \\ \mathbf{K}_\kappa^q &= 2\Delta l, \quad 1<q<n \\ \mathbf{K}_\kappa^n &= \Delta l \end{aligned} \tag{9.32}$$

9.3.3 1D downward looking observations

If q is the point where the sensor is placed, the optical thickness is

$$\tau = \Delta l \left(\frac{\kappa_q + \kappa_{q-1}}{2} + \ldots + \frac{\kappa_2 + \kappa_1}{2} + \ldots + \frac{\kappa_{q-1} + \kappa_q}{2} + \frac{\kappa_q + \kappa_{q+1}}{2} \ldots \right) \quad (9.33)$$

and the absorption LOS WF for this altitude is accordingly

$$\mathbf{K}_\kappa^q = \frac{3}{2}\Delta l \quad (9.34)$$

9.4 Source line of sight weighting functions

The source line of sight weighting functions are defined as

$$\mathbf{K}_\sigma^q = \frac{\partial \mathbf{i}}{\partial \sigma^q} \quad (9.35)$$

These weighting functions express how the intensity is affected by changes of the source function at the points of the line of sight. The source and absorption LOS WFs are tightly related and this section follows closely Section 9.2.

9.4.1 Single pass

As, for example,

$$\psi_q = \frac{\sigma_q + \sigma_{q+1}}{2} \quad (9.36)$$

the derivate of the mean source function values with respect to σ_q is

$$\frac{\partial \psi_{q-1}}{\partial \sigma_q} = \frac{\partial \psi_q}{\partial \sigma_q} = \frac{1}{2} \quad (9.37)$$

This derivate for other ψ terms is zero.

Using 9.7, the source LOS WFs for upward looking observations can be determined to be

$$\begin{aligned} \mathbf{K}_\sigma^q &= \frac{1-\zeta_1}{2}, \quad q = 1 \\ \mathbf{K}_\sigma^q &= \frac{1-\zeta_{q-1}\zeta_q}{2}\Theta_1^{q-1}, \quad 1 < q < n \\ \mathbf{K}_\sigma^q &= \frac{1-\zeta_{n-1}}{2}\Theta_1^{n-1}, \quad q = n \end{aligned} \quad (9.38)$$

For ground points in 2D calculations, the WFs are (cf. Eq. 9.19)

$$\mathbf{K}_\sigma^q = \frac{(1-\zeta_q)(1-e)\zeta_{q-1} + 1 - \zeta_{q-1}}{2}\Theta_1^{q-1}, \quad 1 < q < n \quad (9.39)$$

The practical calculations, such as the updating of Θ, follow the absorption LOS WFs (Sec. 9.2.1).

9.4.2 1D limb sounding

The 1D limb sounding source LOS WFs are (derived using Eq. 9.21)

$$\begin{aligned}
\mathbf{K}_\sigma^q &= \frac{1}{2}\Big(1-\zeta_1\Big)\Big(1+\Big(\Theta_1^1\Big)^2\zeta_1\Big)\Theta_2^n, \quad q=1 \\
\mathbf{K}_\sigma^q &= \frac{1}{2}\Big[(1-\zeta_{q-1}\zeta_q)\Big(\Theta_1^{q-1}\Big)^2\zeta_{q-1}\zeta_q+(1-\zeta_{q-1})\zeta_q+ \\
&\quad +1-\zeta_q\Big]\Theta_{q+1}^n, \quad 1<q<n \\
\mathbf{K}_\sigma^q &= \frac{1}{2}\Big((1-\zeta_{n-1})\Big(\Theta_1^{n-1}\Big)^2\zeta_{n-1}+1-\zeta_{n-1}\Big), \quad q=n
\end{aligned} \tag{9.40}$$

The practical calculations follow the absorption LOS WFs (Sec. 9.2.2).

9.4.3 1D downward looking observations

The source LOS WFs for downward looking observations are determined by the upward and the limb sounding expressions in the same manner as for the absorption LOS WFs (Sec. 9.2.3).

The LOS WF for the index corresponding to the platform altitude is (cf. Eq. 9.28) observations can be determined to be

$$\mathbf{K}_\sigma^q = \frac{1}{2}\Big[(1-\zeta_{q-1}\zeta_q)\Big(\Theta_1^{q-1}\Big)^2\zeta_{q-1}+1-\zeta_{q-1}\Big] \tag{9.41}$$

9.5 Transformation from vertical altitudes to distances along LOS

9.5.1 Basis functions

The source function and the absorption, both as a function of vertical altitude ($\mathbf{k}$) and along the LOS (κ), are assumed to vary linear between the points of the grid of concern. The functions to express the quantities between grid points are denoted as basis functions. For piecewise linear functions, the basis functions decline, from the point of interest, linearly down to zero at neighboring points. Such functions are here denoted as tenth functions (Fig. 9.3).

9.5.2 Transformation from z to l

The forward model uses internally a grid along the line of sight (Sec. 6), while the atmospheric WF matrices are calculated for some user specified vertical grid, and a transformation between these two grids must be performed. This transformation is achieved by the terms, $\partial\kappa/\partial\mathbf{k}^p$ and $\partial\sigma/\partial S^{\mathbf{p}}$. As the source function and the absorption are assumed to have the same functional behaviour (piecewise linear), these two terms are identical if the retrieval grid is the same for both quantities:

$$\frac{\partial\kappa}{\partial\mathbf{k}^p} = \frac{\partial\sigma}{\partial S^{\mathbf{p}}} \tag{9.42}$$

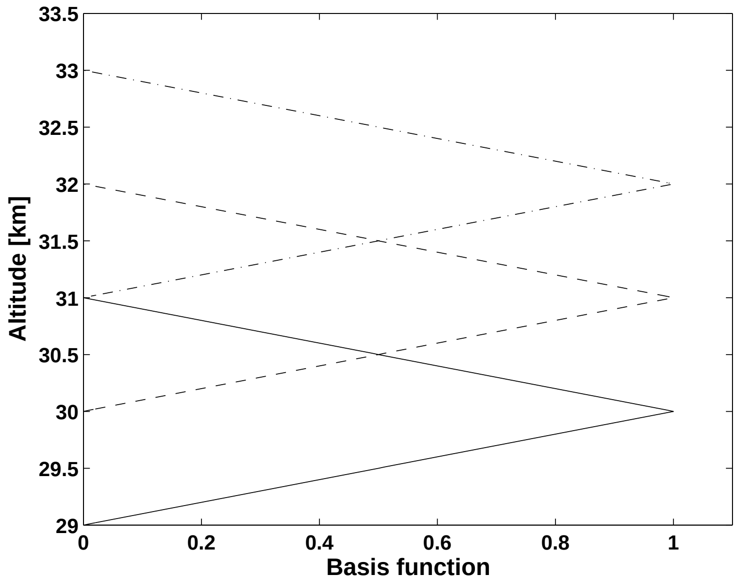

Figure 9.3: Examples on basis functions for a vertical grid with a 1 km spacing: —— 30 km, – – – 31 km and – · – 32 km.

For example, the term $\partial\kappa/\partial\mathbf{k}^p$ gives the relationship between the absorption along the LOS and a change of the absorption at one altitude. Figure 9.4 exemplifies $\partial\kappa/\partial\mathbf{k}^p$ for three altitudes. Ideally, the following relationship should be fulfilled for all z

$$\sum_i \mathbf{k}^i \phi^i_{\mathbf{k}}(z(l)) = \sum_j \kappa^j \phi^j_\kappa(l) \tag{9.43}$$

where $\phi_{\mathbf{k}}$ and ϕ_κ are the basis functions for $\mathbf{k}$ and κ, respectively. However, as can be seen in Figure 9.4, $\phi^i_{\mathbf{k}}$ expressed along the LOS is not a piecewise linear function and cannot be fitted perfectly by the basis ϕ_κ. Hence, some approximation is needed, and the most natural choice for this approximation is to fulfill Equation 9.43 only for the grid points along the LOS:

$$\kappa^q = \sum_i \mathbf{k}^i \phi^i_{\mathbf{k}}(z(\mathbf{l}^q)) \tag{9.44}$$

where $\mathbf{l}^q$ is the distance along the LOS for the corresponding to κ^q. Note that at $\mathbf{l}^q$ all ϕ^j_κ are zero except for ϕ^q_κ, that is unity.

We have now that

$$\frac{\partial\kappa^q}{\partial\mathbf{k}^p} = \phi^p_{\mathbf{k}}(z(\mathbf{l}^q)) \tag{9.45}$$

Hence, term $\partial\kappa/\partial\mathbf{k}^p$ is determined by the values of $\phi^p_{\mathbf{k}}$ at the altitudes corresponding to the grid points of the LOS.

Assuming that the LOS altitude q, z_{κ^q}, is found between retrieval points $p-1$ and p, at the altitudes $z_{\mathbf{k}^{p-1}}$ and $z_{\mathbf{k}^p}$, respectively, we have that

$$\frac{\partial\kappa^q}{\partial\mathbf{k}^p} = \frac{z_{\kappa^q} - z_{\mathbf{k}^{p-1}}}{z_{\mathbf{k}^p} - z_{\mathbf{k}^{p-1}}} \tag{9.46}$$

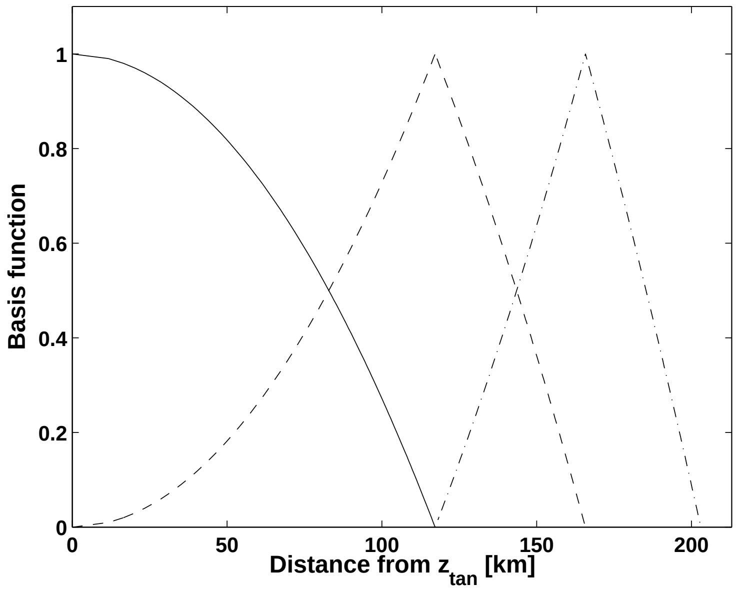

Figure 9.4: The basis functions of Figure 9.3 shown as a function of the distance from the tangent point, where $z_{tan} = 30$ km.

If z_{κ^q} is further away from $z_{\mathbf{k}^p}$ than the neighboring retrieval points, the derivative is zero. The derivative is also treated to be zero if z_{κ^q} is outside the retrieval grid (that is, below or above all retrieval altitudes).

The basis functions for $\mathbf{k}$ change if the retrieval grid is changed, and as the retrieval grid is individual for the species, temperature etc., the term $\partial\kappa/\partial\mathbf{k}^p$ must be determined for each calculation of a WF matrix.

9.6 Species WFs

As it is assumed here that the species have no influence on the source function, species WFs are calculated as (cf. Eq. 9.5)

$$\mathbf{K}_{\mathbf{x}}^p = \mathbf{H}\frac{\partial\mathbf{i}}{\partial\kappa}\frac{\partial\kappa}{\partial\mathbf{k}^p}\frac{\partial\mathbf{k}^p}{\partial\mathbf{x}^p} \tag{9.47}$$

The term $\partial\mathbf{i}/\partial\kappa$ is described in Section 9.2, while the term $\partial\kappa/\partial\mathbf{k}^p$ is treated in Section 9.5, and it remains to determine $\partial\mathbf{k}^p/\partial\mathbf{x}^p$. It is assumed below in this section that $\mathbf{x}$ only represents a single species and that the species absorption can be written as

$$\mathbf{k}^p = \bar{\mathbf{k}}_s^p\mathbf{x}^p + \sum_{i\neq s}\mathbf{k}_i^p \tag{9.48}$$

where p is the altitude of concern, $\bar{\mathbf{k}}_s$ is the absorption of the species of interest, normalized to the units of the corresponding values of $\mathbf{x}$ (or $\mathbf{b}$) and $\mathbf{k}_i$ the total absorption for other species. Equation 9.48 assumes that a change for one species does not influence the absorption of other species, and that the shape of the absorption for one species does not change with the abundance of that species. This assumption is not valid, for example, when the amount of different species must be considered when calculating the pressure broadening, and not only the total absorption. The validity of the analytical expressions for the WFs is discussed in Section 9.1.2.

If Equation 9.48 is valid, we have then that

$$\frac{\partial \mathbf{k}^p}{\partial \mathbf{x}^p} = \bar{\mathbf{k}}_s^p \tag{9.49}$$

Different units for species retrievals are allowed. The possible units are

1. Fractions of linearization state [-], i.e. $\mathbf{x}/\mathbf{x}_0$ where $\mathbf{x}_0$ is the linearization state
2. Volume mixing ratio [-] (no dimension)
3. Number density [molecules/m^3)

Accordingly, for the practical calculations, the absorption of the species of interest is needed, and a possibility to scale to the absorption from the unit used by the forward model to the other two units considered.

It is advantageous for the retrieval that the values of $\mathbf{x}$ are of similar magnitudes [*Schimpf and Schreier*, 1997; *Eriksson*, 1999] as the numerical precision is limited. This fact makes WFs in fractions of the linearization state (or rather, the a priori state) interesting as the values of $\mathbf{x}$ are then all around 1. In addition, Equation 9.49 is especially simple for this case:

$$\frac{\partial \mathbf{k}^p}{\partial \mathbf{x}^p} = \mathbf{k}_s^p \tag{9.50}$$

as $\mathbf{x}^p = 1$.

9.7 Continuum absorption WFs

These WFs are used to fit unknown absorption that varies smoothly inside the frequency range covered. This absorption is added to the species absorption:

$$\mathbf{k}^p = \mathbf{k}_s^p + \mathbf{k}_c^p \tag{9.51}$$

where $\mathbf{k}_s^p$ is the summed species absorption and $\mathbf{k}_s^p$ the continuum absorption.

The continuum absorption is represented by a polynomial for each altitude. The polynomials are characterized by the magnitude of the absorption at a number of points inside the frequency range covered (Fig. 9.5). This approach was selected as it gives the possibility to impose positive constraints in a straightforward manner. A direct polynomial representation ($k = k_0 + k_1\nu + k_2\nu^2...$) is less favorable regarding this aspect.

The number of points is $n_{cont} + 1$ where n_{cont} is the polynomial order selected. The points are equally spaced between the lowest and highest frequency, ν_{min} and ν_{max}, considered. Figure 9.5 exemplifies this for $n_{cont} = 2$. The points are accordingly placed at the following frequencies

$$\nu_i = \nu_{min} + \frac{(\nu_{max} - \nu_{min})(i-1)}{n_{cont}}, \quad 1 \le i \le (n_{cont} + 1) \tag{9.52}$$

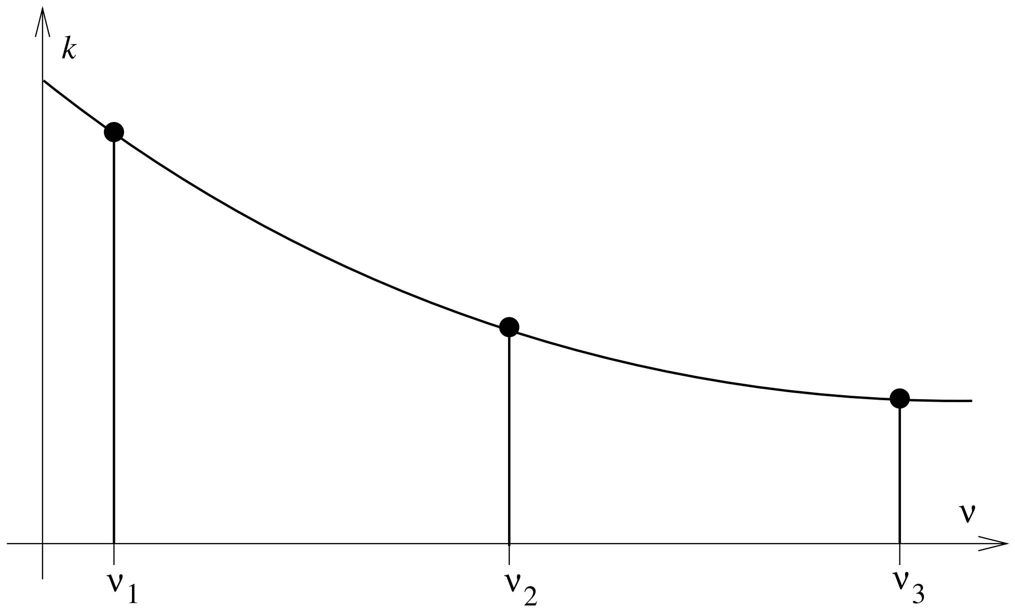

Figure 9.5: Fit of continuum absorption with off-sets at three positions ($n_{cont} = 2$). The outermost frequencies, here ν_1 and ν_3, are placed at the end points of the range covered (ν_{min} and ν_{max}, respectively).

This equation results in that the single point for $n_{cont} = 0$ is placed at ν_{min}, but the position of the frequency point is for this case of no importance as the corresponding WF is constant (as a function of frequency). With other words, if $n_{cont} = 0$, the WFs are simply

$$\frac{\partial \mathbf{k}^p}{\partial \mathbf{x}_1^p} = 1 \tag{9.53}$$

To determine the frequency dependency of the WFs for higher values of n_{cont}, the Lagrange's formula can be used. This formula gives the polynomial of order $N-1$ that passes through N fixed points [*Press et al.*, 1992, Eq. 3.1.1]:

$$\begin{aligned} k(\nu) \;=\; & \frac{(\nu-\nu_2)(\nu-\nu_3)\ldots(\nu-\nu_N)}{(\nu_1-\nu_2)(\nu_1-\nu_3)\ldots(\nu_1-\nu_N)}x_1 + \\ & + \frac{(\nu-\nu_1)(\nu-\nu_3)\ldots(\nu-\nu_N)}{(\nu_2-\nu_1)(\nu_2-\nu_3)\ldots(\nu_2-\nu_N)}x_2 + \cdots + \\ & + \frac{(\nu-\nu_1)(\nu-\nu_2)\ldots(\nu-\nu_{N-1})}{(\nu_N-\nu_1)(\nu_N-\nu_2)\ldots(\nu_N-\nu_{N-1})}x_N \end{aligned} \tag{9.54}$$

where x_i is the absorption at the selected frequency points, ν_i, that are given by Equation 9.52, and $N = n_{cont} + 1$.

The frequency dependency of the continuum WFs can be obtained by differentiating Equation 9.54:

$$\frac{\partial \mathbf{k}^p(\nu)}{\partial \mathbf{x}_i^p} = \frac{(\nu-\nu_1)\ldots(\nu-\nu_{i-1})(\nu-\nu_{i+1})\ldots(\nu-\nu_N)}{(\nu_i-\nu_1)\ldots(\nu_i-\nu_{i-1})(\nu_i-\nu_{i+1})\ldots(\nu_i-\nu_N)} \tag{9.55}$$

This equation gives, for example, for $n_{cont} = 1$

$$\frac{\partial \mathbf{k}^p(\nu)}{\partial \mathbf{x}_1^p} = \frac{\nu_{max} - \nu}{\nu_{max} - \nu_{min}}, \quad \nu_{min} \leq \nu \leq \nu_{max} \tag{9.56}$$

$$\frac{\partial \mathbf{k}^p(\nu)}{\partial \mathbf{x}_2^p} = \frac{\nu - \nu_{min}}{\nu_{max} - \nu_{min}}, \quad \nu_{min} \leq \nu \leq \nu_{max} \tag{9.57}$$

Note that these WFs have no altitude variation. Or with other words, they are identical for all p.

9.8 Temperature profile WFs

A critical factor for the calculation of temperature WFs is if hydrostatic equilibrium is assumed or not. If hydrostatic equilibrium is neglected, the WFs can be calculated by semi-analytical expressions, while if hydrostatic equilibrium is assumed, the WFs are obtained by perturbations. The analytical version is so far only implemented for emission measurements (and not for transmission measurements).

9.8.1 Without hydrostatic equilibrium

For some measurement situations it can be questionable to assume that the pressure, temperature and geometrical altitude, valid for the measurement, fulfill the law of hydrostatic equilibrium. One example is 1D limb sounding when there is a large horizontal distance between the nadir point of the tangent point for the start and end points of the scan. This is, for example, the case for the Odin observations where the tangent point will move in the latitude direction with a speed of about 9 km/s and a scan takes 1 – 2 minutes.

If the constrain of hydrostatic equilibrium is neglected, WFs for the temperature profile can be calculated following Equation 9.5, that is:

$$\mathbf{K}_{\mathbf{x}}^p = \mathbf{H}\left[\frac{\partial \mathbf{i}}{\partial \sigma}\frac{\partial \sigma}{\partial \mathbf{S}^p}\frac{\partial \mathbf{S}^p}{\partial \mathbf{t}^p} + \frac{\partial \mathbf{i}}{\partial \kappa}\frac{\partial \kappa}{\partial \mathbf{k}^p}\frac{\partial \mathbf{k}^p}{\partial \mathbf{t}^p}\right] \tag{9.58}$$

where $\mathbf{t}$ is the vector describing the vertical temperature profile.

The term $\partial\mathbf{i}/\partial\sigma$, the source LOS WFs, are derived in Section 9.4, while the absorption LOS WFs ($\partial\mathbf{i}/\partial\kappa$) are found in Section 9.2. As a single grid is here of concern, Equation 9.42 is valid, that is, $\partial\kappa/\partial\mathbf{k}^p$ equals $\partial\sigma/\partial S^{\mathbf{p}}$. These two terms are discussed in Section 9.5.

It is noteworthy that a change of the temperature inside an atmospheric layer will change the line-of-sights for beams passing this altitude, but this is here neglected. See further Section 9.1.2.

Here it is assumed that S equals the Planck function, B (Equation 5.2), and the derivative of the source function with respect to the temperature is (see also Equation 44 of *Eriksson et al.* [2000])

$$\frac{\partial S}{\partial T} = \frac{h\nu}{k_B T^2}\left(e^{h\nu/k_B T} - 1\right)^{-1} B(\nu, T) \tag{9.59}$$

The term $\partial\mathbf{S}^p/\partial\mathbf{t}^p$ is calculated using Equation 9.59 where T is replaced by $\mathbf{t}^p$.

The term $\partial\mathbf{k}^p/\partial\mathbf{t}^p$ cannot easily be determined analytically. Instead, the total absorption is calculated for a temperature profile that is 1 K higher at all altitudes than the assumed profile. The difference between the two absorption matrices are then interpolated to the temperature profile retrieval grid, giving an estimation of the derivative of the absorption with respect to the temperature at the grid altitudes. Schematically

$$\frac{\partial\mathbf{k}^p}{\partial\mathbf{t}^p} = \Upsilon(k(T_0+1) - k(T_0))$$

where Υ is the interpolating function from the vertical absorption grid to the retrieval grid, k the total absorption, and T_0 the assumed temperature profile.

9.8.2 With hydrostatic equilibrium

The gases in the atmosphere behave like an ideal gas, and the pressure, the temperature and the vertical altitudes above one point are linked by the fact that hydrostatic equilibrium must be fulfilled (see Section 6.7).

The temperature WFs with hydrostatic equilibrium are calculated by perturbations (Eq. 9.1). See further the on-line information (type `arts -d kTemp`).

9.9 Spectroscopic Parameters WFs

The spectroscopic parameters weighting functions are calculated semi-analytically as described in Section 9.1.2. As it is assumed that the spectroscopic parameters have no influence on the source function, the spectroscopic parameters weighting functions are calculated as (cf. Eq. 9.5)

$$\mathbf{K}^s_{\mathbf{b}} = \mathbf{H}\frac{\partial\mathbf{i}}{\partial\kappa}\frac{\partial\kappa}{\partial\mathbf{k}}\frac{\partial\mathbf{k}}{\partial\mathbf{b}} \tag{9.60}$$

The term $\partial\mathbf{i}/\partial\kappa$ is the LOS weighting function, described in Section 9.2, while the term $\partial\kappa/\partial\mathbf{k}$ gives weights of the absorption calculated for LOS grid points to the absorption calculated to vertical grid (described in Section 9.5). The vertical grid in this case is the pressure grid `p_abs}`used to calculate the absorption coefficients (see Section 3.1). The above two terms are easy calculated. The only term which remains to be calculated is $\partial\mathbf{k}/\partial\mathbf{b}$, where $\mathbf{b}$ is the spectroscopic parameter in concern (line intensity, line position, pressure broadening parameters and their temperature dependence, and pressure shift), specific to one line. Since the parameter $\mathbf{b}$ is characteristic for one single line, specific to one species, the total absorption can be written as:

$$\mathbf{k} = \bar{\mathbf{k}}_s + \sum_{i\neq s}\mathbf{k}_i \tag{9.61}$$

where $\bar{\mathbf{k}}_s$ is the absorption of the line of interest, and $\mathbf{k}_i$ is the absorption due to other lines.

Since the parameter of one specific line influence the the absorption of the line itself then one can write:

$$\partial\mathbf{k}/\partial\mathbf{b} = \partial\mathbf{k_s}/\partial\mathbf{b} \tag{9.62}$$

The term $\partial\mathbf{k}_s/\partial\mathbf{b}$ is calculated numerically. For each line, and each parameter the absorption is calculated once again by increasing the value of the parameter $\mathbf{b}$ by $\delta\mathbf{b}$. The difference in the two absorption is then calculated, and divided to the applied perturbation on the parameter. Schematically we have:

$$\frac{\partial \mathbf{k}}{\partial \mathbf{b}} = \frac{\partial \mathbf{k}^s}{\partial \mathbf{b}} = \frac{k(\mathbf{b} + \delta \mathbf{b}) - k(\mathbf{b})}{\delta \mathbf{b}}$$

Chapter 10

Measurement errors

Following Equation 2.2,

$$\mathbf{y} = \mathcal{F} + \varepsilon,$$

measurement errors, ε are here defined as errors that are additive to the spectrum, that is, not dependent on the actual spectrum. Error sources falling into this category are thermal noise and baseline ripples (there is a small influence of the magnitude of the spectrum on the thermal noise but this effect is normally totally negligible).

The term baseline ripple is used here as a common name for all instrumental imperfections causing a distortion of the spectra, for example, reflections inside the receiver, adding theoretically a sinusoidal term to the spectrum.

The content of this chapter is so far only handled by Qpack.

10.1 General

The sensor transfer matrix can be neglected when treating measurement errors as these errors are assumed to be additive to the spectra. On the other hand, a possible data reduction must be considered. This fact can also be understood by Equation 2.11:

$$\mathbf{y} = \mathbf{H}_d\mathbf{y}' = \mathbf{H}_d(\mathbf{H}_s\mathbf{i} + \varepsilon') = \mathbf{H}\mathbf{i} + \varepsilon$$

Using this equation, a measurement error WF can be written as

$$\mathbf{K}_{\mathbf{x}}^p = \frac{\partial \mathbf{y}}{\partial \mathbf{x}^p} = \frac{\partial \varepsilon}{\partial \mathbf{x}^p} = \mathbf{H}_d \frac{\partial \varepsilon'}{\partial \mathbf{x}^p} \tag{10.1}$$

Accordingly, quantities connected with the measurement errors shall be multiplicated with the data reduction matrix $\mathbf{H}_d$, this in contrast to the atmospheric WFs where the total reduction sensor matrix must be applied (Eq. 2.14).

History

000315 Created and written by Patrick Eriksson.

10.2 Thermal noise

The nature of the thermal noise differs from all other variables and error sources. The most distinct feature of the thermal noise is the low correlation between the measurements channels, in fact, the thermal noise is normally assumed to be totally uncorrelated. Such an assumption results in that a variable for each channel would be needed to model, or to fit, the measurement noise, and this is not a practical solution. In addition, it is not even of interest to know the actual magnitude of the thermal noise for each single measurement, we are instead interested in the statistical characteristics of the thermal noise. The special nature of the thermal noise has the consequence that this term is treated differently than the other variables, no weighting functions are calculated, only the covariance matrix is produced.

Thermal noise is introduced in two ways, by the observation of the atmosphere, and by the calibration process. The first part is here denoted as measurement thermal noise, while the latter is denoted as calibration thermal noise. In many cases, there is no practical difference between the two terms and they can together be treated as measurement thermal noise. However, if a single calibration measurement is used for a number of atmospheric spectra that are inverted jointly, as is the normal case for limb sounding, the error introduced by the calibration is totally correlated between the different viewing angles and it could be of importance to consider this fact.

10.2.1 Measurement thermal noise

The thermal noise is often assumed to be uncorrelated between the measurement channels, and the corresponding covariance matrix, $\mathbf{S}$ is then diagonal, where the diagonal elements are

$$\mathbf{S}_{tn}^{ii} = \left(\sigma_{tn}^{i}\right)^2 \tag{10.2}$$

where $\mathbf{S}^{ii}$ is element (i,i) of the matrix.

However, for most spectrometer types there exist in fact some correlation of the noise between the channels as there is an overlap of the channel frequency responses. The inter-channel correlation of the thermal noise can be modeled by three different correlation functions:

(1) gaussian

$$c^{ij} = exp\left(-\left(\frac{\nu_i - \nu_j}{f_c}\right)^2\right) \tag{10.3}$$

(2) exponential

$$c^{ij} = exp\left(-\frac{|\nu_i - \nu_j|}{f_c}\right) \tag{10.4}$$

and (3) tenth

$$\begin{aligned} c^{ij} &= 1 - \frac{|\nu_i - \nu_j|(1-e^{-1})}{\nu_c}, \quad |\nu_i - \nu_j| < \frac{\nu_c}{(1-e^{-1})} \\ c^{ij} &= 0, \quad |\nu_i - \nu_j| \geq \frac{\nu_c}{(1-e^{-1})} \end{aligned} \tag{10.5}$$

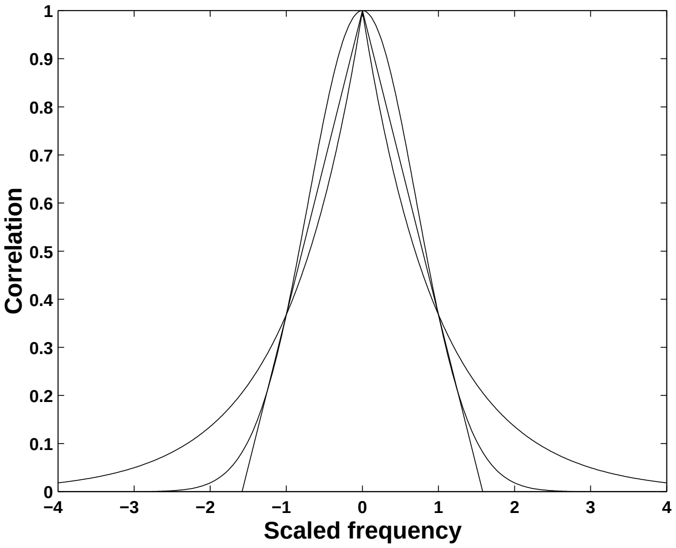

Figure 10.1: The frequency correlation functions. The frequency is scaled to the correlation length as $(\nu_i - \nu_j)/\nu_c$.

where ν_c is the frequency distance where the correlation has declined to e^{-1}, the frequency correlation length, and ν_i the middle frequency of channel i (Fig. 10.1). The numbers given above are used to select the correlation function when specifying the the covariance matrix for thermal noise. A diagonal matrix is flagged as correlation function 0. It is also possible to apply a threshold for the correlation, where all c^{ij} below the threshold value are set to 0.

The covariance matrix for one viewing angle with inter-channel correlation is

$$\mathbf{S}_{tn}^{ij} = c^{ij}\sigma_{tn}^{i}\sigma_{tn}^{j} \tag{10.6}$$

The correlation between different viewing angles is set to 0.

To include the effect of data reduction, the covariance matrix is multiplicated with $\mathbf{H}_d$ as

$$\mathbf{S}_{tn} = \mathbf{H}_d\mathbf{S}'_{tn}\mathbf{H}_d^T \tag{10.7}$$

where $\mathbf{S}'_{tn}$ is the covariance matrix before data reduction.

10.2.2 Calibration thermal noise

In contrast to the measurement thermal noise, the calibration thermal noise is assumed to be totally correlated between the different viewing angles. This latter noise as assumed to be identical between the channels. The correlation functions used for the measurement thermal noise can also be applied for the calibration thermal noise. Data reduction is considered by Equation 10.7.

10.3 Polynomial baseline ripple

A polynomial representation of the baseline ripple can be suitable at many occasions. One example is when a sinusoidal baseline ripple has a period that exceeds significantly the total frequency coverage of the receiver and the exact period length is not known. A baseline

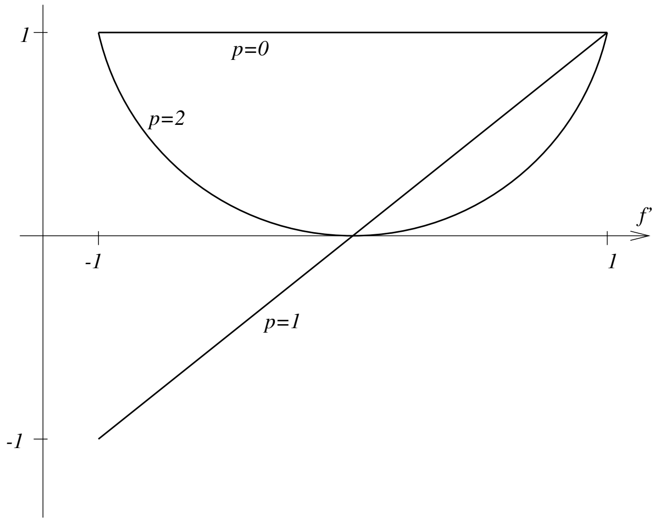

Figure 10.2: Polynomial WFs of order 0, 1 and 2. The scaled frequency is $f' = (\nu - \bar{\nu})/\Delta\nu$.

polynomial can also be used to fit continuum absorption for linear situations, e.g. to fit the unknown emission from the troposphere for ground-based observations.

The polynomial measurement error is modeled as

$$\varepsilon_{pol} = x_0 + \sum_{i=1}^{n_{pol}} x_i \left(\frac{\nu - \bar{\nu}}{\Delta\nu} \right)^i \tag{10.8}$$

where n_{pol} is the polynomial order selected, x_i are the polynomial coefficients to be determined, and $\bar{\nu}$ and $\Delta\nu$ normalization factors. The part of $\mathbf{x}$ corresponding to the polynomial fit of the baseline is accordingly

$$\mathbf{x} = \begin{bmatrix} \vdots \\ x_0 \\ x_1 \\ \vdots \\ x_{n_{pol}} \\ \vdots \end{bmatrix} \tag{10.9}$$

The normalization factors are needed to avoid extreme values (without the factors the quantity ν^i would have been calculated), resulting in that the magnitudes of the coefficients x_i will not deviate too strongly. The factors are calculated as

$$\bar{\nu} = \frac{\nu_{min} + \nu_{max}}{2} \tag{10.10}$$

$$\Delta\nu = \frac{\nu_{max} - \nu_{min}}{2} \tag{10.11}$$

where ν_{min} and ν_{max} are the minimum and maximum value, respectively, of the frequency grid given by the spectrometer. These definitions of the normalization factors give a scaled frequency grid extending from -1 to 1.

The polynomial WFs are

$$\mathbf{K}_{\mathbf{x}}^{p} = \mathbf{H}_d \mathbf{a}_p \tag{10.12}$$

where the elements of $\mathbf{a}_p$ are

$$\mathbf{a}_p^i = \left(\frac{\nu^i - \bar{\nu}}{\Delta\nu} \right)^p \tag{10.13}$$

Note that for $p = 0$, $\mathbf{a}_p = 1$.

Examples on polynomial weighting functions are shown is Figure 10.2.

10.4 Piecewise polynomial baseline ripple

If the spectrum is recorded with a number of spectrometers (or individual spectrometer parts) there could be a difference in the level between the different parts of the spectrum. Figure 10.3 shows an example on such a spectrum.

The baseline for such cases can be retrieved by piecewise polynomials where an individual polynomial is applied for each part of the spectrum. For frequencies inside the part of concern the WFs are given by Equation 10.13, while for remaining frequencies the WFs are 0.

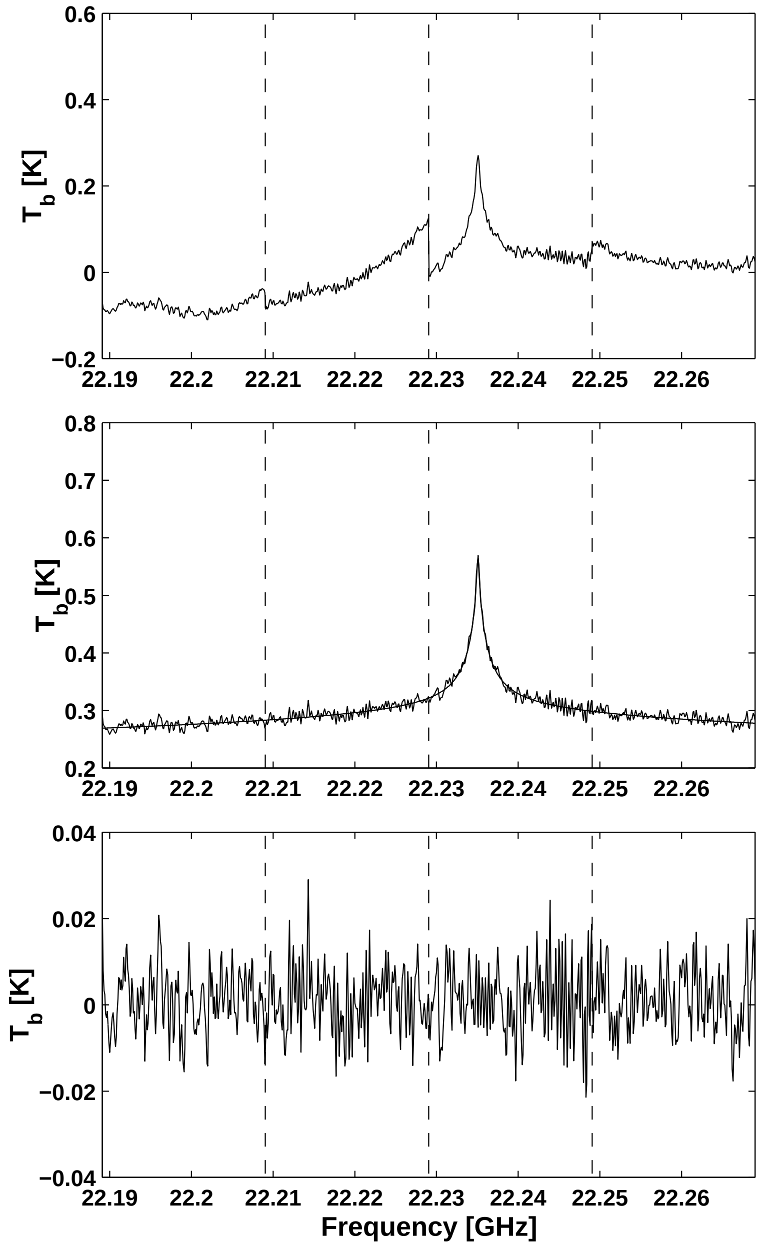

Figure 10.3: Example on fit of baseline with piecewise polynomials. The top figure shows a (poor!) test measurement with the 22.2 GHz water vapor radiometer at Onsala Space Observatory, Sweden. The spectrum was recorded by an auto-correlator spectrometer having four 20 MHz wide individual parts, clearly seen in the spectrum. The middle figure shows the measurement spectrum after a correction based on the retrieved baseline variables, and the simulated spectrum corresponding to the retrieved profile. The baseline is fitted by 3:rd order polynomial over the whole frequency range, and a 2:nd oder polynomial inside each 20 MHz range. The lower figure shows the difference between the spectra in the middle figure, the residual.

Part II

Implementation Issues

Chapter 11

The art of developing ARTS

This section is supposed to become the ARTS developers manual one day. Its aim is to describe how the program is organized and to give detailed instructions how to make extensions.

11.1 Organization

ARTS is written in C++ with the help of the GNU development tools (Autoconf, Automake, etc.). It is organized in a similar manner as most GNU packages. The top-level ARTS directory is either called `arts` or `arts-x.y`, where x.y is the release number. It contains various sub-directories, notably `doc` for documentation, `src` for the C++ source code, `ami` for the MATLAB interface, and `aii` for the IDL interface. The document that you are reading right now, the ARTS User Guide, is located in `doc/uguide`.

There are two different versions of the ARTS package: The developers version and the end-user version. Both contain the complete source code, the only difference is that the developers version also includes the CVS housekeeping data. If you want to join in the ARTS development (which we of course encourage you to do), you should write an email to the authors to obtain access to the developers version, which makes it easier to merge your changes with the 'official' ARTS program. Furthermore, for serious development work you need a computer running Unix, the GNU development tools, LaTeX, and the Doxygen program. All this is freely and easily available on the Internet, and, what is more, all these tools are included in the Suse linux distribution. (Most likely they are also included in the Redhat distribution, but I did not check.)

The end-user version contains everything that you need in order to compile and install ARTS in a fairly automatic manner. The only thing you should need is an ANSI-C++ compiler and the standard Unix `make` utility. Please see files `arts/README` and

History

011005	Stefan Buehler: Fixed TeX warnings, updated.
000728	Stefan Buehler: Added stuff about build system and howto cut a release.
000615	Created by Stefan Buehler. For now, this is basically the former content of the file `notes.txt`.

`arts/INSTALL` for installation instructions. We are developing with the GNU C++ compiler, no other compilers have been tried so far.

11.2 The ARTS build system

As mentionend above, GNU tools are used to construct the ARTS build system. A good introduction to the GNU build system can be found in:

`http://www.amath.washington.edu/~lf/tutorials/autoconf/`

Using these tools makes a lot of things very easy, but also some things slightly more complicated.

The most important thing to keep in mind is that an ARTS release is not just a copy of the ARTS development tree. Instead there is a special make target 'dist' that you can use to cut a release. How this is done in detail is described in Section 11.5.4. Mostly, the GNU tools are smart enough to figure out automatically what should go into the release. However, this can be controlled by editing the Makefile.am files which can be found in almost all directories.

The support for documentation other than info and man pages is not very good in the GNU system, so we had to use some tricks to make sure that the Doxygen automatic documentation and the User Guide work as they should.

If you add directories or just files, you have to make sure that they also go into the distribution. For program source code files, this is done automatically. **But if you add any other kind of file, for example a data or a documentation file, you have to edit the `Makefile.am` file in that directory to make sure that your stuff goes into the distribution.** It is a good idea to always check the release in order to see if the things you added are really there.

11.3 Conventions

Here are some general rules for ARTS programming:

11.3.1

Never use `float` or `double` explicitly, use the type `Numeric` instead. This is set by `configure` (to `double` by default). Thus, it is possible to compile the program for `float` by simply running configure with a different option. In the same way, use Index for all integers. It can take on positive or negative values.

11.3.2

Use `Vector` and `Matrix` for mathematical vectors and matrices (with elements of type `Numeric`). Use `Array<something>` to create an array of `somethings`. Commonly used Arrays have been predefined, they have names like `ArrayOfString`, `ArrayOfMatrix`, and so forth.

11.3.3 Terminology

Calculations are carried out in the so called workspace (WS), on workspace variables (WSVs). A WSV is for example the variable containing the absorption coefficients. The WSVs are manipulated by workspace methods (WSMs). The WSMs to use are specified in the controlfile in the same order in which they will be executed.

11.3.4 Global variables

Are not visible by default. To use them you have to declare them like this:

```
extern const Numeric PI;
```

which will make the global constant PI=3.14... available. Other important globals are:

`full_name`	Full name of the program, including version.
`parameters`	All command line parameters.
`basename`	Used to construct output file names.
`out_path`	Output path.
`messages`	Controls the verbosity level.
`wsv_data`	WSV lookup data.
`wsv_group_names`	Lookup table for the names of *types* of WSVs.
`WsvMap`	The map associated with `wsv_data`.
`md_data`	WSM lookup data.
`MdMap`	The map associated with `md_data`.
`workspace`	The workspace itself.
`species_data`	Lookup information for spectroscopic species.
`SpeciesMap`	The map associated with `species_data`.

The only exception from this rule are the output streams `out0` to `out3`, which are visible by default.

11.3.5 Files

Always use the `open_output_file` and `open_input_file` functions to open files. This switches on exceptions, so that any error occurring later on with this file will result in an exception. (Currently not really implemented in the GNU compiler, but please use it anyway.)

11.3.6 Version numbers

The package version number is set in file `configure.in` in the top level ARTS directory. Always increase this when you do a CVS commit, even for small changes. in such cases increase the last digit by one. If you make a new distribution, increase the middle digit by one and omit the last digit. If you make a bug-fix distribution, you can add the last digit to indicate this.

11.3.7 Header files

The global header file `arts.h` *must* be included by every file. Apart from that you have to see yourself what header files you need. If you use functions from the C or C++ standard library, you have to also include the apropriate header file.

11.3.8 Documentation

Doxygen is used to generate automatic sourec code documentation. See

`http://www.stack.nl/~dimitri/doxygen/`

for information. There is a complete User manual there. At the moment we only generate the output as HTML, although latex, man-page, and rtf format is also possible. The HTML version is particularly useful for source code browsing, since it includes the complete source code! You should add doxygen headers to the following:

1. Files
2. Classes (Including all private and public members)
3. Functions
4. Global Variables

The documentation headers are comment blocks that look like the examples below. They should be put above the *definition* of a function, i.e., in the `.cc` file. Some functions are defined in the `.h` file (e.g., inline member functions). In that case the comment can be put in the `.h` file. **The first sentence will be used as a short description for the entity, so it should be explanatory.**

There are some emacs macros that insert these comment blocks automatically. You can find them in the ARTS distribution in `doc/emacs`. Documentation on the macros can be found in `doc/index.html`.

File comment:

```
/**
   \file   dummy.cc

   A dummy file.
   This file has no purpose at all,
   it just servers as an example...

   \author Stefan Buehler
   \date   2000-09-13
*/
```

Function comment:

The emacs macro here inserts only `\param` for all arguments. If arguments are modified by the function you should change this to `\retval`.

```
/**
   A dummy function.
   This function has no purpose at all,
   it just serves as an example...

   \retval a This parameter is modified by the function.
   \param  b This is the other input parameter.
   \return A dummy value computed from a and b.

   \author Stefan Buehler
   \date   2000-09-13
*/
int dummy(int& a, int b);
```

Generic comment:

```
/**
   This is a dummy comment. You can
   write as much as you want here...
*/
```

11.4 Extending ARTS

11.4.1 How to add a workspace variable

1. Create a record entry in file `workspace.cc`. (Just add another one of the `wsv_data.push_back` blocks.) Take the already existing entries as templates. The ARTS concept works best if WSVs are only of a rather limited number of different types, so that generic WSMs can be used extensively, for example for IO.

   The name must be *exactly* like you use it in the source code, because this is used to generate interface functions.

   Make sure that the documentation string you give explains the variable and its purpose well. **In particular, state the dimensions (in the case of matrices) and the units!** This string is used for the online documentation. Please take some time to write it carefully.

2. That's it!

11.4.2 How to add a workspace variable group

1. Add a `wsv_group_names.push_back("your_type")` function to the function `define_wsv_group_names()` in `groups.cc`. The name must be *exactly*

like you use it in the source code, because this is used to generate interface functions.

2. That's it! (But as stated above, use this feature wisely)

11.4.3 How to add a workspace method

1. Create an entry in the function `define_md_data` in file `methods.cc`. (Make a copy of an existing entry (one of the `md_data.push_back(...)` blocks) and edit it to fit your new method.) Don't forget the documentation string! Please refer to the example at the beginning of the file to see how to format it.

2. Run: `make`.

3. Look in `auto_md.h`. There is a new function prototype

   ```
   void <YourNewMethod>(...)
   ```

4. Add your function to one of the `.cc` files which contain method functions. Such files must have names starting with `m_`. (See separate HowTo if you want to create a new source file.) The header of your function must be compatible with the prototype in `md.h`.

5. Check that everything looks nice by running

   ```
   arts -d YourNewMethod
   ```

   If necessary, change the documentation string.

6. Thats it!

11.4.4 How to add a source code file

1. Create your file. Names of files containing workspace methods should start with `m_`.

2. You have to register your file in the file `src/Makefile.am`. This file states which source files are needed for arts. Should be self-explanatory where you have to add your file. The above goes for source (`.cc`) and header (`.h`) files likewise.

3. Then go to the top level arts directory and run: `autogen.sh`.

4. Go to `src` and run: `cvs add <my_file>` to make your file known to CVS.

11.4.5 How to add an example file

1. Create your own example file. The filename should end with `_example.arts.in`.

2. If your example uses files from the arts-data package, replace the path to the data package (e.g. `/pool/lookup2/arts-data`) with `@ac_arts_data@`. Configure will replace this with the correct path.

3. Add your file to the variable `arts_examples` in the file `doc/examples/Makefile.am`.

4. Add your file to the AC_OUTPUT list near the end of configure.in.

5. The next time when you call make the `.arts.in` file will be automatically converted to `.arts`.

11.5 CVS issues

The arts project is controlled by CVS. This section describes some basic CVS commands. For more information see the extensive CVS documentation or our own CVS Howto on:

`http://www.sat.uni-bremen.de/docs/`

11.5.1 How to check out arts

1. Go to a temporary directory.

2. Run: `cvs co -P arts`.

11.5.2 How to update (if you already have a copy)

1. Go to the top ARTS directory (called simply `arts`).

2. Run: `cvs update -P`

   IMPORTANT! Always update, before you start to make changes to the program, especially after a longer pause. If you edit an outdated copy, it will be a lot more work to bring your changes into the current copy of the program.

11.5.3 How to commit your changes

1. You should make sure that the program compiles and runs without obvious errors before you commit.

2. If you have created a new source file, make it known to CVS by running the command `cvs add <my_file>` in the directory where the file resides.

   In general, when you run `cvs update`, it will warn you about any files it doesn't know by marking them with a `?`. Files that are created during the compilation process, but should not be part of the package are listed in the `.cvsignore` files in each directory.

3. Have you added the documentation for your new features?

4. Increase the subversion number in file `configure.in` in the top level ARTS directory.

5. Open the file `ChangeLog` in the top level ARTS directory with your favorite editor.

   With Emacs, you can very easily add an entry by typing either

   ```
   M-x add-change-log-entry
   ```

or `C-x 4 a`.

Specify the new version number and describe your changes.

These keystrokes work also while you are editing some other file in Emacs. Thus it is best to write your ChangeLog entry already while you work on a file. Whenever you make a change to a file, there should be a Changelog Entry!

6. Make sure that you have saved all your files. Go to the top level ARTS directory and run: `cvs commit`.

7. This will pop up an editor. Use the mouse to cut and paste the Change-Log message also to this editor window. Safe the file and exit the editor. If you made changes in different directories, another editor will pop up, already containing your message. Save again and exit. Do this until no more editors come up. (Note: This works well if you set

   ```
   export EDITOR=xedit
   ```

   in you shell startup file.

   With smarter editors there might be problems, because they might refuse to safe your file if you haven't made changes to it. So you would have to add a blank to the message each time a new directory is commited.)

8. You have to give your version of the program a symbolic name, so that it can be retrieved later on if necessary. Do this by running: `cvs tag arts-x-y-z` where x,y,z must be replace by the version numbers. You have to use dashes to separate the numbers, a point (`.`) will not work.

9. Tell the other developers about it. The best way to do this is to send an email to `arts-dev@sat.physik.uni-bremen.de`.

11.5.4 How to cut a release

1. Change the release number in the file `configure.in` in the top-level ARTS directory. (The line that you have to change is the one with `AM_INIT_AUTOMAKE`.) Omit the subversion number (last digit).

2. Commit your changes (see other howto).

3. In the top-level ARTS directory, run `autogen.sh`.

4. In the top-level ARTS directory, run `make distcheck`. This will not only cut the release, but also immediately try to build it, to see if it works. Unless you are on a very fast machine, this may take a while. Maybe you should go and have a cup of coffee.

5. If all goes well, you can find the release inside the top-level ARTS directory as a file `arts-x.y.tar.gz`, where x.y is the release number.

6. Check the release carefully by trying to build and install the program.

11.5.5 How to move your arts working directory

Never try to move CVS directories! Instead:

1. Commit your changes.
2. Go *above* the top level ARTS directory.
3. Run: `cvs release -d arts`.
   This will ask for confirmation, and if you say `y` delete your working copy of arts.
4. Go to the directory where you want to have your ARTS copy in the future.
5. Check out a new copy (see other howto above).

11.6 Debugging (use of assert)

This section is taken more or less literally from the GNU tools manual of Eleftherios Gkioulekas:

```
http://www.amath.washington.edu/~lf/tutorials/autoconf/
```

The idea behind assert is simple. Suppose that at a certain point in your code, you expect two variables to be equal. If this expectation is a precondition that must be satisfied in order for the subsequent code to execute correctly, you must assert it with a statement like this:

```
assert(var1 == var2);
```

In general assert takes as argument a boolean expression. If the boolean expression is true, execution continues. Otherwise the `abort` system call is invoked and the program execution is stopped. If a bug prevents the precondition from being true, then you can trace the bug at the point where the precondition breaks down instead of further down in execution or not at all. The `assert` call is implemented as a C preprocessor macro, so it can be enabled or disabled at will. One way to enable assertions is to include `assert.h`.

```
#include <assert.h>
```

Then it's possible to disable them by defining the 'NDEBUG' macro.

During debugging and testing it is a good idea to leave assertions enabled. However, for production runs it's best to disable them. If your program crashes at an assertion, then the first thing you should do is to find out where the error happens. To do this, run the program under the `gdb` debugger. First invoke the debugger:

```
gdb
```

Then load the executable and set a breakpoint at the `exit` system call:

```
(gdb) file arts
(gdb) break exit (or break __assert_fail)
```

Now run the program:

```
(gdb) run
```

Instead of crashing, under the debugger the program will be paused when the `exit` system call is invoked, and you will get back the debugger prompt. Now type:

```
(gdb) where
```

to see where the crash happened. You can use the `print` command to look at the contents of variables and you can use the `up` and `down` commands to navigate the stack. For more information, see the GDB documentation or type `help` at the prompt of gdb.

For ARTS, the assertion failures mostly happen inside the Matrix / Vector package (usually because you triggered a range check error, i.e., you tried to read or write beyond array bounds). In this case the `up` command of GDB is particularly useful. If you give this a couple of times you will finally end up in the part of your code that caused the error.

Recommendation: In Emacs there is a special GDB mode. With this you can very conveniently step through your code.

Chapter 12

Vectors, matrices, and arrays

This section describes how vectors and matrices are implemented in ARTS and how they are used. Furthermore it describes how arrays of arbitrary type can be constructed and used.

12.1 Implementation files

The `Matrix` and `Vector` classes described below reside in the files:

- `matpackI.h`
- `make_vector.h`
- `matpackI.cc`
- `make_vector.cc`

There is also a file `matpackII.h`, which contains the draft of a sparse matrix class, but this is at the moment not used. The template class `Array` (also described below) is implemented in the files:

- `array.h`
- `make_array.h`

The file `test_matpack.cc` contains test cases and usage examples.

12.2 Vectors

The class `Vector` implements the mathematical concept of a vector. (Surprise, surprise.) This means that:

- A Vector contains a list of floating point values of type `Numeric`.

History
011018 Created and written by Stefan Buehler.

- A Vector can be multiplied with another Vector (scalar product), or with a Matrix.
- Sub-ranges of a Vector can easily be accessed, and used as if they were Vectors.
- Resizing a Vector is expensive and should be avoided.

12.2.1 Constructing a Vector

You can construct an object of class Vector in any of these ways:

```
Vector a;          // Create empty Vector.
Vector b(3);       // Create Vector of length 3, if
                   // created like this it will contain
                   // arbitrary values.
Vector c(3,0.0);   // Create Vector of length 3, and
                   // fill it with 0.

Vector d=c;        // Make d a copy of c.

Vector e(1,5,1);   // 1, 2, 3, 4, 5
Vector f(1,5,.5);  // 1, 1.5, 2, 2.5, 3
Vector g(5,5,-1);  // 5, 4, 3, 2, 1
```

The last three examples all use the same constructor, which takes the three arguments 'start', 'extent', and 'stride'. It will create a Vector containing 'extent' elements, starting with 'start', with a step of 'stride'.

There also exists a special sub-class of Vector that can be initialized explicitly. This must be a special class in order to avoid ambiguities with the standard constructors. Usage:

```
MakeVector a(1.0,2.0,3.0);   // Creates a vector of length 3
                             // containing the values
                             // 1.0, 2.0, and 3.0.
```

You can use MakeVectors just like Vectors, except that the constructors are different. Otherwise you can mix them freely with Vectors.

12.2.2 VectorViews

An object of class `VectorView` is, like the name says, just another view on an existing Vector. It does not have its own data. This has the important consequence that it cannot be resized, since that would mess up the original Vector that the view is referring to. You can create VectorViews from Vectors using the index operator '[]', the class `Range`, and the special `joker` object. Examples:

```
MakeVector x(1,2,3,4,5,6,7);
VectorView a = x;                 // Now a refers to the
                                  // whole of x;
VectorView b = x[Range(joker)];   // Same effect.
```

```
VectorView c = x[Range(0,2)];      // Take 2 elements of x,
                                   // starting at the
                                   // beginning,
                                   // in this case: 1,2.
VectorView d = x[Range(0,3,2)];    // In this case: 1,3,5.
VectorView e = x[Range(3,joker)]; // In this case: 4,5,6,7.
```

As you can see, most useful ways to create VectorViews involve the Range class. The general constructor to this class takes three arguments, 'start', 'extent', and 'stride'. This means that you will select 'extent' elements from the Vector, starting with index 'start', with a step-width of stride. Note that indices are 0-based, so 0 refers to the first element. The last argument, 'stride', can be omitted, in that case the default of 1 is assumed. As a special case, 'extent'==`joker` means 'to the end', and calling Range with only one argument `joker` means 'all elements'.

Usually, you will not have to use VectorView explicitly, because you can use expressions like:

```
Vector a(1,5,1);                // a = 1,2,3,4,5
Vector b = a[Range(1,3)];       // b = 2,3,4
```

However, `VectorView` and the related class `ConstVectorView` are extremely useful as the argument types of functions operating on Vectors. You should define your functions like this:

```
void silly_function(VectorView a,       // Output argument
                    ConstVectorView b   // Input argument
                                        // (read only)
                   )
{
   // Do some silly stuff with a and b.
}
```

Note that there must not be any '&' after VectorView or ConstVectorView. In other words they have to be passed by value, not by reference. This is ok, since they do not contain the actual data, so that passing by value is efficient. Passing VectorViews by reference is forbidden.

You should use these kind of arguments for all input Vectors, and also for the output if you have a function that does not resize the output Vector. This has the great advantage that you can call the function with Vector sub-ranges, e.g.,

```
Vector a(1,5,1);                   // a = 1,2,3,4,5
Vector b(3);                       // Set size of b.
silly_function(b,a[Range(0,3)]);   // Call fuction with
                                   // sub-range of a.
```

An exception to this rule are workspace methods, which use conventional argument types `const Vector&` for input and `Vector&` for output.

12.2.3 What you can do with a Vector (or VectorView)

All examples below (except for the first) assume that `a` is a Vector, MakeVector, or VectorView.

Resize (only for Vector, not for VectorView!):

```
a.resize(5);
```

This makes `a` a 5 element vector. The new Vector is not initialized (i.e., the contents will be unpredictable). Also, note that the previous content will be completely lost. Appending to a Vector is not possible.

Get the number of elements:

```
cout << a.nelem();
```

Sum up all elements:

```
cout << a.sum();
```

Element access:

```
cout << a[3];    // Print 4th element.
a[0] = 3.5;      // Assign 3.5 to first element.
```

Note that we use 0-based indexing! Furthermore note that the operator '[]' can be also used with `Range`, as explained above.

Copying Vectors:

```
Vector b;
b.resize(a.nelem());
b = a;
```

This one is a bit tricky. The '=' operator always copies the *contents*, so the sizes of the left-hand and right-hand argument must match. Vector internally uses assertions to make sure of this. So, if you get an assertion failure one reason could be that you forgot to make b the correct size.

The reason for this behavior is to preserve computational efficiency. Resizing Vectors is expensive, so it is never done automatically. Another reason is, that this way sub-ranges can be used to full advantage, e.g.:

```
b[Range(5,5,-1)] = a[Range(3,5)];  // Copy 5 elements from
                                   // a to b, reversing
                                   // the order and starting
                                   // with index 3 in a.
```

Maybe you have noticed that there is a way to formulate the first example above in a much shorter fashion:

```
Vector b = a;
```

The result is exactly the same. Note, though, that in this case b is *constructed* from a, not copied (see section about constructing Vectors above). The Vector b is just generated in this case, therefore its size can be adjusted to that of a automatically.

Assigning a scalar:

```
a = 1.0;                          // Assign 1 to all elements.
```

Mathematical operators:

```
Vector a(1,3,1), b(3,1); // a = 1,2,3; b = 1,1,1
a *= 2;                  // a = 2,4,6
                         // Similarly, /=, +=, -=
a += b;                  // a = 3,5,7
                         // Similarly, -=, *=, /=
a += a;                  // a = 6,10,14
                         // So a can appear on both sides.
```

All these operate element-wise. Note, that there are no return versions of these operators (i.e., expressions like `b = a+1` are not possible). This is again for efficiency reasons. It is currently an active area of research in programming techniques how to make this kind of expression efficient. None of the available solutions works, so ARTS has to live without it.

Maximum and minimum:

```
cout << max(a);
cout << min(a);
```

Scalar product:

```
cout << a*a;
```

This is an exception to the rule not to have return versions of operators. The reason is quite obvious: The return value is only a scalar.

Arbitrary single-argument math functions:

```
Vector b(a.nelem());
transform(b,sin,a);  // b = sin(a)
transform(b,cos,b);  // b = sin(b)
                     // So b can appear on both sides.
```

The transform function operates on each element of `a` with the function you specify and puts the result in `b`. Note that the order of the arguments is swapped compared to the old function `trans` that we had in the pre-Matpack era.

12.3 Matrices

The class `Matrix` implements the mathematical concept of a matrix. (Who would have guessed this?) This means that:

- A Matrix contains floating point values of type `Numeric`.
- The values are arranged in rows and columns and can be accessed by indices. The first index is the row, the second the column. In other words, we use *row-major* order, similar to C, Matlab, and most math textbooks. Note, however, that some languages like FORTRAN and IDL use *column-major* order.
- A Matrix can be multiplied with a Vector, or with another Matrix.
- A sub-range of a Matrix in both dimensions (submatrix) can easily be accessed, and used as if it was just a normal matrix.
- Resizing a Matrix is expensive and should be avoided.

12.3.1 Constructing a Matrix

You can construct an object of class Matrix in any of these ways:

```
Matrix a;           // Create empty Matrix.
Matrix b(3,4);      // Create Matrix with 3 rows
                    // and 4 columns. When
                    // created like this it will contain
                    // arbitrary values.
Matrix c(3,4,0.0); // Similar, but
                    // fill it with 0.

Matrix d=c;         // Make d a copy of c.
```

That is all. More fancy constructors, like for Vector, do not exist for Matrix. There is also no equivalent to the `MakeVector` class.

12.3.2 MatrixViews

A `MatrixView` is a view on an existing Matrix, in the same way as a `VectorView` is a view on an existing Vector. Like a VectorView, a MatrixView cannot be resized and does not contain the actual data. A view is generated by using Ranges:

```
Matrix x(10,20);                  // Create 10x20 matrix.
MatrixView a = x;                 // Now a refers to the
                                  // whole of x;
MatrixView b = x(Range(joker),Range(joker));
                                  // Same effect.
MatrixView c = x[Range(0,2),Range(0,2)];
                                  // 2x2 sub-matrix.
```

I think you get the idea. Note that the second argument of Range gives the number of elements to take, not the index of the last element. See the section about Vectors for more examples how to use Range. You can use `joker`, and also the third argument of Range to select only every nth row, or column, or reverse the order of the rows or columns.

In analogy to the Vector case, you should use the two classes `MatrixView` and `ConstMatrixView` as function arguments. Please refer to the discussion in the Vector section for details. As in the case of VectorViews, all arguments of these types should be passed by value, not by reference. Also, similar to the Vector case, workspace methods are the exception, because they have to use the conventional `const Matrix&` or `Matrix&` as input/output arguments.

12.3.3 What you can do with a Matrix (or MatrixView)

All examples below (except for the first) assume that `a` is a Matrix or MatrixView.

Resize (only for Matrix, not for MatrixView!):

```
a.resize(5,10);
```

This makes `a` a 5x10 Matrix (5 rows, 10 columns). The new Matrix is not initialized (i.e., the contents will be unpredictable). Also, note that the previous content will be completely lost.

Get the number of rows or columns:

```
cout << a.nrows();
cout << a.ncols();
```

Refer to a row or column:

```
Vector x = a(0,Range(joker));           // First row.
Vector y = a(Range(joker),a.ncols()-1); // Last column.
```

Of course, you can use more complicated Range expressions to refer to only parts of a row or column. Technically, expressions of this kind return the type `VectorView`. This means, they can be used in all cases where an object of that type is expected, for example with the function defined in Section 12.2.2:

```
silly_function(a(0,Range(joker)),
               a(1,Range(joker))); // Call silly_function
                                   // with first and
                                   // second row of a.
```

Element access:

```
cout << a(3,4); // Print that element.
a(0,0) = 3.5;   // Assign 3.5 to the top-left element
```

Note that we use 0-based indexing! Furthermore note that the operator '()' can be also used with one or two `Range` arguments, as explained above. To summarize:

- (`Index`,`Index`) returns `Numeric` (element access).
- (`Index`,`Range`) or (`Range`,`Index`) returns `VectorView` (row or column access).
- (`Range`,`Range`) returns `MatrixView` (sub-matrix access).

You may find it unlogical, that Matrix uses '()' for indexing, whereas Vector uses '[]'. However, using '[]' for Matrix is not possible, since it can have only one argument. On the other hand, using '()' for Vector element access seemed not a good idea, since that would break with the established use of '[]' for element access in C and C++.

Copying Matrices:

```
Matrix b;
b.resize(a.nrows(),a.ncols());
b = a;
```

As in the case of Vectors, the '=' operator copies only the *contents*, so the dimensions must match. An attempt to justify this behavior has been made above in the Section about Vector. As for Vector, you can use '=' with complicated expressions. Here is a more elaborate example:

```
b(Range(0,3),Range(0,4)) =
   a(Range(10,3),Range(3,4,-1)); // Copy a row 10-12,
                                 // column 0-3
                                 // to b row 0-2,
                                 // column 0-3, reversing
                                 // the order of columns.
```

If you do not understand the use of Range here, refer to Section 12.2.2. Also, please keep in mind what has been said there about the difference between using '=' for copying, and using it for *constructing* something. In the first case the dimensions of the left operand must match the right operand, in the second case the left operand is created to match the right operand.

Assigning a scalar:

```
a = 1.0;                        // Assign 1 to all elements.
```

Mathematical operators:

You can use the operators '+=', '-=', '*/', and '/=', which operate element-vise, just as for Vector.

Maximum and minimum:

```
cout << max(a);
cout << min(a);
```

Arbitrary single-argument math functions:

The function `transform` works just like for Vector.

Transpose:

```
Matrix b = transpose(a); // Make b the transpose of a.
```

The function `transpose` creates a MatrixView, for which rows and columns are interchanged. Note, that only the way the data is accessed is changed, not the data itself. So Matrix `a` in the example above is not changed. For this reason, transposing is very efficient. You can use `transpose(a)` instead of `a` in any matrix expression practically without additional cost. (This is not strictly true, after all, the view has to be generated and passed. But that cost should be negligible except for very small matrices.)

Matrix multiplication:

```
// Matrix-Vector:
Vector b(a.nrows()), c(a.ncols());
mult(b,a,c);                            // b = a * c

// Matrix-Matrix:
Matrix d(a.nrows(),5), e(a.ncols(),5);
mult(d,a,e);                            // d = a * e
```

Note, that the result is put in the first argument, consistent with the general ARTS policy, but different from the old MTL based multiplication function. Furthermore note, that as you can see from the first example, a Vector is always considered to be a 1-column Matrix. You can use transpose, of course:

```
// Define b and c as in first example above.
mult(c,transpose(a),b);                      // c = a' * b

// Vector-Matrix:
mult(transpose(c),transpose(b),a);           // c' = b' * a
```

These two last examples should obviously give the same result.

12.4 Arrays

The template class `Array` can be used to make arrays out of anything. I do not know a good definition for 'array', but I guess anybody who has written a computer program in any programming language is familiar with the concept. Of course, it is rather similar to the concept of a Vector, just missing all the mathematical functionality like Matrix-Vector multiplication and sub-range access.

The implementation of our `Array` class is based on the STL class `std::vector`, whereas the implementation of our `Vector` class is done from scratch. So the two implementations are completely independent. Nevertheless, I tried to make `Array` behave consistently with `Vector`, as much as possible. There are a number of important differences, though, hopefully sufficiently explained in this part. A short summary of important differences:

- An Array can contain elements of any type, whereas a Vector always contains elements of type Numeric.
- No mathematical functionality for Array (no sub-ranges (nothing like VectorView); no +=, -=, *=, /=; no scalar product; no `transform` function; no `mult` function; no `transpose` function).
- On the other hand, resizing (for example adding to the end) of an Array is ok. (See the `push_back` method below.) It is still rather expensive, though, at least for large Arrays.

12.4.1 Constructing an Array

You can construct an object of an Array class like this:

```
Array<Index>  a;          // Empty Array of class Index.

Array<String> b(5);       // String Array with 5
                          // elements. Without initialization,
                          // elements contain random values.
Array<String> c(5,"x");  // The same, but fill with "x".

Array<Index>  d=a;        // Make d a copy of a;
```

There are already a lot of predefined Array classes. The naming convention for them is: `ArrayOfIndex`, `ArrayOfString`, etc.. Normally you should use these predefined classes. But if you want to define an Array of some uncommon type, you can do it with ‘<>’, as in the above examples.

As for Vector, there is a special sub-class of Array that can be initialized explicitly. Usage:

```
MakeArray<String> a("ARTS",
                    "is",
                    "great"); // Creates an array of String
                              // with these 3 elements.
```

12.4.2 What you can do with an Array

All examples below assume that `a` is an ArrayOfString.

Resize:

```
a.resize(5);
```

This adjusts the size of `a` to 5. Resizing is more efficiently implemented than for Vector, but still expensive.

Get the number of elements:

```
cout << a.nelem();  // Just as for Vector.
```

In particular, note that the return type of this method is `Index`, just as for Vector. This is an extension compared to std::vector, which just has a method `size()` that returns the positive integer type `size_t`.

Element access:

```
cout << a[3];    // Print 4th element.
a[0] = "Hello"; // Assign string "Hello" to first element.
```

In other words, this works just like for Vector.

Copying Arrays:

This works also the same as for Vector. The size of the target must match! In this respect, I have modified the behavior with respect to the underlying std::vector, which has different copy semantics.

Assigning a scalar of the base type:

```
a = "Hello";     // Assign string "Hello" to all elements.
```

Append to the end:

```
a.push_back("Hello"); // Adds this new element at the
                      // end of a.
```

This can be an expensive operation, especially for large Arrays. Therefore, use it with care. Actually, the `push_back` method comes from the `std::vector` class that Array is based on. You can do a lot more with `std::vector`, all of which also works with `Array`. However, to explain the Standard Template Library is beyond the scope of this text. You can read about it in C++ or even dedicated STL textbooks.

Chapter 13

Workspace variable groups and file formats

This section defines the data types, basic mathematical operations and file formats supported by ARTS. The implementation of vectors, matrices, and sparse matrices is based on the handmade MATPACK package, which is part of the ARTS source code (files matpackI.h, matpackII.h, matpackI.cc, and matpackII.cc). The implementation of arrays is based on the Standard Template Library (STL).

You can read ARTS variables from Ascii or Binary files, and also write them to Ascii or Binary files. The Ascii file format is very simple and explained in the online help (`arts -d ArrayOfMatrixWriteAscii`). Binary files are created and read by using the Hierarchical Data Format (HDF). Some information and help how to install HDF is given below.

ARTS workspace variables are organized in *groups*. Such a group is similar to a type in C or C++. For example, `String` and `Vector` are both groups. To get a complete list of workspace variable groups, call ARTS like this: `arts -g`.

13.1 Important workspace variable groups

13.1.1 Atomic groups

The most basic, the atomic, groups of ARTS are:

- `Index`
- `Numeric`
- `String`

A variable of group `Index` is a positive or negative integer. `Index` is the general purpose integer type of ARTS. Internally this type is set to the C data type `long int`. `Index`

History	
001027	Started by Patrick Eriksson.
010904	Started new text about handmade matrix/vector package. Stefan Bühler.

is used for indexing vectors, matrices, and arrays. The type is also used for all function flags, i.e., to make a selction among a limited number of choices. Accordingly, characters or strings shall not be used as flags.

A variable of group `Numeric` is a floating point number. Internally, it is either set to be double or float by `configure`. If `Numeric` is set to be double, the calculations will be more accurate and there is a smaller risk to encounter numerical problems in, e.g., matrix inversions. On the other hand, when `Numeric` is set to be float the calculations will be more rapid (about twice as fast), and the program will need only half as much memory. The type selected for `Numeric` is also reflected in the size of output files.

A variable of group `String` is not a true 'atomic' variable, as it consists of a number of characters, but as characters are not used in ARTS, strings are the most basic text type in ARTS.

13.1.2 Numeric groups

Numeric values can be stored in vectors or matrices for which mathematical operations like computing a matrix/vector product are possible. ARTS uses the following types:

- `Vector`
- `Matrix`

Both of these use 0-based indexing, i.e., `a[0]` is the first element of the `Vector a`. Because these types are quite powerfull, they are described in more detail in a separate section, which you should read if you want to do ARTS development (to be written).

13.1.3 Arrays based on atomic and numeric groups

Arrays correspond to vectors but are not treated as mathematical objects, they are only used as containers to hold different data. The arrays (as vectors and matrices) have 0-based indexing, that is, the first element has index 0 (not 1). Some examples for `Arrays` are:

- ArrayOfIndex
- ArrayOfString
- ArrayOfVector
- ArrayOfMatrix

13.1.4 Structures based on atomic and numeric groups

The `Los` is a structure to describe the line of sight (LOS). The structure holds for example the pressures along the LOS. To make the calculations more efficient for 1D calulations, only one half of the LOS is stored. For this reason, the LOS structure includes indecies to describe the iteration order. The structure also contains the index for ground reflections and the geometrical step length along the LOS. The structure is defined in `los.h`. The LOS calculations are further described in Section 6.

Numeric	Vector	Matrix
ArrayOfVector	ArrayOfMatrix	
Index	ArrayOfIndex	
String	ArrayOfString	

Table 13.1: ARTS data types that can be stored using the ASCII file formats (.aa).

13.2 File formats

All ARTS data, beside the spectroscopic variables, can be stored to, or loaded from, binary files using HDF. For some data types an ASCII file format also exists (Table 13.1). The default extension for ASCCI files is `.aa` (ARTS ASCII) and for binary files it is `.ab` (ARTS binary).

13.2.1 ASCII

All data types based on `Numeric` and `Index` that can be represented by an `ArrayOfMatrix` are stored using a common ASCII file format. Table 13.1 contains the data types that fulfills this criteria. Numeric ASCII files have the following structure:

```
# The file can start with an arbitrary number of comment lines.
# These lines starts with the hash symbol (#)
# The first row after the comment lines give the number of matrices
# in the array. After this follows, for each matrix, a row giving
# the matrix size followed by the data in row order.
2
2 3
1.1 2.2 3.3
4.4 5.5 6.6
1 1
3.1415
```

Index arrays (`ArrayOfIndex`) are stored as integer vectors to make the files easier to inspect.

The sizes given in the file must be compatible with the data type of the variable that is read. Vectors can be given both as columns or row matrices.

The types `STRING` and `ArrayOfSTRING` are stored using a similar file format. String ASCII files have the following structure:

```
# The file can start with an arbitrary number of comment lines.
# These lines starts with the hash symbol (#)
# The first row after the comment lines give the number of strings
# in the array, followed by the strings (one on each row).
3
String 1
String 2
String 3
```

13.2.2 Binary

Binary files are created and read by using HDF 4 (Sec. 13.3). Hence, HDF must be installed to use binary files. The Vdata format is applied. Most data types are stored using a common approach but for some data types a special format is used. The existing solution is temporary and the file format file will be changed.

General binary file format

The binary files for data types that can be treated as special cases of a matrix or can be broken down to a number of matrices have a common layout. In this context strings are treated as vectors of characters and scalars as 1x1 matrices. For example, the data types in Table 13.1 meat this criteria.

A Vdata contains only a single scalar, vector or matrix. The file for an array contains thus a number of Vdatas. The fields of a Vdata contains a single number or character. The field order for the common format is thus throughout 1. Matrices are in row order, that is, the data order is (1,1), (1,2), (1,3),. . . ,(2,1),(2,2),. . . .

The matrix dimensions are stored as an attribute to each Vdata. The name of the attribute is `SIZE`. The order of `SIZE` is 2 where the first value is the number of rows and the second value is the number of columns. The data type of `SIZE` is unsigned 4 byte integers (= HDF type `DFNT_UINT32`). Vectors and strings are treated as column objects (i.e. 1 column).

The data type of the file data is indicated by the class name of the Vdata. `Index` data are stored as 4 byte unsigned integers and the class name is set to `UINT` (= HDF type `DFNT_UINT32`). The class name for characters of strings is `CHAR` and the file data type is 1 byte characters (= HDF type `DFNT_CHAR`).

The record size for floating point values can either be 4 or 8 bytes. The corresponding class names are `FLOAT` and `DOUBLE` (= HDF type `DFNT_FLOAT32` and `DFNT_FLOAT64`, respectively). The type of `Numeric` determines the file type when writing from ARTS. Data is automatically converted to the type of `Numeric` when reading file data.

The field name of a Vdata (remember that a Vdata has here only a singel field) describe the structure of the data. Treated structure types are `SCALAR`, `VECTOR`, `Matrix` and `STRING`. The following combinations of data structure and data type are allowed:

```
SCALAR: UINT, FLOAT, DOUBLE
VECTOR: UINT, FLOAT, DOUBLE
MATRIX: FLOAT, DOUBLE
STRING: CHAR
```

The Vdata name is used to handle arrays. However, for simplicity reasons and for consistency with the ASCII format, index arrays (`ArrayOfIndex`) are stored as index vectors (field name `VECTOR` and Vdata class `UINT`). For other type of arrays, the Vdata name is set to the data structure name followed by a sequential number (starting at 1). The length of the array is given by a seperate Vdata holding an index number (`SCALAR`, `UINT`), named as `N_`*string_type*. An example should clarify this. A file holding a matrix array of length 3 has the the Vdatas `N_MATRIX`, `MATRIX1`, `MATRIX2` and `MATRIX3`.

The described approach to store binary data results in that each ARTS data type has a corresponding format for binary files. This gives an automatic check that a file matches the data type of an ARTS variable when reading from a binary file. The drawback is that, for example, a file holding a matrix cannot be read to create an matrix array of length 1, which is possible for the ASCII format, but consistency was emphasized when designing the binary format. In addition, the size attribute is used to check that the data have the expected size, for example, vectors are expected to only have one column.

Display tools

The content of the binary files can be displayed using some command line HDF utilities. A first utility is `vshow`. The syntax is

`vshow` *filename* `+`

where *filename* is the binary file of interest. The final + indicates that the values of the stored data shall be displayed. Without the + symbol, only the data structure is reported.

Another utility is `hdp`. Type `hdp -h` for some on-line help. To display the values of the stored data, type

`hdp dumpvd` *filename*

13.3 HDF

The HDF home page is found at

`http://hdf.ncsa.uiuc.edu/`

The HDF 4 data format is used. The present version of ARTS has been tested with HDF 4.1r3. The precompiled binaries were used.

HDF is not supplied with ARTS, it must be installed seperately as a library. For example, to install HDF on a Linux system, try the following:

1 Download the precompiled version for your system. Unpack.

2 Copy the contents of /bin, /include, /man and /lib to the corresponding sub-directories of /usr/local. You need to be superuser to do this.

Part III

Utilities

Chapter 14

Utilities

This section will describe the utility programs and functions that are distributed along with ARTS. So far only the IDl interface is described. However, the described read and write functions also exist for Matlab. Type `help` *function_name* (in Matlab) to get some information about the functions.

14.1 The ARTS-IDL interface: AII

14.1.1 Introduction

The following sections show the usage of the IDL reading and writing routines.

14.1.2 IDL reading routines

read_datafile

This function reads data from a file in ARTS ASCII data format.

Calling Sequence	*x* = `read_datafile` (*filename*)	
Argument	*filename*	full file name
Keyword	check	flag to check the data
Output	*x*	the data

Depending on the number of stored matrices the data are returned as an array or a structure of arrays.

If there is only one matrix in the file, just type `print, x` at the IDL prompt to get it. If there are several matrices in the file, type `print, x.mat`*n* to get the matrix with the number $n + 1$ $(n = 0, 1, 2, \ldots)$ or make an assignment like `mat = x.mat`*n*.

In order to make a check of consistency of the stored data, i. e. the correctness of the indicated dimension of each matrix in reference to the number of actually available matrix

---

History

001101 Started by Stefan Buehler.

000228 Wolfram Haas

values, you can set the keyword 'check'. In this case the reading process will be considerably slower.

Example Reading the file `test.dat` (see 14.1.3)

```
IDL> x = read_datafile('test.dat')
```

The variable x is a structure of matrices. The first matrix is `x.mat0`. In order to get the first three matrices, enter

```
IDL> print, x.mat0
       1.2340000       2.3450000       3.4560000
       4.5670000       5.6780000       6.7890000
IDL> print, x.mat1
       3.1415930
IDL> print, x.mat2
      -1.3467120       2.4578230
       3.5689340      -4.6790450
```

Error messages

- `Blank lines are not allowed.`
    1. The message appears if there are blank lines at the beginning of the file or blank lines between the number of matrices and the size of the following matrix or between a matrix and the size of the following matrix in case of several matrices.
    2. The number of matrices is greater than the actually available number of matrices and there are blanks or blank lines at the end of the file.
- `Missing number of matrices.`
- `Could not read number of matrices.`
  The number of matrices is less than one.
- `Wrong number of matrices.`
  The number of matrices is greater than the number of matrices in the file.
- `Could not read matrix size.`
  The line in which the size of the matrix should be contains more than two numbers. Possibly the size of the matrix is completely missing.
- `There is some garbage at the end of the file.`
    1. There are additional numbers and symbols at the end of the file.
    2. The number of matrices is less than the number of matrices in the file.
    3. The size of the matrix can be wrong if there is only one matrix in the file and if you use the keyword 'optimize'.

If the 'check'-keyword is set, the following error messages can occur:

- `One or more rows are missing.`

- `Blank lines are not allowed within a matrix.`
  1. The message appears if there are blank lines between the size of a matrix and the following rows. Possibly matrix values are missing.
  2. If it is only one matrix stored in the file one or more rows can miss and instead of these rows one or more blank lines are there.
- `Wrong number of column elements.`
  The number of columns indicated in the file is larger or smaller than the actually available number of columns.

read_artsvar

This function reads an ARTS variable.

Calling Sequence	*x* = `read_artsvar` (*basename*, *varname*)	
Arguments	*basename*	the ARTS basename
	varname	variable name
Keyword	check	flag to check the data
Output	*x*	the data

The data is read from the file '*basename.varname*.am'. For details see 'read_datafile'.

14.1.3 IDL writing routines

write_datafile

This procedure writes data to a file in ARTS format.

Calling Sequence	`write_datafile`, *filename*, *x*, *heading [*, *prec]*	
Arguments	*filename*	full file name
	x	the data to store
	heading	heading text
Optional	*prec*	number of decimals to use, default 6

The data can be transferred to the procedure in form of an array or a structure of arrays. See also 'read_datafile'.

You can create a structure in this way:

$$VariableName = \{Tag\_Name_1 : Tag\_Definition_1, \ldots, Tag\_Name_n : Tag\_Definition_n\}$$

If *prec* is equal to zero, integer values are assumed.

Example Creating a file of matrices, vectors and scalars.

```
IDL> m = {a: dblarr(3, 2), b: dblarr(1, 1), c: dblarr(2, 2),
d: 0., e: dblarr(3, 1), f: dblarr(1, 3)}
IDL> m.a = [[1.234, 2.345, 3.456], [4.567, 5.678, 6.789]]
IDL> m.b = [3.1415926536]
IDL> m.c = [[-1.346712, 2.457823], [3.568934, -4.679045]]
IDL> m.d = 2.718281828
IDL> m.e = [1, 5, 8]
IDL> m.f = [[1.2], [2.3], [3.4]]
```

In order to write the structure m to the file 'test.dat', type at the IDL prompt

```
IDL> write_datafile, 'test.dat', m, 'File test.dat'
```

An empty heading text is made by typing " instead of 'File test.dat'.
The result is:

```
# File test.dat
#
# This file is created by IDL.
6
2  3
1.234000e+00 2.345000e+00 3.456000e+00
4.567000e+00 5.678000e+00 6.789000e+00
1  1
3.141593e+00
2  2
-1.346712e+00 2.457823e+00
3.568934e+00 -4.679045e+00
1  1
2.718282e+00
1  3
1.000000e+00 5.000000e+00 8.000000e+00
3  1
1.200000e+00
2.300000e+00
3.400000e+00
```

write_artsvar

This procedure writes an ARTS variable to a file in ARTS format.

Calling Sequence	`write_artsvar`, *basename*, *varname*, *x [*, *prec]*	
Arguments	*basename*	the ARTS basename
	varname	variable name
	x	the data to store
Optional	*prec*	number of digits to use, default 6

The data is written to a file called '*basename.varname.*am'. See also 'write_datafile'.

Part IV

Bibliography and Appendices

Bibliography

Balluch, M., and D. Lary, Refraction and atmospheric photochemistry, *J. of Geophys. Res.*, *102*, 8845–8854, 1997.

Bauer, A., M. Godon, M. Kheddar, and J. M. Hartmann, Temperature and perturber dependence of water vapor line broadening: Experiments at 183 GHz; calculations below 1000 GHz, *Journal of Quantitative Spectroscopy and Radiative Transfer*, *41*, 49–54, 1989.

Bauer, A., M. Godon, J. Carlier, Q. Ma, and R. H. Tipping, Absorption by H_2O and H_2O-N_2 mixtures at 153 GHz, *Journal of Quantitative Spectroscopy and Radiative Transfer*, *50*, 463–475, 1993.

Bauer, A., M. Godon, J. Carlier, and Q. Ma, Water vapor absorption in the atmospheric window at 239 GHz, *Journal of Quantitative Spectroscopy and Radiative Transfer*, *53*, 411–423, 1995.

Becker, G. E., and S. H. Autler, Water vapor absorption of electromagnetic radiation in the centimeter wavelength range, *Phys. Rev.*, *70*, 300–307, 1946.

Bernath, P., *Spectra of Atoms and Molecules*, Oxford University Press, 1995, ISBN 0-195-07598-6.

Berton, R. P. H., Statistical distribution of water content and sizes for clouds above Europe, *Ann. Geophys.*, *18*, 385–397, 2000.

Borysow, A., and L. Frommhold, Collision induced rototranslational absorption spectra of N_2–N_2 pairs for temperatures from 50 to 300 K, *Astrophysical Journal*, *311*, 1043–1057, 1986.

Brussaard, G., and P. A. Watson, *Atmospheric Modelling and Millimetre Wave Propagation*, Chapman & Hall, 1995, ISBN 0-412-56230-8.

Chase, M. W., C. A. Davies, J. R. Downey, D. J. Frurip, and R. A. McDonald, Janaf thermochemical tables, third edition, *JPCRD*, *14*, 1274, 1985.

Clough, S. A., F. X. Kneizys, and R. W. Davis, Line shape and water vapor continuum, *Atmospheric Research*, *23*, 229–241, 1989.

Costa, A. A., G. P. Almeida, and A. J. C. Sampaio, A bin-microphysics cloud model with high-order, positive-definite advection, *Atmospheric Research*, *55*, 225–255, 2000.

Cruz Pol, S. L., C. S. Ruf, and S. J. Keihm, Improved 20– to 32–GHz atmospheric absorption model, *Radio Science*, *33*, 1319–1333, 1998, updated version can be downloaded from `http://ece.uprm.edu/~pol/Atmosphere.html`.

Dagg, I. R., G. E. Reesor, and J. L. Urbaniak, Collision induced absorption in N_2, CO_2, and H_2 at $2\,\mathrm{cm}^{-1}$, *Canadian Journal of Physics*, *53*, 1764–1776, 1975.

Dagg, I. R., G. E. Reesor, and M. Wong, A microwave cavity measurement of collision-induced absorption in N_2 and CO_2 at $4.6\,\mathrm{cm}^{-1}$, *Can. J. Phys.*, *56*, 1037–1045, 1978.

Drayson, S. R., Rapid computation of the Voigt profile, *Journal of Quantitative Spectroscopy and Radiative Transfer*, *16*, 611, 1976.

English, S. J., C. Guillou, C. Prigent, and D. C. Jones, Aircraft measurements of water vapor continuum absorption at millimeter wavelengths, *Quarterly Journal of the Royal Meteorological Society*, *120*, 603–625, 1994.

Eriksson, P., Microwave radiometric observations of the middle atmosphere: Simulations and inversions, Ph.D. thesis, School of Electrical and Computer Engineering, Chalmers University of Technology, Sweden, 1999.

Eriksson, P., and F. Merino, On simulating passive observations of the middle atmosphere in the range 1 - 1000 GHz, *Tech. Rep. 179*, Department of Radio and Space Science, Chalmers University of Technology, Sweden, 1997.

Eriksson, P., F. Merino, D. Murtagh, P. Baron, P. Ricaud, and J. de la Nöe, Studies for the Odin sub-millimetre radiometer: 1. Radiative transfer and instrument simulation, *to appear in Canadian Journal of Physics*, 2000.

Eriksson, P., C. Jiménez, S. Bühler, and D. Murtagh, A Hotelling transformation approach for rapid inversion of atmospheric spectra, *J. Quant. Spectrosc. Radiat. Transfer, in press*, 2001a.

Eriksson, P., C. Jiménez, D. Murtagh, G. Elgered, T. Kuhn, and S. Bühler, Assessment of uncertainties in LEO-LEO transmission observations through the troposphere/stratosphere, *Tech. rep.*, ESTEC / Contract No ???, 2001b.

Godon, M., J. Carlier, and A. Bauer, Laboratory studies of water vapor absorption in the atmospheric window at 213 GHz, *Journal of Quantitative Spectroscopy and Radiative Transfer*, *47*, 275–285, 1992.

Goody, R. M., *Principles of atmospheric physics and chemistry*, Oxford University Press, 1995.

Goody, R. M., and Y. L. Yung, *Atmospheric radiation, theoretical basis*, Oxford University Press, 1989, second edition.

Gordy, W., and R. Cook, *Microwave Molecular Spectra*, Interscience Publishers, 1970, SBN 471 93161 6.

Herzberg, G., *Infrared and Raman Spectra of Polyatomic Mollecules*, Van Nostrand Reinhold Company, 1945, ISBN 0-442-03386-9.

Hess, M., P. Koepke, and I. Schult, Optical properties of aerosols and clouds: The software package OPAC, *Bulletin of the American Meteorological Society*, *79*, 831–844, 1998.

Ho, W., I. A. Kaufman, and P. Thaddeus, Laboratory measurement of microwave absorption in models of the atmosphere of Venus, *J. of Geophys. Res.*, *71*, 5091–5108, 1966.

Hufford, G. A., A model for the complex permittivity of ice at frequencies below 1 thz, *Interntl. J. Infrared & Millimeter Waves*, *12*, 677–682, 1991.

Hui, A. K., B. H.Armstrong, and A. A. Wray, Rapid computation of the voigt and complex error functions, *Journal of Quantitative Spectroscopy and Radiative Transfer*, *19*, 509–516, 1978.

Kneizys, F. X., et al., The MODTRAN2/3 report and LOWTRAN7 model laboratory studies and propagation modelling, *Tech. Rep. 1/11/96*, Phillips Laboratory, Geophysical Directorate, PL/GPOS, 1996, editors: L. W. Abreu and G. P. Anderson.

Kursinski, E. R., G. A. Hajj, J. T. Schofield, R. P. Linfield, and K. R. Hardy, Observing Earth's atmosphere with radio occultation measurements using the Global Positioning System, *J. of Geophys. Res.*, *102*, 23429–23465, 1997.

Kyle, T., *Atmospheric transmission, emission and scaterring*, Pergamon Press, 1991.

Larsen, H., J.-F. Gayet, G. Febvre, H. Chepfer, and G. Brogniez, Measurement errors in cirrus cloud microphysical properties, *Ann. Geophys.*, *16*, 266–276, 1998.

Li, L., J. Vivekanandan, C. H. Chan, and L. Tsang, Microwave radiometric technique to retrieve vapor; liquid and ice; part I – developement of a neural network-based inversion method, *IEEE Transactions on Geoscience and Remote Sensing*, *35*, 224–235, 1997.

Liebe, H. J., Modelling attenuation and phase of radio waves in air at frequencies below 1000 ghz, *Radio Science*, *16*, 1183–1199, 1981.

Liebe, H. J., The atmospheric water vapor continuum below 300 GHz, *Interntl. J. Infrared & Millimeter Waves*, *5(2)*, 207–227, 1984.

Liebe, H. J., MPM – an atmospheric millimeter–wave propagation model, *Interntl. J. Infrared & Millimeter Waves*, *10*, 631–650, 1989.

Liebe, H. J., and D. H. Layton, Millimeter-wave properties of the atmosphere: Laboratory studies and propagation modelling, *Tech. Rep. 87224*, U.S. Dept. of Commerce, National Telecommunications and Information Administration, Institute for Communication Sciences, 1987, 80p.

Liebe, H. J., T. Manabe, and G. A. Hufford, Millimeter–wave attenuation and delay rates due to fog/cloud conditions, *IEEE Trans. Antennas Propag.*, *37*, 1617–1623, 1989.

Liebe, H. J., G. A. Hufford, and T. Manabe, A model for the complex permittivity of water at frequencies below 1 thz, *Interntl. J. Infrared & Millimeter Waves*, *12*, 659–675, 1991.

Liebe, H. J., P. W. Rosenkranz, and G. A. Hufford, Atmospheric 60-GHz oxygen spectrum: new laboratory measurements and line parameters, *Journal of Quantitative Spectroscopy and Radiative Transfer*, *48*, 629–643, 1992.

Liebe, H. J., G. A. Hufford, and M. G. Cotton, Propagation modeling of moist air and suspended water/ice particles at frequencies below 1000 GHz., in *AGARD 52nd Specialists Meeting of the Electromagnetic Wave Propagation Panel, Palma de Mallorca, Spain*, 1993, `ftp://ftp.its.bldrdoc.gov/pub/mpm93/`.

Lipton, A. E., M. K. Griffin, and A. G. Ling, Microwave transfer model differences in remote sensing of cloud liquid water at low temperatures, *IEEE Transactions on Geoscience and Remote Sensing*, *37*, 620–623, 1999.

Ludlam, F. H., and B. J. Mason, The physics of clouds, in *Encyclopedia of Physics*, edited by S. Flğge, vol. 48, Springer, Berlin, 1957.

Ma, Q., and R. H. Tipping, Water vapor continuum in the millimeter spectral region, *Journal of Chemical Physics*, *93*, 6127–6139, 1990.

Oliveiro, J. J., and R. L. Longbothum, Empirical fits to voight line width: A brief review, *Journal of Quantitative Spectroscopy and Radiative Transfer*, *17*, 233–236, 1977.

Pawlowska, H., J. L. Brenguier, and F. Burnet, Microphysical properties of stratocumulus clouds, *Atmospheric Research*, *55*, 15–33, 2000.

Press, W., S. Teukolsky, W. Vetterling, and B. Flannery, *Numerical recipes in FORTRAN*, 2nd ed., Cambridge University Press, 1992.

Ray, P., Broadband complex refractive indices of ice and water, *Appl. Opt.*, *11*, 1836–1844, 1972.

Reburn, W. J., R. Siddans, B. J. Kerridge, S. Bühler, A. von Engeln, P. Erikson, T. Kuhn-Sander, C. Verdes, and K. Künzi, Critical assessments in millimetre-wave atmospheric limb sounding, final report, *Tech. rep.*, ESTEC / Contract No 13348 / 98 / NL / GD, 2000.

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, *J. of Geophys. Res.*, *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`.

Rosenkranz, P. W., Water vapor microwave continuum absorption: A comparison of measurements and models, *Radio Science*, *33*, 919–928, 1998, (correction in 34, 1025, 1999), `ftp://mesa.mit.edu/phil/lbl_rt`.

Rothman, L. S., et al., The HITRAN molecular database: editions of 1991 and 1992, *Journal of Quantitative Spectroscopy and Radiative Transfer*, *48*, 469–507, 1992.

Rothman, L. S., et al., The HITRAN molecular spectroscopic database and HAWKS (HITRAN atmospheric workstation): 1996 edition, *Journal of Quantitative Spectroscopy and Radiative Transfer*, *60*, 665–710, 1998.

Salby, M. L., *Fundamentals of Atmospheric Physics*, vol. 61 of *International Geophysics Series*, Academic Press, 1996, ISBN 0-12-615160-1.

Schimpf, B., and F. Schreier, Robust and efficient inversion of vertical sounding atmospheric high-resolution spectra by means of regularization, *J. of Geophys. Res.*, *102*, 16037–16055, 1997.

Seinfeld, J. H., and S. N. Pandis, *Atmospheric Chemistry and Physics : from Air Pollution to Climate Change*, Wiley, 1998, ISBN 0-471-17816-0.

Shupe, M. D., T. Uttal, S. Y. Matrosov, and A. S. Frisch, Cloud water content and hydrometeor sizes during the FIRE-Arctic Clouds Experiment, *J. of Geophys. Res.*, *106*, 15015–15028, 2000.

Stankevich, K. S., Absorption of sub–millimeter–range radio waves in a dry atmosphere, *Radiophys. Quantum Electron. (Engl. Transl.)*, *17*, 579–581, 1974.

Stone, N. W., W. G. Read, A. Anderson, I. R. Dagg, and W. Smith, Temperature-dependent collision-induced absorption in nitrogen, *Can. J. Phys.*, *62*, 338–347, 1984.

Thorne, A., U. Litzen, and S. Johansson, *Spectrophysics : principles and applications*, Springer, 1999, 1999, ISBN 3-540-65117-9.

Townes, C. H., and A. L. Schawlow, *Microwave Spectroscopy*, Mc Graw-Hill, 1955.

Van Vleck, J. H., The relation between absorption and dispersion, in *Propagation of Short Radio Waves*, edited by D. E. Kerr, pp. 641–664, Peter Peregrinus Ltd, 1987, first published in 1951 by McGraw-Hill Book Comp. Inc.

Van Vleck, J. H., and D. L. Huber, Absoption, emission, and linebreadths: a semihistorical perspective, *Reviews of Modern Physics*, *49*, 939–959, 1977.

Van Vleck, J. H., and V. F. Weisskopf, On the shape of collision–broadening lines, *Reviews of Modern Physics*, *17*, 227—236, 1945.

Westwater, E. R., J. B. Snider, and M. J. Falls, Ground-based microwave radiometric retrieval of precipitable water vapor in the presence of clouds with high liquid content, *Radio Science*, *15*, 947–957, 1980.

Appendix A

Workspace variables

This appendix reports the existing ARTS workspace variables. The data type of the variables and the on-line text description are also given. For an on-line list of all workspace variables, type

```
arts -w all
```

To get a description of a specific workspace variable, type

arts -d *variable*

where *variable* is the name of the variable of interest. To list all methods that can be used to create or calculate a workspace variable, type

arts -m *variable*

To list all methods that need a given workspace variable as input, type

arts -i *variable*

For a complete list of online help options, type

```
arts -h
```

```
VARIABLE : lines
DATA TYPE: ArrayOfLineRecord
DESCRIPTION:
A list of spectral line data.

VARIABLE : lines_per_tg
DATA TYPE: ArrayOfArrayOfLineRecord
```

History

001110 Created by Patrick Eriksson.

```
DESCRIPTION:
A list of spectral line data for each tag.
Dimensions: (tag_groups.nelem()) (# of lines for this tag)

VARIABLE : tgs
DATA TYPE: TagGroups
DESCRIPTION:
This is an array of arrays of OneTag tag definitions.
It defines the available tag groups for the calculation
of absorption coefficients and weighting functions.
Contrary to the original Bredbeck definition, tags within a
group must belong to the same species, because one VMR profile
is associated with each tag group.

VARIABLE : wfs_tgs
DATA TYPE: TagGroups
DESCRIPTION:
This is an array of arrays of tag group definitions.
It defines the tag groups for the calculation of weighting
functions. The selected tag groups must be a subgroup of the
tag groups defined for the absorption coefficient calculation.

VARIABLE : wfss_tgs
DATA TYPE: TagGroups
DESCRIPTION:
This is an array of arrays of tag group definitions.
It defines the tag groups for the calculation of weighting
functions. The selected tag groups must be a subgroup of the
tag groups defined for the absorption coefficient calculation.

VARIABLE : lineshape
DATA TYPE: ArrayOfLineshapeSpec
DESCRIPTION:
Lineshape specification: function, norm, cutoff. There is one entry for
each abs_tag, not for each species. This means if you have several
abs_tags for different isotopes or transitions of a species, you
may use different lineshapes.

VARIABLE : cont_description_names
DATA TYPE: ArrayOfString
DESCRIPTION:
Continuum / full model absorption tag names. This variable should
contain a list of tag names of continuum and full models, respectively.
Associated with this WSV is the WSV
‘cont_description_models’ which contains the specific model version of
each continuum / full model absorption tag and the WSV
‘cont_description_parameters’ which should contain the continuum / full model
user defined parameters. The user defined parameters are only used when
the specified model is ’user’. See also the online documentation in
arts/doc/doxygen/html/continua_cc.html.

The following full water vapor models are implemented:
’H2O-MPM87’: absorption model (line and continuum) according to
   H. J. Liebe,
   A contribution to modeling atmospheric millimeter-wave properties,
   Frequenz,  41, 1987, 31-36
```

```
   and
   H. J. Liebe and D. H. Layton,
   Millimeter-wave properties of the atmosphere:
   Laboratory studies and propagation modeling,
   U.S. Dept. of Commerce, National Telecommunications and Information
   Administration, Institute for Communication Sciences,
   325 Broadway, Boulder, CO 80303-3328, report 87224.
'H2O-MPM89': absorption model (line and continuum) according to
   H. J. Liebe,
 Int. J. Infrared and Millimeter Waves, 10(6), 1989, 631
'H2O-MPM93': absorption model (line and continuum) according to
   H. J. Liebe and G. A. Hufford and M. G. Cotton,
   Propagation modeling of moist air and suspended water/ice
   particles at frequencies below 1000 GHz,
   AGARD 52nd Specialists Meeting of the Electromagnetic Wave
   Propagation Panel,
 Palma de Mallorca, Spain, 1993, May 17-21
   (ftp.its.bldrdoc.gov/pub/mpm93/)
'H2O-CP98': absorption model (line and continuum) according to
   S. L. Cruz-Pol et al.,
 Radio Science, 33(5), 1319, 1998   (ece.uprm.edu/~pol/Atmosphere.html)
'H2O-PWR98': absorption model (line and continuum) according to
   P. W. Rosenkranz,
    Radio Science, 33(4),  919, 1998, Radio Science, 34(4), 1025, 1999
   (ftp: mesa.mit.edu/phil/lbl_rt).

The following full oxygen models are implemented:
'O2-MPM93': absorption model (line and continuum) according to
   H. J. Liebe and G. A. Hufford and M. G. Cotton,
   Propagation modeling of moist air and suspended water/ice
   particles at frequencies below 1000 GHz,
   AGARD 52nd Specialists Meeting of the Electromagnetic Wave
   Propagation Panel,
 Palma de Mallorca, Spain, 1993, May 17-21
   (ftp.its.bldrdoc.gov/pub/mpm93/)
'O2-PWR93': absorption model (line and continuum) according to
   P. W. Rosenkranz,
 Chapter 2, in M. A. Janssen,
   Atmospheric Remote Sensing by Microwave Radiometry
   John Wiley & Sons, Inc., 1993 (mesa.mit.edu/phil/lbl_rt)

The following continuum parameterizations are implemented:
H2O-H2O ('H2O-SelfContStandardType'):
   P. W. Rosenkranz,
   Radio Science, Vol. 33, No 4, Pages 919-928, 1998 and
   Radio Science, Vol. 34, No 4, Page 1025, 1999 (mesa.mit.edu/phil/lbl_rt)
H2O-air ('H2O-ForeignContStandardType'):
   P. W. Rosenkranz,
   Radio Science, Vol. 33, No 4, Pages 919-928, 1998 and
   Radio Science, Vol. 34, No 4, Page 1025, 1999 (mesa.mit.edu/phil/lbl_rt)
H2O-air ('H2O-ContMPM93'):
   H. J. Liebe and G. A. Hufford and M. G. Cotton,
   Propagation modeling of moist air and suspended water/ice
   particles at frequencies below 1000 GHz,
   AGARD 52nd Specialists Meeting of the Electromagnetic Wave
   Propagation Panel,
```

```
 Palma de Mallorca, Spain, 1993, May 17-21
   (ftp.its.bldrdoc.gov/pub/mpm93/)
O2-air ('O2-SelfContStandardType'):
   P. W. Rosenkranz,
   Chapter 2, in M. A. Janssen,
   Atmospheric Remote Sensing by Microwave Radiometry,
   John Wiley & Sons, Inc., 1993
   (mesa.mit.edu/phil/lbl_rt)
   and also described in
   H. J. Liebe and G. A. Hufford and M. G. Cotton,
   Propagation modeling of moist air and suspended water/ice
   particles at frequencies below 1000 GHz,
   AGARD 52nd Specialists Meeting of the Electromagnetic Wave
   Propagation Panel,
 Palma de Mallorca, Spain, 1993, May 17-21
   (ftp.its.bldrdoc.gov/pub/mpm93/)
N2-N2 ('N2-SelfContStandardType'):
   The functional form of Rosenkranz but with more input parameters.
   P. W. Rosenkranz,
   Chapter 2, in M. A. Janssen,
   Atmospheric Remote Sensing by Microwave Radiometry,
   John Wiley & Sons, Inc., 1993 (mesa.mit.edu/phil/lbl_rt)
N2-N2 ('N2-SelfContMPM93'):
   H. J. Liebe and G. A. Hufford and M. G. Cotton,
   Propagation modeling of moist air and suspended water/ice
   particles at frequencies below 1000 GHz,
   AGARD 52nd Specialists Meeting of the Electromagnetic Wave
   Propagation Panel, Palma de Mallorca, Spain, 1993, May 17-21
   (ftp.its.bldrdoc.gov/pub/mpm93/)
CO2-CO2 ('CO2-SelfContPWR93'):
   P. W. Rosenkranz,
   Chapter 2, in M. A. Janssen,
   Atmospheric Remote Sensing by Microwave Radiometry,
   John Wiley & Sons, Inc., 1993 (mesa.mit.edu/phil/lbl_rt)
CO2-N2 ('CO2-ForeignContPWR93'):
   P. W. Rosenkranz,
   Chapter 2, in M. A. Janssen,
   Atmospheric Remote Sensing by Microwave Radiometry,
   John Wiley & Sons, Inc., 1993 (mesa.mit.edu/phil/lbl_rt)

The following cloud absorption models are implemented:
Suspended water droplet ('liquidcloud-MPM93')
   absorption parameterization from the MPM93 model:
   H. J. Liebe and G. A. Hufford and M. G. Cotton,
   Propagation modeling of moist air and suspended water/ice
   particles at frequencies below 1000 GHz,
   AGARD 52nd Specialists Meeting of the Electromagnetic Wave
   Propagation Panel,
 Palma de Mallorca, Spain, 1993, May 17-21
   (ftp.its.bldrdoc.gov/pub/mpm93/)
Ice water droplet absorption ('icecloud-MPM93')
   parameterization from MPM93 model:
   H. J. Liebe and G. A. Hufford and M. G. Cotton,
   Propagation modeling of moist air and suspended water/ice
   particles at frequencies below 1000 GHz,
   AGARD 52nd Specialists Meeting of the Electromagnetic Wave
```

```
   Propagation Panel,
  Palma de Mallorca, Spain, 1993, May 17-21
   (ftp.its.bldrdoc.gov/pub/mpm93/)

The following rain extinction model is implemented:
Rain extinction parameterization ('rain-MPM93') from the
   MPM93 model, described in:
   H. J. Liebe,
   MPM - An Atmospheric Millimeter-Wave Propagation Model,
   Int. J. Infrared and Millimeter Waves, vol. 10(6),
   pp. 631-650, 1989;
   and based on:
   Olsen, R.L., D.V. Rogers, and D. B. Hodge,
   The aR^b relation in the calculation of rain attenuation,
   IEEE Trans. Antennas Propagat., vol. AP-26, pp. 318-329, 1978.
   IMPORTANT NOTE: rain-MPM93 parameterizes the EXTINCTION by rain,
    not just the absorption. Therefore it is not suitable for
    calculating thermal emission by rain!
    Please use rain-MPM93 only for calculation of attenuation.



VARIABLE : cont_description_models
DATA TYPE: ArrayOfString
DESCRIPTION:
Continuum / full model absorption model description parameter.
See the WSV 'cont_description_names' for a detailed description
of the allowed continuum models. There should be one string here
for each entry in 'cont_description_names'.See also the onlinedocumentation in arts/doc/doxygen/


VARIABLE : cont_description_parameters
DATA TYPE: ArrayOfVector
DESCRIPTION:
Continuum model parameters. See the WSV 'cont_description_names'
for a detailed description of the allowed continuum models. There
should be one parameter vector here for each entry in
'cont_description_names'. See also the online documentation in
arts/doc/doxygen/html/continua_cc.html.


VARIABLE : raw_ptz
DATA TYPE: Matrix
DESCRIPTION:
Matrix has rows:
1. Pressure in Pa
2. Temperature in K
3. Altitude in m

VARIABLE : raw_vmrs
DATA TYPE: ArrayOfMatrix
DESCRIPTION:
The individual VMR profiles. Each species VMR profile comes with a
pressure profile. The different species can hence be on different
grids.
The matrix has rows:
```

```
1. Pressure in Pa
2. VMR profile (absolute number)
The array dimension is determined by the number of tag groups.

VARIABLE : p_abs
DATA TYPE: Vector
DESCRIPTION:
The pressure grid for the absorption coefficients [Pa]. This
is the basic independent grid for the absorption calculation, both
in the 1D and 2D case. Therefore it remains a vector, even in 2D.
The "raw" atmospheric data shall be interpolated to p_abs before
the absorption calculations starts.

VARIABLE : f_mono
DATA TYPE: Vector
DESCRIPTION:
The monochromatic frequency grid [Hz].

VARIABLE : t_abs
DATA TYPE: Vector
DESCRIPTION:
Temperature associated with the pressures in p_abs [K]

VARIABLE : z_abs
DATA TYPE: Vector
DESCRIPTION:
Vertical altitudes associated with the pressures in p_abs [m]

VARIABLE : h2o_abs
DATA TYPE: Vector
DESCRIPTION:
The total water profile associated with the pressures in p_abs [-]

VARIABLE : n2_abs
DATA TYPE: Vector
DESCRIPTION:
The total nitrogen profile associated with the pressures in p_abs [-]

VARIABLE : vmrs
DATA TYPE: Matrix
DESCRIPTION:
The VMRs (unit: absolute number) on the p_abs grid.
Dimensions: [tag_groups.nelem(), p_abs.nelem()]

VARIABLE : abs
DATA TYPE: Matrix
DESCRIPTION:
The matrix of absorption coefficients (in units of [1/m]).
Dimensions: [f_mono.nelem(), p_abs.nelem()]

VARIABLE : abs0
DATA TYPE: Matrix
DESCRIPTION:
A second absoprtion matrix. This matrix can be used, for example,
to store absorption read from a file that shall be added to *abs*.
```

```
VARIABLE : abs_per_tg
DATA TYPE: ArrayOfMatrix
DESCRIPTION:
These are the absorption coefficients individually for each
tag group. The Array contains one matrix for each tag group,
the matrix format is the same as that of abs

VARIABLE : xsec_per_tg
DATA TYPE: ArrayOfMatrix
DESCRIPTION:
These are the cross sections individually for each tag
group. The Array contains one matrix for each tag group,
the matrix format is the same as that of abs

VARIABLE : hse
DATA TYPE: Vector
DESCRIPTION:
This vector holds the parameters for calculating hydrostatic
equilibrium (HSE). The length of the vector is either 1 or 5, where
the values are:
   1: On/off flag. 0 = ignore HSE, 1 = consider HSE.
   2: The pressure of the reference point [Pa].
   3: The altitude of the reference point [m].
   4: Gravitational acceleration at the geoid surface [m/s2].
   5: Number of iterations of the calculations.
If the on/off flag is set to 1, the length of the vector must be 5,
while if the flag is 0 a length of 1 is OK.
See the function hseCalc for some more details.

VARIABLE : emission
DATA TYPE: Index
DESCRIPTION:
Boolean to include emssion in the calculation of spectra.
If this variable is set to 0 (zero) pure transmission calculations
will be simulated and, for example, yCalc will give optical
thicknesses instead of radiation intensities.

VARIABLE : za_pencil
DATA TYPE: Vector
DESCRIPTION:
Pencil beam zenith angle grid [deg].
The observation direction is specified by the angle between zenith
and the LOS.

VARIABLE : z_tan
DATA TYPE: Vector
DESCRIPTION:
Tangent altitude for each spectrum [m].
These tangent altitudes include the effect of refraction (if set).
In the case of a ground intersection, a geometrical prolongation
below the ground is applied to determine the tangent altitude.
For upward observations where there are no tangent altitudes,
*z_tan* is set to 999 km.
It should be noted that the LOS calculations take *za_pencil* as
input, not *z_tan*. However, *za_pencil* can be calculated from
*z_tan* by the function *zaFromZtan*.
```

```
VARIABLE : z_plat
DATA TYPE: Numeric
DESCRIPTION:
Vertical altitude, above the geoid, of the observation platform [m].

VARIABLE : l_step
DATA TYPE: Numeric
DESCRIPTION:
The maximum length, along the LOS, between the points of LOS [m].
The final step length will in most cases equal the selected length.
There are two rare exceptions:
  1. Downward observations from within the atmosphere, where the step
     length is adjusted downwards to get an integer number of steps
     between the sensor and the tangent or ground point.
  2. Limb sounding and the distance from the tangent point to the
     atmospheric limit (the highest absorption altitude) is smaller
     than the selected length. The length is then adjusted to this
     distance

VARIABLE : refr
DATA TYPE: Index
DESCRIPTION:
Boolean for inclusion of refraction (0=no refraction, 1=refraction).

VARIABLE : refr_lfac
DATA TYPE: Index
DESCRIPTION:
This factor determines the step length used during the ray tracing
performed when considering refraction.
The step length applied is *l_step* divided by *refr_lfac*.
Accordingly, this factor gives how many ray tracing steps that are
performed for each step of the LOS.

VARIABLE : refr_model
DATA TYPE: String
DESCRIPTION:
A string giving what parameterization to use for the calculation of
refractive index. See *refrCalc* for existing choices.

VARIABLE : refr_index
DATA TYPE: Vector
DESCRIPTION:
The refractive index at the pressure levels in p_abs [-].


VARIABLE : r_geoid
DATA TYPE: Numeric
DESCRIPTION:
The local curvature radius of the geoid along the LOS [m].

VARIABLE : z_ground
DATA TYPE: Numeric
DESCRIPTION:
The vertical altitude above the geoid of the ground [m].
```

```
VARIABLE : t_ground
DATA TYPE: Numeric
DESCRIPTION:
The physical temperature of the ground [K].

VARIABLE : e_ground
DATA TYPE: Vector
DESCRIPTION:
The ground emission factor for the frequencies in f_mono [0-1].

VARIABLE : los
DATA TYPE: Los
DESCRIPTION:
Structure to define the line of sight (LOS). See los.h for
definition of the structure.

VARIABLE : source
DATA TYPE: ArrayOfMatrix
DESCRIPTION:
Mean source function between the points of the LOS.

VARIABLE : trans
DATA TYPE: ArrayOfMatrix
DESCRIPTION:
The transmissions between the points of the LOS [-].

VARIABLE : y_space
DATA TYPE: Vector
DESCRIPTION:
Radiation entering the atmosphere at the top of the atmosphere,
typically cosmic background radiation. This variable is most easily
set by the function *y_spaceStd*.

VARIABLE : y
DATA TYPE: Vector
DESCRIPTION:
The working set of spectra.
The spectra from the different zenith angles are appended to form *y*.

VARIABLE : y0
DATA TYPE: Vector
DESCRIPTION:
A reference spectrum. This variable can be used e.g. to save a copy
of *y* or to compare the spectra before and after some operation(s).

VARIABLE : h
DATA TYPE: Matrix
DESCRIPTION:
The H matrix.

Can be used to apply the sensor model to monochromatic pencil beam
spectra and weighting functions.


VARIABLE : absloswfs
DATA TYPE: ArrayOfMatrix
```

```
DESCRIPTION:
Line of sight weighting functions.
See AUG for definition of this quantity.

VARIABLE : k_grid
DATA TYPE: Vector
DESCRIPTION:
Retrieval grid to be used in calculation of weighting functions (WFs)
For example, *k_grid* is the pressure altitude grid for species WFs.
Not all WFs need 'k_grid* as input.

VARIABLE : k
DATA TYPE: Matrix
DESCRIPTION:
The weighting functions (WFs) for a single retrieval/error group.

VARIABLE : k_names
DATA TYPE: ArrayOfString
DESCRIPTION:
Names of the retrieval identies in *k*.

VARIABLE : k_aux
DATA TYPE: Matrix
DESCRIPTION:
Auxiliary data for *k*. The number of rows of this matrix equals the
length of the state vector for the retrieval group (the number of
columns of k).
The columns hold different quantities:
  Col 1: retrieval grid (or correspondingly)
  Col 2: a priori values

VARIABLE : kx
DATA TYPE: Matrix
DESCRIPTION:
The state weighting function matrix.

VARIABLE : kx_names
DATA TYPE: ArrayOfString
DESCRIPTION:
Names of the retrieval identities in *kx*.

VARIABLE : kx_lengths
DATA TYPE: ArrayOfIndex
DESCRIPTION:
The length of the state vector for each retrieval identity in *kx*.

VARIABLE : kx_aux
DATA TYPE: Matrix
DESCRIPTION:
Auxiliary data for *kx*. As *k_aux* but with the data of the
different retrieval groups appended vertically.

VARIABLE : kb
DATA TYPE: Matrix
DESCRIPTION:
The model parameters weighting function matrix.
```

```
VARIABLE : kb_names
DATA TYPE: ArrayOfString
DESCRIPTION:
Names of the model parameter identities in *kb*.

VARIABLE : kb_lengths
DATA TYPE: ArrayOfIndex
DESCRIPTION:
The length of the model vector for each retrieval identity in *kb*.

VARIABLE : kb_aux
DATA TYPE: Matrix
DESCRIPTION:
Auxiliary data for *kb*. As *k_aux* but with the data of the
different forward model groups appended vertically.

VARIABLE : S_S
DATA TYPE: Matrix
DESCRIPTION:
Stores the accuracy of the spectroscopic parameters read from catalog
This are necessary for the the spectroscopic error analysis
number columns = 2; first keeps the absolute error, second the error in percents
number lines equal of spectroscopic parameters investigated (the number of
columns of k).

VARIABLE : batchname
DATA TYPE: String
DESCRIPTION:
Default basename for batch data.

VARIABLE : ybatch
DATA TYPE: Matrix
DESCRIPTION:
A batch of spectra.
The spectra are stored as columns in a matrix

VARIABLE : absbatch
DATA TYPE: ArrayOfMatrix
DESCRIPTION:
A batch of absorption coefficients.
FIXME

VARIABLE : jacbatch
DATA TYPE: ArrayOfMatrix
DESCRIPTION:
A batch of jacobian matrices.
FIXME

VARIABLE : radiosonde_data
DATA TYPE: ArrayOfMatrix
DESCRIPTION:
An array of Matrix holding data for many radiosonde launches. The
dimension of the Array is the number of radiosonde launches. Each
Matrix within the Array has dimension nx4, where n is the number of
pressure levels. The columns of the Matrix are:
```

```
pressure [Pa] temperature [K] altitude [m] VMR [1]

VARIABLE : coolrate
DATA TYPE: Matrix
DESCRIPTION:
Cooling rate matrix, in unit of K/s/Hz.Dimensions: [f_mono.nelem(), z_crates.nelem()]

VARIABLE : p_coolrate
DATA TYPE: Vector
DESCRIPTION:
Pressures for which to calculate cooling rates.

VARIABLE : method_list
DATA TYPE: ArrayOfIndex
DESCRIPTION:
A list of methods. See "arts -d MethodListDefine"
for an explanation what a method list is.
```

Appendix B

ARTS Units and Conversion Factors

This appendix gives an overview of the arts physical units and how they can be converted to other common unit systems. The internal physical units of arts for frequencies is [Hz], for pressure [Pa] and for the absorption coefficient [1/m]. Table B.1 gives some common conversion coefficients to arts units.

Especially for the MPM model versions [*Liebe and Layton*, 1987; *Liebe*, 1989; *Liebe et al.*, 1993] we get for the pre-factor:

$$\frac{4 \cdot \pi}{c} \cdot 10 \cdot \log(e) = 0.1820 \cdot 10^{6} \text{ dB/km/GHz} \tag{B.1}$$

$$= 0.1820 \cdot 10^{-6} \text{ dB/m/Hz} \tag{B.2}$$

where c is the speed of light ($c = 2.9979 \cdot 10^{8}$m/s).

History

2001-21-05 Created by Thomas Kuhn.

x	g/cm^3	=	y	kg/m^3	$\Leftrightarrow$	y	=	x	$\times$	10^{3}
x	g/m^3	=	y	kg/m^3	$\Leftrightarrow$	y	=	x	$\times$	10^{-3}
x	GHz	=	y	Hz	$\Leftrightarrow$	y	=	x	$\times$	10^{9}
x	1/GHz	=	y	1/Hz	$\Leftrightarrow$	y	=	x	$\times$	10^{-9}
x	hPa	=	y	Pa	$\Leftrightarrow$	y	=	x	$\times$	10^{2}
x	1/hPa	=	y	1/Pa	$\Leftrightarrow$	y	=	x	$\times$	10^{-2}
x	1/cm	=	y	1/m	$\Leftrightarrow$	y	=	x	$\times$	10^{2}
x	1/km	=	y	1/m	$\Leftrightarrow$	y	=	x	$\times$	10^{-3}
x	dB	=	y	Np	$\Leftrightarrow$	y	=	x	/	$[10 \cdot \log(e)]$
x	dB/km	=	y	1/m	$\Leftrightarrow$	y	=	x	$\times$	$10^{-3}/[10 \cdot \log(e)]$
x	Np/km	=	y	1/m	$\Leftrightarrow$	y	=	x	$\times$	10^{-3}

Table B.1: Conversion factors for the physical units used in arts.