ARTS User Guide

by

Stefan Bühler
Institute of Remote Sensing
University of Bremen, Germany

and

Patrick Eriksson
Department of Radio and Space Science
Chalmers University of Technology, Sweden

September 26, 2000
ARTS Version 0.0

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`**.**

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.

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.

1.2 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.

[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.

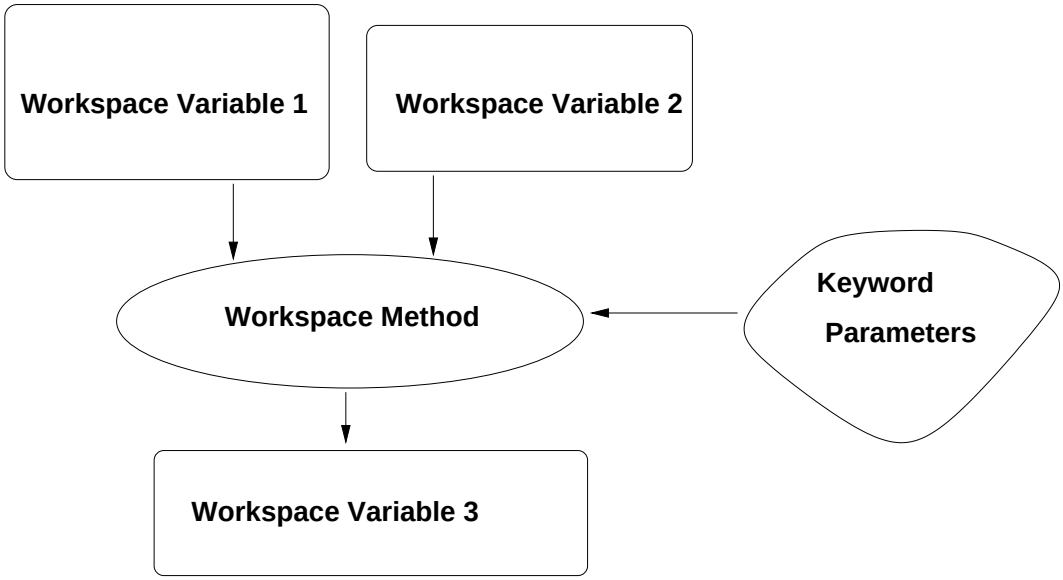

Figure 1.1: *Specific* workspace methods act on specific workspace variables to generate other specific workspace variables. Additional input parameters can be specified as keyword parameters in the controlfile.

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 10.

1.3 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 `VectorReadFromFile`, which can be used to read any workspace variable which is a vector from an ASCII file. For example

```
VectorReadFromFile(f_grid){"freqeuency_grid.dat"}
```

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

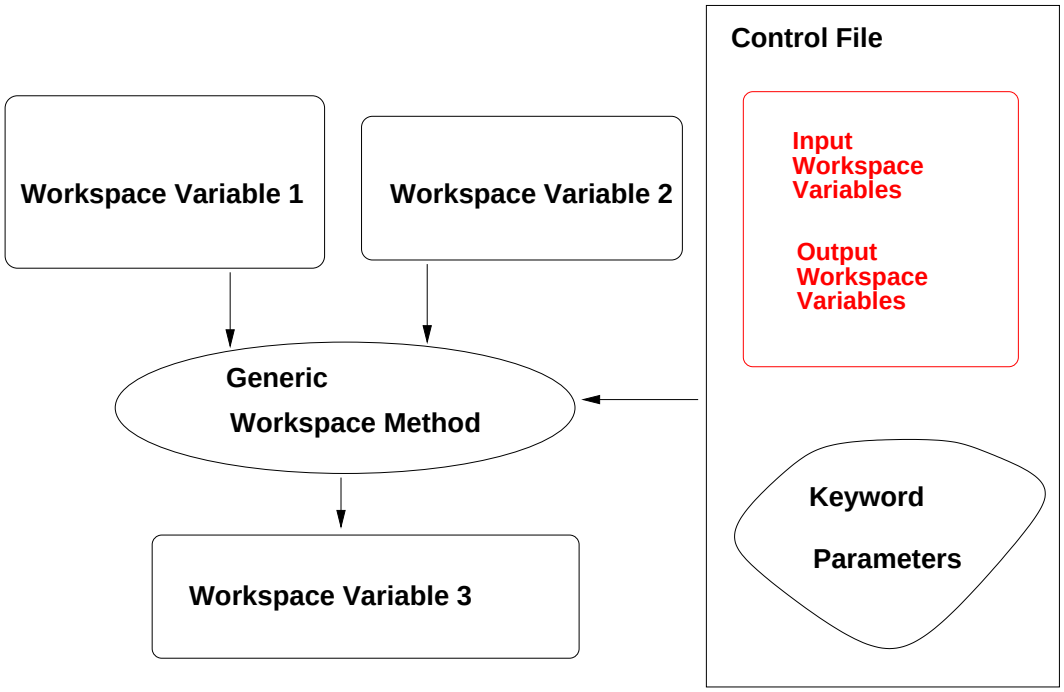

Figure 1.2: For *generic* workspace methods the workspace variables to act on are specified in the controlfile.

types already known to the program (for example vectors or matrices). Section 1.4 gives a short example of a controlfile which illustrates the use of both generic and specific methods.

1.4 An example controlfile

```
# An example ARTS controlfile that calculates absorption
# coefficients.
# SAB 16.06.2000

# --------------------< A specific method >--------------------
#                       -------------------
# Read the spectroscopic line data from the HITRAN catalogue and
# create the workspace variable ‘lines’:
linesReadFromHitran {
   filename = "../../data/spectroscopy/hitran96/hitran96_h2o.par"
   fmin     = 0
   fmax     = 1000e12
}

# Optionally write the line list to a file:
linesWriteToFile{""}
```

```
# This defines the list of tag groups (‘tag_groups’). Absorption
# coefficients will be calculated separately for each tag group. This
# is necessary in order to calculate weighting functions later on.
# The lines are assigned to the tag groups in the order as the groups
# are specified here. That means the last group H2O gets assigned all
# the H2O lines that do not fit in any other group.
tag_groupsDefine{
      [ "H2O-161",
        "H2O-181",
        "H2O-171",
        "H2O" ]
}

# This separates the lines into the different tag groups and creates
# the workspace variable ‘lines_per_tg’:
lines_per_tgCreateFromLines{}

lines_per_tgWriteToFile{""}


# --------------------< A generic method >--------------------
#                       ------------------
# Read the pressure, temperature, and altitude profiles and create
# the workspace variable ‘raw_ptz_1d’:
MatrixReadFromFile (raw_ptz_1d)
        {"../../data/atmosphere/fascod/midlatitude-summer.tz.am"}

# The same for the input VMR profiles:
raw_vmrs_1dReadFromScenario
        {"../../data/atmosphere/fascod/midlatitude-summer"}

# Optionally write this to a file:
ArrayOfMatrixWriteToFile (raw_vmrs_1d) {""}


# Create the pressure grid ‘p_abs’:
VectorLinSpace(p_abs){
        start = 1000
        stop  = 1
        step  = -1
}

VectorWriteToFile(p_abs){""}
```

```
# Now interpolate all the raw atmospheric input onto the pressure
# grid and create the atmospheric variables ‘t_abs’, ‘z_abs’, ‘vmrs’
AtmFromRaw1D{}

# Optionally write these to files:
VectorWriteToFile (t_abs) {""}
VectorWriteToFile (z_abs) {""}
ArrayOfVectorWriteToFile (vmrs)  {""}


# Create the frequency grid ‘f_abs’:
VectorLinSpace(f_abs){
        start = 1
        stop  = 1000
        step  = 1
}


# Calculate absorption coefficients, both total (‘abs’) and
# separately for each tag group (‘abs_per_tg’):
absCalc{}

# These we definitely want to write to files!
MatrixWriteToFile (abs) {""}
ArrayOfMatrixWriteToFile (abs_per_tg) {""}
```

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 here.

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 $\mathbf{x}$, 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

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

History

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

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. It should be noted that a possible data reduction is also part of the forward model.

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 (i.e. $\mathbf{i}^i = I(\nu^i, \phi^i)$, 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 if the part of the vectors used is made 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 $\mathbf{H}$ can be set up to 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 5 and 6, 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.

The practical calculation of weighting functions is discussed in detail in Sections 7 and 9.

2.3.2 Transformation between vector spaces

It could be of interest to transform a weighting function matrix from one vector space to another. 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 $\mathbf{B} \in \mathbf{R}^{pxn}$. 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 $\mathbf{A}_{\mathbf{x}} \in \mathbf{R}^{pxn}$ and $\mathbf{A}_{\mathbf{x}'} \in \mathbf{R}^{pxp}$, that is, only $\mathbf{A}_{\mathbf{x}'}$ is square.

Chapter 3

Basic radiative transfer

This section presents the atmospheric radiative transfer equation (RTE) for a non-scattering atmosphere in local thermodynamic equilibrium. However, the calculation scheme described here can easily be extended to include effects of e.g. scattering. The radiative transfer equation gives the monochromatic (infinite frequency resolution) pencil beam (infinite spatial resolution) spectrum. The main problem, in this context, 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 and such observations are also discussed briefly last in the section.

The equations of this section are valid for monochromatic pencil beam spectra, that is, no effects of the sensor are considered. How to incorporate sensor effects in the spectra is discussed separately (Sec. 5).

3.1 Introduction

Atmospheric radiative transfer can be expressed generally as

$$I = I_1 e^{-\int_{l_1}^{l_2} \kappa(l) dl} + \int_{l_1}^{l_2} \kappa(l)\sigma(l) e^{-\int_{l}^{l_2} \kappa(l') dl'} dl \tag{3.1}$$

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 3.1 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.

Note that Eq. 3.1 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:

$$\sigma = B(\nu, T) = \frac{2h\nu^3}{c^2} \frac{1}{e^{h\nu/k_B T} - 1} \tag{3.2}$$

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

$$I = I_1 e^{-\tau} + \sigma \left(1 - e^{-\tau}\right) \tag{3.3}$$

where τ is the optical thickness

$$\tau = \int_{l_1}^{l_2} \kappa(l) dl \tag{3.4}$$

The transmission corresponding to τ is

$$\zeta = e^{-\tau} \tag{3.5}$$

3.2 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, e.g. species WFs, 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.

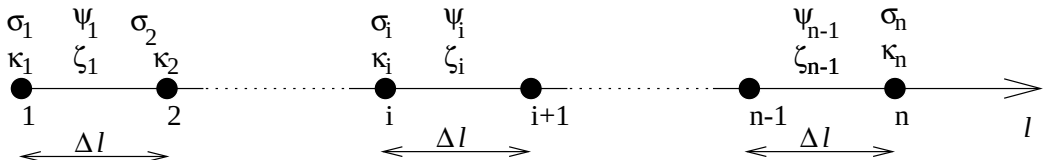

Figure 3.1: Schematic description of the LOS and associated variables. The absorption and the source function at the LOS points are denoted κ_i and σ_i, respectively, while ζ_i is the transmission between the points and Ψ_i is the mean of neighbouring source function values. Only $zeta$ and Ψ are stored 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 4.

3.3 Practical solution

The LOS is here assumed to be defined with n points where the distance between the points is constant (see Fig 3.1). 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.

3.3.1 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

$$\tau_i = \frac{\Delta l}{2}\left(\kappa_i + \kappa_{i+1}\right), \quad 1 \leq i < n \tag{3.6}$$

The relationship between the optical thicknesses and the transmission is

$$\zeta_i = e^{-\tau_i} \tag{3.7}$$

Note that

$$e^{-(\tau_1+\tau_2\ldots\tau_n)} = \zeta_1\zeta_2\ldots\zeta_n \tag{3.8}$$

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]. For the moment, linear interpolation is the only choice, but if it is found that cubic and/or spline interpolation give better accuracy, such interpolations will also be implemented.

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

3.3.2 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:

$$\Psi_i = \frac{\sigma_i + \sigma_{i+1}}{2}, \quad 1 \leq i < n \tag{3.9}$$

The source function at the LOS points (σ) is simply by interpolating linearly the temperature profile, and calculating the Planck function (Eq. 3.2) 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. 3.2), 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.

3.3.3 Solving the radiative transfer equation

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

$$I = I_1 \prod_{j=1}^{n-1} \zeta_j + \sum_{i=1}^{n-1} \left[\Psi_i (1 - \zeta_i) \prod_{j=i+1}^{n-1} \zeta_j \right] \tag{3.10}$$

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 3.3, the following iterative expression can be determined [*Eriksson and Merino*, 1997]

$$I_{i+1} = I_i \zeta_i + \Psi_i \left(1 - \zeta_i\right) \qquad i = 1, 2, ..., n-1 \tag{3.11}$$

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.

3.3.4 Considering ground reflection

The effect of a ground reflection is modeled as

$$I^{after} = I^{before}(1 - e) + eB(\nu, T_{ground}) \tag{3.12}$$

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 4.6.

3.4 Total atmospheric transmission

The atmospheric emission can be neglected if the observation is performed towards a sufficiently strong source, such as the Sun, and the measurement gives basically the total atmospheric transmission, ζ^{tot}. This transmission is

$$\zeta^{tot} = e^{-\int_{l_1}^{l_2} \kappa(l) dl} \tag{3.13}$$

The corresponding iterative formula used in the forward model is simply (cf. Eq. 3.11)

$$\zeta^{tot} = \prod_{i=1}^{n-1} \zeta_i \tag{3.14}$$

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. The transmission can here be calculated as

$$\zeta^{tot} = \prod_{i=1}^{n-1} \zeta_i^2 \tag{3.15}$$

If there is a ground reflection, it is considered as

$$\zeta^{tot} = (1 - e) \prod_{i=1}^{n-1} \zeta_i \tag{3.16}$$

where e is the ground emission factor.

Chapter 4

Line of sight, 1D

This section describes how the line of sight (LOS) can be 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.

4.1 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. 4.1):

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.

000914 First version finished by Patrick Eriksson.

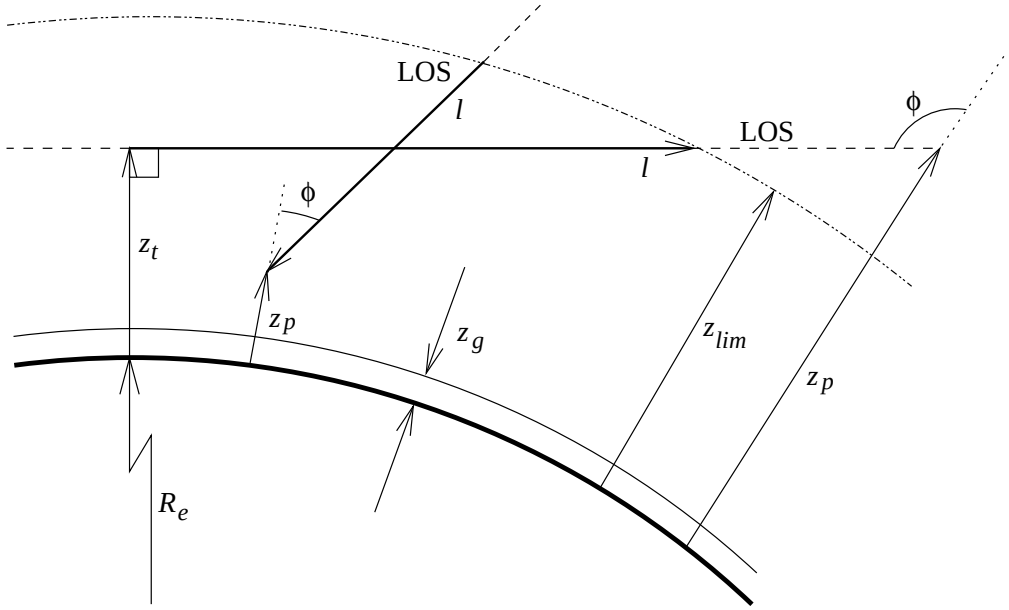

Figure 4.1: Schematic description of the main variables of the observation geometry and the LOS. R_e is the Earth radius. Other variables defined in the text.

If $\phi > 90^\circ$ 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:

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

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.

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

4.2 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. This question is discussed a bit further in Section 4.6. 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.

4.3 The step length

As described in Section 3, 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 4.4.3.

4.4 Geometrical calculations

Most of the equations of this section are described further, both in text and by figures (with a slightly different notation and definition of the zenith angle) in Section 4 of *Eriksson and Merino* [1997].

4.4.1 Upward looking

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

$$(R_e + z)^2 = (R_e + z_p)^2 + l^2 + 2l(R_e + z)\cos(\phi) \tag{4.1}$$

This equation gives

$$z = \sqrt{(R_e + z_p)^2 + l^2 + 2l(R_e + z)\cos(\phi)} - R_e \tag{4.2}$$

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

$$l_{lim} = \sqrt{(R_e + z_{lim})^2 - (R_e + z_p)^2 \sin^2(\phi)} - (R_e + z_p)\cos(\phi) \tag{4.3}$$

4.4.2 Limb sounding

The tangent altitude is

$$z_t = (R_e + z_p)\sin(\phi) - R_e \qquad \phi \geq 90^\circ \tag{4.4}$$

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 Pythagorean relation gives the distance from the tangent point to the atmospheric limit:

$$l_{lim} = \sqrt{(R_e + z_{lim}) - (R_e + z_t)} \tag{4.5}$$

If l_{lim} is smaller than upper limit for Δl specified by the user, Δl is set to l_{lim} as also described in Section 4.3.

The vertical altitude as a function of the distance from the tangent point is

$$z = \sqrt{(R_e + z_t) + l^2} - R_e \tag{4.6}$$

If the tangent point is below ground, the LOS is determined by the upward expressions (Sec. 4.4.1) by setting

$$\begin{aligned} z_p &\leftarrow z_g \\ \phi &\leftarrow \sin^{-1}\left((R_e + z_t)/(R_e + z_g)\right) \end{aligned}$$

4.4.3 Downward looking

This observation geometry can be handled by the upward and limb sounding functions by suitable exchange of variables. However, as the lowest point of the LOS is either the tangent point or the ground, and one point of LOS must fit the sensor altitude, the step length must be adjusted to this distance.

The distance between the sensor and a tangent point is

$$l_p = \sqrt{(R_e + z_p) - (R_e + z_t)} \qquad z_t \geq z_g \tag{4.7}$$

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

$$l_p = \sqrt{(R_e + z_p) - (R_e + z_t)} - \sqrt{(R_e + z_g) - (R_e + z_t)} \qquad z_t < z_g \tag{4.8}$$

where z_t is determined by Equation 4.4.

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

$$m = 1 + \mathbf{ceil}(l_{lim}/\Delta l_{max}) \tag{4.9}$$

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

$$\Delta l = \frac{l_{lim}}{m - 1} \tag{4.10}$$

If the tangent altitude is above the ground ($z_{tan} \geq z_t$), the LOS is determined by the same expressions as applied for limb sounding, but with the adjusted value for Δl. If there is an intersection with the ground, the upward looking expressions can be used as described above for limb sounding, again with the adjusted value for Δl.

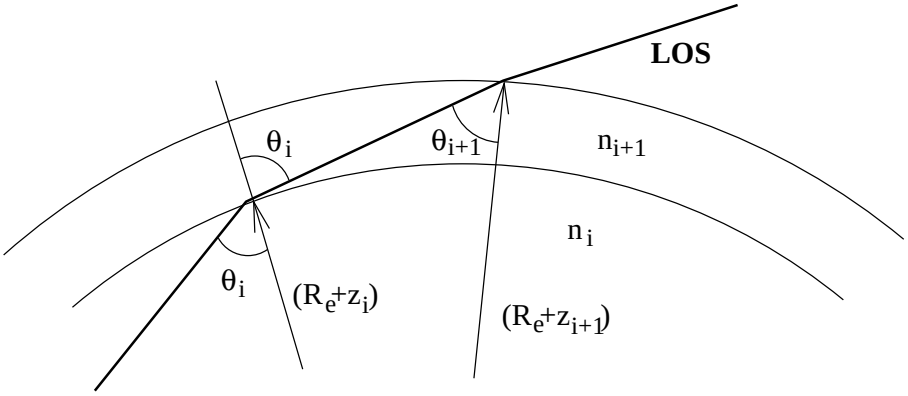

Figure 4.2: Geometry to derive Snell's law for a spherical atmosphere. The Earth radius is R_e, the vertical altitude z, the refractive index n and the angle between the LOS and the vector to the Earth centre θ.

4.5 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 limb sounding observation the tangent point is moved both vertically and horisontally. If the atmosphere is assumed to be horizontally stratified, as done here (1D), a horisontal 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.

4.5.1 General theory

When determining the LOS through the atmosphere geometrical optics can be applied because the change of the refractivce index over a wavelength can be neglected. Applying Snell's law to the geometry shown in Figure 4.2 gives

$$n_i \sin(\theta_i) = n_{i+1} \sin(\theta_i') \tag{4.11}$$

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

$$\frac{\sin(\theta_{i+1})}{R_e + z_i} = \frac{\sin(180^\circ - \theta_{i+1}')}{R_e + z_{i+1}} = \frac{\sin(\theta_i')}{R_e + z_{i+1}} \tag{4.12}$$

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]:

$$c = (R_e + z_i) n_i \sin(\theta_i) = (R_e + z_{i+1}) n_{i+1} \sin(\theta_{i+1}) \tag{4.13}$$

where c is a constant. With other words, the Snell's law for spherical atmospheres says 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 4.13 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

$$g(z) = \frac{1}{\cos(\theta)} \tag{4.14}$$

which can be rewritten using trigonometric identities and Equation 4.13:

$$g(z) = \frac{(R_e + z)n(z)}{\sqrt{(R_e + z)^2 n^2(z) - c^2}} \tag{4.15}$$

A possible solution for calculating the LOS would be to integrating the geometrical term as [*Eriksson et al.*, 2000]

$$l_1^2 = \int_{z_1}^{z_2} g(z)\mathrm{d}z$$

where z_1 and z_2 are two vertical altitudes and l_1^2 the length along the LOS between these two altitudes. However, this appraoch is problematic for limb sounding as the geometric factor is singular at the tangent point (Sec. !!).

4.5.2 The relative geometrical factor

To avoid the singularity of the geometrical factor at tangent points, the relative geometrical factor is here introduced. This factor is defined as

$$h(z) = \frac{(R_e + z)n(z)}{\sqrt{(R_e + z)^2 n^2(z) - c^2}} \tag{4.16}$$

4.6 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 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. 3.12) 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.

When solving the radiative transfer (Eq. 3.11) for 2D cases, the ground reflection is treated between evaluating step i_g and i_{g-1} (the iteration goes from n to 1), as shown in Figure 4.4. Ground reflections for 2D cases are treated internally in ARTS by the upward looking functions (Sec. 4.2).

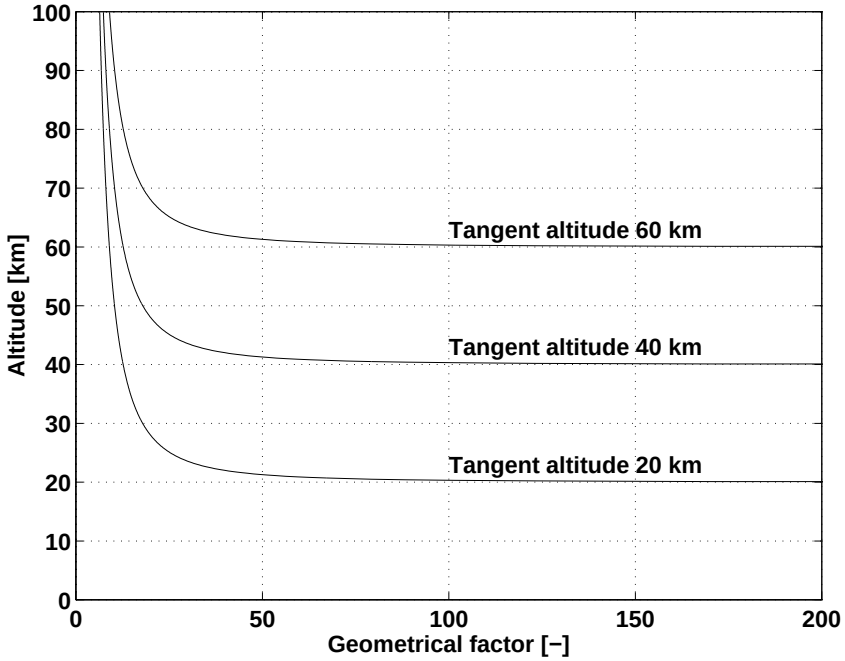

Figure 4.3: The geometrical factor, as a function of altitude, for limb sounding and three tangent altitudes.

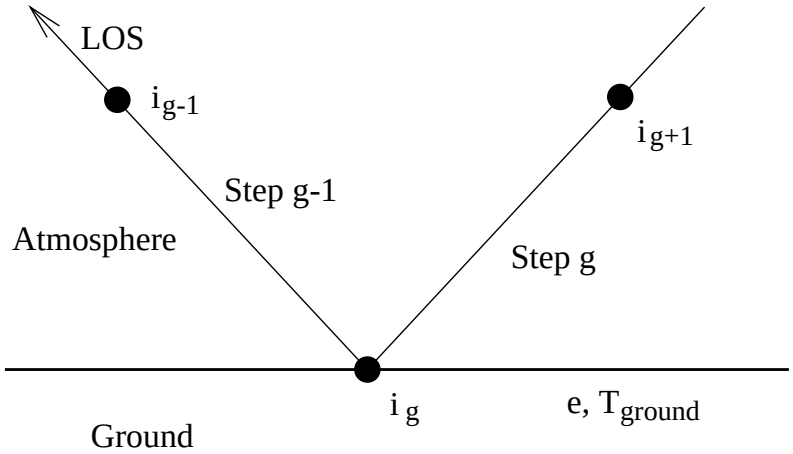

Figure 4.4: Schematic of ground reflections for 2D cases. The index of the point corresponding to the ground is i_g. Point 1 of the LOS is the point closest to the sensor.

Chapter 5

Sensor modeling

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.

5.1 Implementation strategy

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]:

$$\mathbf{y} = \mathbf{H}_n \ldots \mathbf{H}_2 \mathbf{H}_1 \mathbf{i} + \varepsilon \tag{5.1}$$

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)

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

History

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

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

5.2 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:

$$\mathbf{hg} = \int f(x)g(x)\mathrm{d}x \tag{5.2}$$

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. Here it will be assumed that both functions are piecewise linear.

Following Figure 5.1, 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:

$$g(x) = g_1 + (g_2 - g_1)\frac{x - x_1}{x_4 - x_1} = g_1\frac{x_4 - x}{x_4 - x_1} + g_2\frac{x - x_1}{x_4 - x_1} \tag{5.3}$$

which can be rewritten as

$$g(x) = g_1(a + bx) + g_2(c - bx), \qquad x_1 \le x \le x_4 \tag{5.4}$$

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:

$$f(x) = (d + ex), \qquad x_2 \le x \le x_3 \tag{5.5}$$

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 5.2 can now for ranges between x_2 and x_3 be calculated analytically in a straightforward manner:

$$\int_{x_a}^{x_b} f(x)g(x)\mathrm{d}x = \int_{x_a}^{x_b} (d + ex)(g_1(a + bx) + g_2(c - bx))\mathrm{d}x = \ldots =$$
$$\left[g_1 x\Big(ad + \frac{x}{2}(bd + ae) + \frac{x^2}{3}be\Big) + g_2 x\Big(cd + \frac{x}{2}(ce - bd) - \frac{x^2}{3}be\Big)\right]_{x_a}^{x_b} \tag{5.6}$$

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,

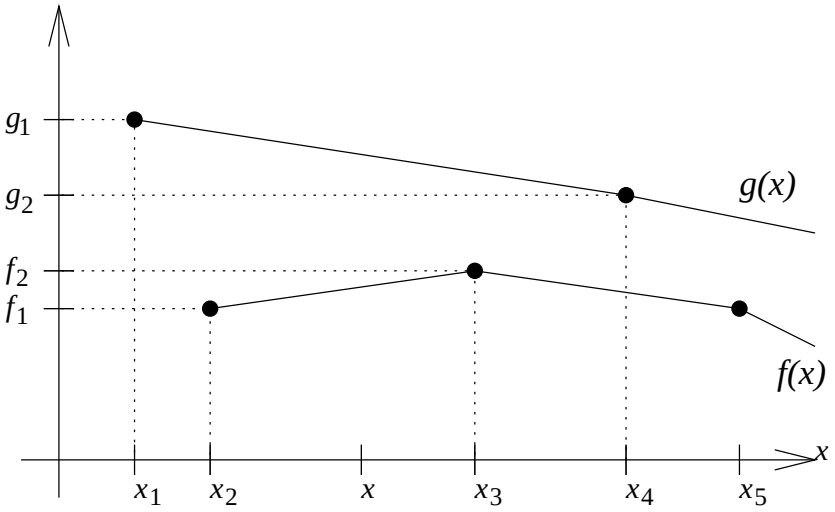

Figure 5.1: The quantities used in Section 5.2.

$f = 0$ for $x < x_2$). For the case shown in Figure 5.1, the integration order would be $(x_a, x_b) = (x_2, x_3)$, $(x_a, x_b) = (x_3, x_4)$, $(x_a, x_b) = (x_4, x_5)$ ...

Using Equation 5.6, 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.

5.3 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

$$\nu' = 2\nu_{LO} - \nu \tag{5.7}$$

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

$$I_{IF}(\nu) = \frac{f_s(\nu)I(\nu) + f_s(\nu')I(\nu')}{f_s(\nu) + f_s(\nu')} \tag{5.8}$$

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

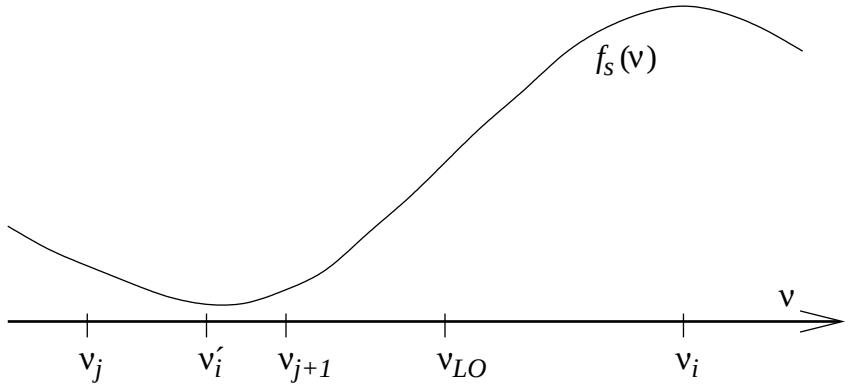

Figure 5.2: Schematic description of image frequency and sideband filtering.

If the intensity is assumed to vary linearly between the points of the frequency grid, Equation 5.8 can be written as

$$\begin{aligned} I_{IF}(\nu^i) &= \frac{1}{f_s(\nu_i)+f_s(\nu_i')}\Big[f_s(\nu_i)I(\nu_i)+ \\ &\quad +\frac{f_s(\nu_i')}{\nu_{j+1}-\nu_j}\Big(I(\nu_j)(\nu_{j+1}-\nu_i')+I(\nu_{j+1})(\nu_i'-\nu_j)\Big)\Big] \end{aligned} \tag{5.9}$$

where f_s for the different frequencies is obtained by an analytical expression or 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

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

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

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

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.

Chapter 6

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.

6.1 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

$$\mathbf{h} = [0, \ldots, 0, \frac{1}{n_v}, 0, \ldots, 0, \frac{1}{n_v}, 0, \ldots, 0, \frac{1}{n_v}, 0, \ldots, 0] \tag{6.1}$$

where n_v is the number of viewing angles to combine.

6.2 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

$$\mathbf{y}^j = \frac{1}{\sum_{i=i_1}^{i_2} \Delta\nu^i} \sum_{i=i_1}^{i_2} \Delta\nu^i (\mathbf{y}')^i \tag{6.2}$$

Row j of $\mathbf{H}_d$ is accordingly

$$\mathbf{h}^i = \frac{\Delta\nu^i}{\sum_{i=i_1}^{i_2} \Delta\nu^i}, \qquad i_1 \le i \le i_2 \tag{6.3}$$

Other values of $\mathbf{h}$ are zeros. The matrix $\mathbf{H}_d$ is for data binning highly sparse.

History

000321 Created and written by Patrick Eriksson.

6.3 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

$$\mathbf{S_y} = \mathbf{E}\Lambda\mathbf{E}^T \tag{6.4}$$

where Λ is a diagonal matrix holding the eigenvalues corresponding to the eigenvectors, the columns of $\mathbf{E}$. The eigenvectors form an orthogonal basis:

$$\mathbf{I} = \mathbf{E}_j^T\mathbf{E}_j \tag{6.5}$$

where $\mathbf{E}_j$ signifies the j first columns of the matrix.

The data reduction for this case is performed as

$$\mathbf{y} = \mathbf{E}_j^T\mathbf{y}' \tag{6.6}$$

that is

$$\mathbf{H}_d = \mathbf{E}_j^T \tag{6.7}$$

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.

Chapter 7

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.

7.1 Calculation approaches

7.1.1 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

$$\mathbf{K}_{\mathbf{x}}^{p} = \frac{\mathcal{F}(\mathbf{x} + \Delta\mathbf{x}^{p}\mathbf{e}^{p}, \mathbf{b}) - \mathcal{F}(\mathbf{x}, \mathbf{b})}{\Delta\mathbf{x}^{p}} \tag{7.1}$$

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)

$$\mathbf{K}_{\mathbf{x}}^{p} = \mathbf{H}\left[\frac{\mathcal{F}_{r}(\mathbf{x}_{r} + \Delta\mathbf{x}^{p}\mathbf{e}^{p}, \mathbf{b}_{r}) - \mathcal{F}_{r}(\mathbf{x}_{r}, \mathbf{b}_{r})}{\Delta\mathbf{x}^{p}}\right] \tag{7.2}$$

where $\mathbf{x}_{r}$ is the atmospheric part of the state vector etc (see further Sec. 2).

7.1.2 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,

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

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]

$$\mathbf{K}_{\mathbf{x}}^p = \mathbf{H}\frac{\partial \mathbf{i}}{\partial \mathbf{x}^p} = \mathbf{H}\left[\frac{\partial \mathbf{i}}{\partial \mathbf{S}^p}\frac{\partial \mathbf{S}^p}{\partial \mathbf{x}^p} + \frac{\partial \mathbf{i}}{\partial \mathbf{k}^p}\frac{\partial \mathbf{k}^p}{\partial \mathbf{x}^p}\right] \tag{7.3}$$

where $\mathbf{S}^p$ and $\mathbf{k}^p$ are the source function and the absorption at the (vertical) altitude p, respectively.

The absorption and source function in Equation 7.3 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 7.3 is expanded one step further

$$\mathbf{K}_{\mathbf{x}}^p = \mathbf{H}\left[\frac{\partial \mathbf{i}}{\partial \sigma}\frac{\partial \sigma}{\partial \mathbf{S}^p}\frac{\partial \mathbf{S}^p}{\partial \mathbf{x}^p} + \frac{\partial \mathbf{i}}{\partial \kappa}\frac{\partial \kappa}{\partial \mathbf{k}^p}\frac{\partial \mathbf{k}^p}{\partial \mathbf{x}^p}\right] \tag{7.4}$$

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 function line of sight weighting functions (source LOS WFs) and is discussed in Section 7.3. The term $\partial \mathbf{i}/\partial \kappa$ is denoted as absorption LOS WFs and is discussed in Section 7.2. These terms are treated seperately 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 7.7. 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 seperately.

7.2 Absorption line of sight weighting functions

The absorption line of sight weighting functions are defined as

$$\mathbf{K}_{\kappa}^q = \frac{\partial \mathbf{i}}{\partial \kappa^q} \tag{7.5}$$

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 3.

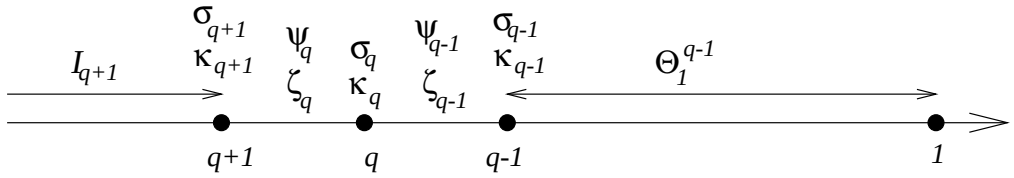

Figure 7.1: The terms used for the derivation of line of sight weighting functions when the individual atmospheric parts are passed a single time. The variables are defined in Figure 3.1.

7.2.1 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 7.2.2 and 7.2.3, respectively.

By rewriting Equation 3.10, the monochromatic pencil beam intensity can be expressed as (see Fig. 7.1)

$$\begin{aligned} I &= I_2\zeta_1 + \psi_1(1-\zeta_1), \quad q = 1 \\ I &= \Big[I_{q+1}\zeta_q\zeta_{q-1} + \psi_q(1-\zeta_q)\zeta_{q-1} + \psi_{q-1}(1-\zeta_{q-1})\Big]\Theta_1^{q-1}, \quad 1 < q < n \\ I &= \Big[I_n\zeta_{n-1} + \psi_{n-1}(1-\zeta_{n-1})\Big]\Theta_1^{n-1}, \quad q = n \end{aligned} \tag{7.6}$$

where it assumed that the LOS has n points, index 1 is the point closest to the sensor,

$$I_q = I_n\Theta_q^n + \sum_{i=q}^{n-1}\psi_i(1-\zeta_i)\Theta_q^i, \quad 1 \le q < n \tag{7.7}$$

is the intensity reaching point q along the LOS, I_n is the radiation at point n (the radiation entering the atmosphere), and

$$\Theta_q^p = \prod_{i=q}^{p-1}\zeta_i \quad \text{for} \quad p > q, \quad \text{and} \quad \Theta_p^p = 1 \tag{7.8}$$

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 seperately below.

The transmissions ζ_{q-1} and ζ_q are separated in Equation 7.6 as they are the only terms including the absorption at point q. For example

$$\zeta_{q-1} = e^{-\Delta l(\kappa_{q-1}+\kappa_q)/2} \tag{7.9}$$

and we have that

$$\frac{\partial\zeta_q}{\partial\kappa_q} = -\frac{\Delta l}{2}\zeta_q \tag{7.10}$$

$$\frac{\partial \zeta_{q-1}}{\partial \kappa_q} = -\frac{\Delta l}{2}\zeta_{q-1} \tag{7.11}$$

$$\frac{\partial \zeta_{q-1}\zeta_q}{\partial \kappa_q} = -\Delta l \zeta_{q-1}\zeta_q \tag{7.12}$$

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

$$\mathbf{K}_\kappa^q = -\frac{\Delta l}{2}\Big[2I_{q+1}\zeta_q\zeta_{q-1} + \psi_q(1-2\zeta_q)\zeta_{q-1} - \psi_{q-1}\zeta_{q-1}\Big]\Theta_1^{q-1}, \quad 1 < q < n \tag{7.13}$$

which can be rewritten as

$$\begin{aligned}
\mathbf{K}_\kappa^q &= -\frac{\Delta l}{2}[I_2 - \psi_1]\Theta_1^2, \quad q = 1 \\
\mathbf{K}_\kappa^q &= -\frac{\Delta l}{2}[2(I_{q+1} - \psi_q)\zeta_q + \psi_q - \psi_{q-1}]\Theta_1^q, \quad 1 < q < n \\
\mathbf{K}_\kappa^q &= -\frac{\Delta l}{2}[I_n - \psi_{n-1}]\Theta_1^n, \quad q = n
\end{aligned} \tag{7.14}$$

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 7.14 can be written

$$\mathbf{K}_\kappa^q = -\frac{\Delta l}{2}[(I_{q+1} - \psi_q)\zeta_q + I_q - \psi_{q-1}]\Theta_1^q, \quad 1 < q < n, \tag{7.15}$$

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 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

$$I_{q+1} = \frac{I_q - \psi_q(1-\zeta_q)}{\zeta_q} \tag{7.16}$$

$$\Theta_1^{q+1} = \Theta_1^q\zeta_q \tag{7.17}$$

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 correspondance to Equation 7.6 for a ground point is (c.f. Equations 3.12 and 3.16)

$$\begin{aligned}
I &= \Big[I_{q+1}\zeta_q(1-e)\zeta_{q-1} + \psi_q(1-\zeta_q)(1-e)\zeta_{q-1} + eB\zeta_{q-1} + \\
&\quad +\psi_{q-1}(1-\zeta_{q-1})\Big]\Theta_1^{q-1}, \quad 1 < q < n
\end{aligned} \tag{7.18}$$

and the corresponding absoprtion LOS WF for this point is (cf. Eq. 7.14)

$$\mathbf{K}_\kappa^q = -\frac{\Delta l}{2}[2(I_{q+1} - \psi_q)\zeta_q(1-e) + \psi_q(1-e) + eB - \psi_{q-1}]\Theta_1^q \tag{7.19}$$

$$\tag{7.20}$$

The intensity and the transmission are here updated as

$$\begin{aligned} I_{q+1} &= \frac{I_q - \psi_q(1-\zeta_q)(1-e) - eB}{\zeta_q(1-e)} \\ \Theta_1^{q+1} &= \Theta_1^q \zeta_q (1-e) \end{aligned}$$

It is noteworthy that the effect of a ground intersection is included in I_1 when the iteration starts.

7.2.2 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. 7.2) 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.

Using a nomenclature similar to the one used for Equation 7.6, the intensity of a limb sounding observations can be expressed as (Fig. 7.2)

$$\begin{aligned} I &= \Big(I_2\Big(\zeta_1\Theta_1^1\Big)^2 + \psi_1(1-\zeta_1)\Big(\Theta_1^1\Big)^2\zeta_1 + I_1^1\zeta_1 + \psi_1(1-\zeta_1)\Big)\Theta_2^n, \quad q=1 \\ I &= \Big[\Big(I_{q+1}\zeta_q\zeta_{q-1} + \psi_q(1-\zeta_q)\zeta_{q-1} + \psi_{q-1}(1-\zeta_{q-1})\Big)\Big(\Theta_1^{q-1}\Big)^2\zeta_{q-1}\zeta_q + \\ &\quad + I_{q-1}^{q-1}\zeta_{q-1}\zeta_q + \psi_{q-1}(1-\zeta_{q-1})\zeta_q + \psi_q(1-\zeta_q)\Big]\Theta_{q+1}^n, \quad 1<q<n \\ I &= \Big(I_n\zeta_{n-1} + \psi_{n-1}(1-\zeta_{n-1})\Big)\Big(\Theta_1^{n-1}\Big)^2\zeta_{n-1} + I_{n-1}^{n-1}\zeta_{n-1} + \\ &\quad + \psi_{n-1}(1-\zeta_{n-1}), \quad q=n \end{aligned} \tag{7.21}$$

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,

$$I_q = I_n\Theta_q^n + \sum_{i=q}^{n-1}\psi_i(1-\zeta_i)\Theta_q^{i-1} \tag{7.22}$$

is the intensity reaching point q from the part of the atmosphere furthest away from the sensor, I_n the intensity at point n,

$$I_q^q = \Big[\sum_{i=1}^{q-1}(\psi_i(1-\zeta_i)\Theta_1^{i-1}\Big]\Theta_1^q + \sum_{i=1}^{q-1}\psi_i(1-\zeta_i)\Theta_{i+1}^q, \qquad q>1 \tag{7.23}$$

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 7.8. 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 7.21 valid for cases with ground reflections. The treatment of ground reflections are discussed seperately last in the section.

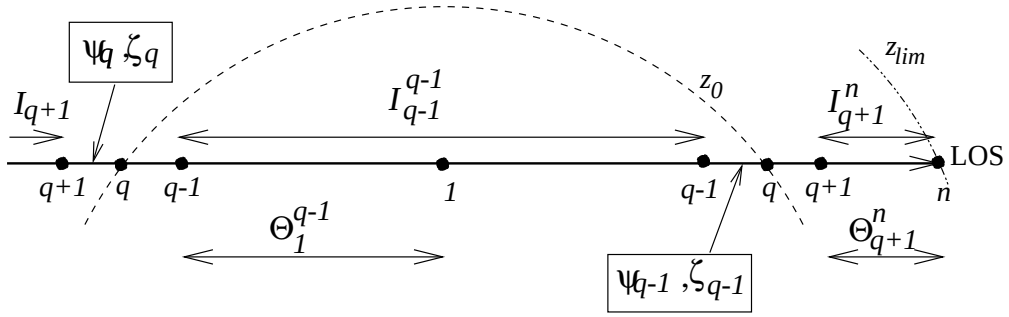

Figure 7.2: The terms used for the derivation of line of sight weighting functions for 1D limb sounding.

If the different combinations of ζ_{q-1} and ζ_q are grouped, for example, Equation 7.21 becomes

$$\begin{aligned} I &= \Big[\Big((I_{q+1}-\psi_q)\zeta_{q-1}^2\zeta_q^2 + (\psi_q-\psi_{q-1})\zeta_{q-1}^2\zeta_q + \psi_{q-1}\zeta_{q-1}\zeta_q\Big)\Big(\Theta_1^{q-1}\Big)^2 + \\ &\quad +(I_{q-1}^{q-1}-\psi_{q-1})\zeta_{q-1}\zeta_q + (\psi_{q-1}-\psi_q)\zeta_q + \psi_q\Big]\Theta_{q+1}^n \end{aligned} \tag{7.24}$$

This equation has some higher products between ζ_{q-1} and ζ_q than Equation 7.6, and the derivatives, with respect to κ_q, of these product are

$$\frac{\partial \zeta_{q-1}^2\zeta_q}{\partial \kappa_q} = -\frac{3\Delta l}{2}\zeta_{q-1}^2\zeta_q \tag{7.25}$$

$$\frac{\partial \zeta_{q-1}^2\zeta_q^2}{\partial \kappa_q} = -2\Delta l\zeta_{q-1}^2\zeta_q^2 \tag{7.26}$$

Using Equations 7.10, 7.12, 7.25 and 7.26, the LOS WFs for 1D limb sounding can be determined to be

$$\begin{aligned} \mathbf{K}_\kappa^q &= -\frac{\Delta l}{2}\Big[\Big(2I_2\zeta_1 + \psi_1(1-2\zeta_1)\Big)\Big(\Theta_1^1\Big)^2 + I_1 - \psi_1\Big]\Theta_1^n \quad q=1 \\ \mathbf{K}_\kappa^q &= -\frac{\Delta l}{2}\Big[\Big(4(I_{q+1}-\psi_q)\zeta_{q-1}\zeta_q + 3(\psi_q-\psi_{q-1})\zeta_{q-1} + 2\psi_{q-1}\Big)\Big(\Theta_1^{q-1}\Big)^2\zeta_{q-1} + \\ &\quad +2(I_{q-1}^{q-1}-\psi_{q-1})\zeta_{q-1} + \psi_{q-1} - \psi_q\Big]\Theta_q^n, \quad 1<q<n \\ \mathbf{K}_\kappa^q &= -\frac{\Delta l}{2}\Big[\Big(2I_n\zeta_{n-1} + \psi_{n-1}(1-2\zeta_{n-1})\Big)\Big(\Theta_1^{n-1}\Big)^2\zeta_{n-1} + \\ &\quad +I_{n-1}^{n-1} - \psi_{n-1}\Big]\zeta_{n-1}, \quad q=n \end{aligned} \tag{7.27}$$

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

$$I_{q-1} = I_q\zeta_{q-1} + \psi_{q-1}(1-\zeta_{q-1})$$

$$\Theta_1^{q-1} = \frac{\Theta_1^q}{\zeta_{q-1}}$$

$$I_{q-1}^{q-1} = \frac{I_q - \psi_{q-1}(1-\zeta_{q-1})(1+(\Theta_1^{q-1})^2\zeta_{q-1})}{\zeta_{q-1}}$$

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.

7.2.3 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

$$\begin{aligned} I &= \Big(I_{q+1}\zeta_q\zeta_{q-1} + \psi_q(1-\zeta_q)\zeta_{q-1} + \psi_{q-1}(1-\zeta_{q-1})\Big)\Big(\Theta_1^{q-1}\Big)^2\zeta_{q-1} + \\ &\quad + I_{q-1}^{q-1}\zeta_{q-1} + \psi_{q-1}(1-\zeta_{q-1}) \end{aligned} \tag{7.28}$$

and the corresponding WF is

$$\begin{aligned} \mathbf{K}_\kappa^q &= -\frac{\Delta l}{2}\Big[\Big(3(I_{q+1}-\psi_q)\zeta_{q-1}\zeta_q + 2(\psi_q-\psi_{q-1})\zeta_{q-1} + \psi_{q-1}\Big)\Big(\Theta_1^{q-1}\Big)^2 + \\ &\quad + I_{q-1}^{q-1} - \psi_{q-1}\Big]\zeta_{q-1} \end{aligned} \tag{7.29}$$

7.3 Source function line of sight weighting functions

The source function line of sight weighting functions are defined as

$$\mathbf{K}_\sigma^q = \frac{\partial \mathbf{i}}{\partial \sigma^q} \tag{7.30}$$

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 7.2.

7.3.1 Single pass

As, for example,

$$\psi_q = \frac{\sigma_q + \sigma_{q+1}}{2} \tag{7.31}$$

the derivate of the mean source function values with respect to σ_q is

$$\frac{\partial \psi_{q-1}}{\partial \sigma_q} = \frac{\partial \psi_q}{\partial \sigma_q} = \frac{1}{2} \tag{7.32}$$

This derivate for other ψ terms is zero.

Using 7.6, the source LOS WFs for upward looking observations can be determined to be

$$\begin{aligned}
\mathbf{K}_\sigma^q &= \frac{1-\zeta_1}{2}, \quad q = 1 \\
\mathbf{K}_\sigma^q &= \frac{1-\zeta_{q-1}\zeta_q}{2}\Theta_1^{q-1}, \quad 1 < q < n \\
\mathbf{K}_\sigma^q &= \frac{1-\zeta_{n-1}}{2}\Theta_1^{n-1}, \quad q = n
\end{aligned} \tag{7.33}$$

For ground points in 2D calculations, the Wfs are (cf. Eq. 7.18)

$$\mathbf{K}_\sigma^q = \frac{(1-\zeta_q)(1-e)\zeta_{q-1} + 1 - \zeta_{q-1}}{2}\Theta_1^{q-1}, \quad 1 < q < n \tag{7.34}$$

The practical calculations, such as the updating of Θ, follow the absorption LOS WFs (Sec. 7.2.1).

7.3.2 1D limb sounding

The 1D limb sounding source LOS WFs are (derived using Eq. 7.21)

$$\begin{aligned}
\mathbf{K}_\sigma^q &= \frac{1}{2}\Big(1-\zeta_1\Big)\Big(1+\Big(\Theta_1^1\Big)^2\zeta_1\Big)\Theta_2^n, \quad q = 1 \\
\mathbf{K}_\sigma^q &= \frac{1}{2}\Big[(1-\zeta_{q-1}\zeta_q)\Big(\Theta_1^{q-1}\Big)^2\zeta_{q-1}\zeta_q + (1-\zeta_{q-1})\zeta_q + \\
&\quad +1-\zeta_q\Big]\Theta_{q+1}^n, \quad 1 < q < n \\
\mathbf{K}_\sigma^q &= \frac{1}{2}\Big((1-\zeta_{n-1})\Big(\Theta_1^{n-1}\Big)^2\zeta_{n-1} + 1 - \zeta_{n-1}\Big), \quad q = n
\end{aligned} \tag{7.35}$$

The practical calculations follow the absorption LOS WFs (Sec. 7.2.2).

7.3.3 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. 7.2.3).

The LOS WF for the index corresponding to the platform altitude is (cf. Eq. 7.28) observations can be determined to be

$$\mathbf{K}_\sigma^q = \frac{1}{2}\Big[(1-\zeta_{q-1}\zeta_q)\Big(\Theta_1^{q-1}\Big)^2\zeta_{q-1} + 1 - \zeta_{q-1}\Big] \tag{7.36}$$

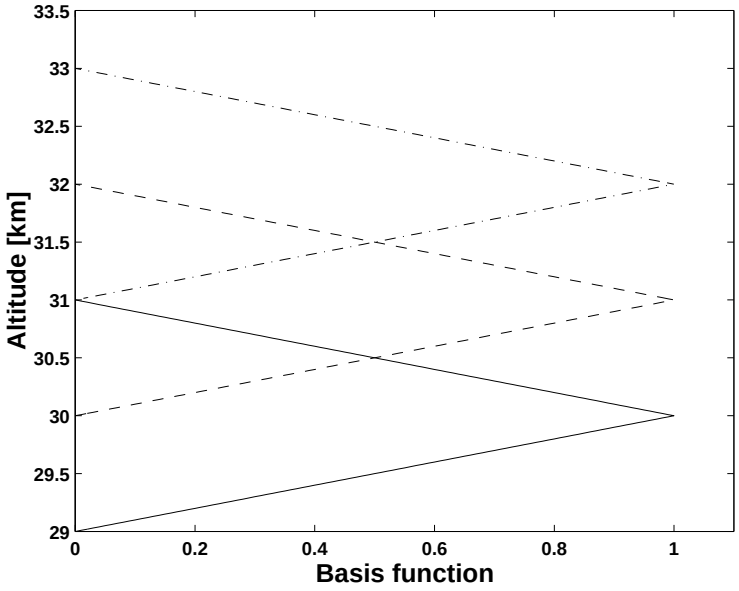

Figure 7.3: Examples on basis functions for a vertical grid with a 1 km spacing: —— 30 km, – – – 31 km and – · – 32 km.

7.4 Transformation from vertical altitudes to distances along LOS

7.4.1 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 picewise 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. 7.3).

7.4.2 Transformation from z to l

The forward model uses internally a grid along the line of sight (Sec. 4), 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 funcional behavoiur (piece wise linear), these two terms are identical if the retrieval grid is the same for both quantities:

$$\frac{\partial\kappa}{\partial\mathbf{k}^p} = \frac{\partial\sigma}{\partial S^{\mathbf{p}}} \tag{7.37}$$

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 7.4 exemplifies $\partial\kappa/\partial\mathbf{k}^p$ for three

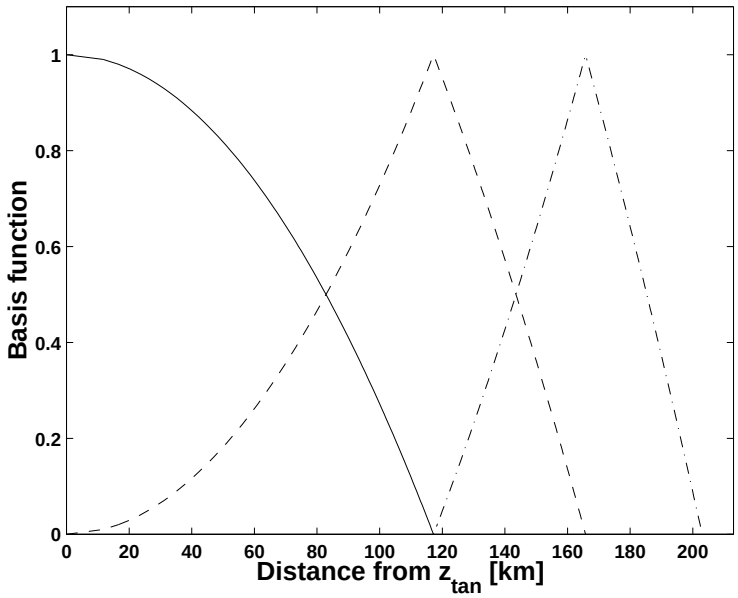

Figure 7.4: The basis functions of Figure 7.3 shown as a function of the distance from the tangent point, where $z_{tan} = 30$ km.

altitudes. Ideally, the following relationship should be fulfilled for all z

$$\sum_i \mathbf{k}^i \phi^i_{\mathbf{k}}(z(l)) = \sum_j \kappa^j \phi^j_\kappa(l) \tag{7.38}$$

where $\phi_{\mathbf{k}}$ and ϕ_κ are the basis functions for $\mathbf{k}$ and κ, respectively. However, as can be seen in Figure 7.4, $\phi^i_{\mathbf{k}}$ 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 7.38 only for the grid points along the LOS:

$$\kappa^q = \sum_i \mathbf{k}^i \phi^i_{\mathbf{k}}(z(\mathbf{l}^q)) \tag{7.39}$$

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

$$\frac{\partial \kappa^q}{\partial \mathbf{k}^p} = \phi^p_{\mathbf{k}}(z(\mathbf{l}^q)) \tag{7.40}$$

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

$$\frac{\partial \kappa^q}{\partial \mathbf{k}^p} = \frac{z_{\kappa^q} - z_{\mathbf{k}^{p-1}}}{z_{\mathbf{k}^p} - z_{\mathbf{k}^{p-1}}} \tag{7.41}$$

If z_{κ^q} is further away from $z_{\mathbf{k}^p}$ than the neighbouring 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.

7.5 Species WFs

As it is assumed here that the species have no influence on the source function, species WFs are calculated as (cf. Eq. 7.4)

$$\mathbf{K}_{\mathbf{x}}^p = \mathbf{H}\frac{\partial\mathbf{i}}{\partial\kappa}\frac{\partial\kappa}{\partial\mathbf{k}^p}\frac{\partial\mathbf{k}^p}{\partial\mathbf{x}^p} \tag{7.42}$$

The term $\partial\mathbf{i}/\partial\kappa$ is described in Section 7.2, while the term $\partial\kappa/\partial\mathbf{k}^p$ is treated in Section 7.4, 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.

The species absorption can be written as

$$\mathbf{k}^p = \bar{\mathbf{k}}_s^p\mathbf{x}^p + \sum_{i\neq s}\mathbf{k}_i^p \tag{7.43}$$

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. We have then that

$$\frac{\partial\mathbf{k}^p}{\partial\mathbf{x}^p} = \bar{\mathbf{k}}_s^p \tag{7.44}$$

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 7.44 is especially simple for this case:

$$\frac{\partial\mathbf{k}^p}{\partial\mathbf{x}^p} = \mathbf{k}_s^p \tag{7.45}$$

as $\mathbf{x}^p = 1$.

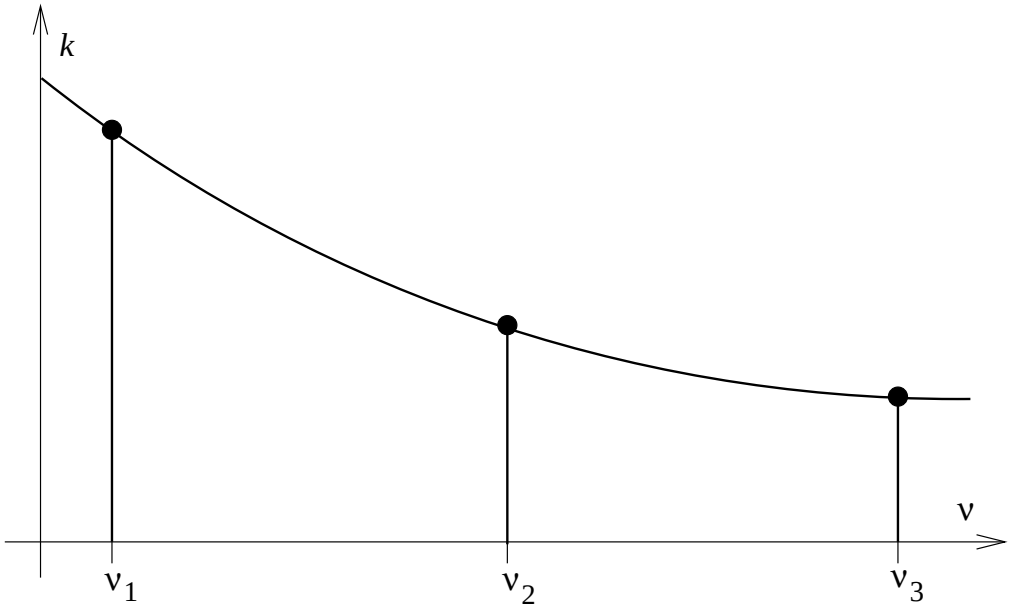

Figure 7.5: Fit of continuum absorption with off-sets at three positions ($n_{cont} = 2$). The outermost frequencies, here ν_1 and ν_3, are placed at the end points of the range covered (ν_{min} and ν_{max}, respectively).

7.6 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:

$$\mathbf{k}^p = \mathbf{k}_s^p + \mathbf{k}_c^p \tag{7.46}$$

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. 7.5). 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 7.5 exemplifies this for $n_{cont} = 2$. The points are accordingly placed at the following frequencies

$$\nu_i = \nu_{min} + \frac{(\nu_{max} - \nu_{min})(i-1)}{n_{cont}}, \quad 1 \le i \le (n_{cont} + 1) \tag{7.47}$$

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

$$\frac{\partial \mathbf{k}^p}{\partial \mathbf{x}_1^p} = 1 \tag{7.48}$$

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]:

$$\begin{aligned} k(\nu) &= \frac{(\nu-\nu_2)(\nu-\nu_3)\dots(\nu-\nu_N)}{(\nu_1-\nu_2)(\nu_1-\nu_3)\dots(\nu_1-\nu_N)}x_1 + \\ &+\frac{(\nu-\nu_1)(\nu-\nu_3)\dots(\nu-\nu_N)}{(\nu_2-\nu_1)(\nu_2-\nu_3)\dots(\nu_2-\nu_N)}x_2+\dots+ \\ &+\frac{(\nu-\nu_1)(\nu-\nu_2)\dots(\nu-\nu_{N-1})}{(\nu_N-\nu_1)(\nu_N-\nu_2)\dots(\nu_N-\nu_{N-1})}x_N \end{aligned} \tag{7.49}$$

where x_i is the absorption at the selected frequency points, ν_i, that are given by Equation 7.47, and $N = n_{cont}+1$.

The frequency dependency of the continuum WFs can be obtained by differentiating Equation 7.49:

$$\frac{\partial \mathbf{k}^p(\nu)}{\partial \mathbf{x}_i^p} = \frac{(\nu-\nu_1)\dots(\nu-\nu_{i-1})(\nu-\nu_{i+1})\dots(\nu-\nu_N)}{(\nu_i-\nu_1)\dots(\nu_i-\nu_{i-1})(\nu_i-\nu_{i+1})\dots(\nu_i-\nu_N)} \tag{7.50}$$

This equation gives, for example, for $n_{cont}=1$

$$\frac{\partial \mathbf{k}^p(\nu)}{\partial \mathbf{x}_1^p} = \frac{\nu_{max}-\nu}{\nu_{max}-\nu_{min}}, \quad \nu_{min}\le\nu\le\nu_{max} \tag{7.51}$$

$$\frac{\partial \mathbf{k}^p(\nu)}{\partial \mathbf{x}_2^p} = \frac{\nu-\nu_{min}}{\nu_{max}-\nu_{min}}, \quad \nu_{min}\le\nu\le\nu_{max} \tag{7.52}$$

Note that these WFs have no altitude variation. Or with other words, they are identical for all p.

7.7 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.

A change of the temperature inside an atmospheric layer will change the line of sights for beams passing this altitude. This effect should however normally be small, and it is here neglected.

7.7.1 Without hydrostatic equilibrium

For some measurement situations it can be questionable to assume that the pressure, temparature 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 7.4, that is:

$$\mathbf{K}_{\mathbf{x}}^{p} = \mathbf{H}\left[\frac{\partial \mathbf{i}}{\partial \sigma}\frac{\partial \sigma}{\partial \mathbf{S}^{p}}\frac{\partial \mathbf{S}^{p}}{\partial \mathbf{t}^{p}} + \frac{\partial \mathbf{i}}{\partial \kappa}\frac{\partial \kappa}{\partial \mathbf{k}^{p}}\frac{\partial \mathbf{k}^{p}}{\partial \mathbf{t}^{p}}\right] \tag{7.53}$$

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 7.3, while the absorption LOS WFs ($\partial \mathbf{i}/\partial \kappa$) are found in Section 7.2. As a single grid is here of concern, Equation 7.37 is valid, that is, $\partial \kappa/\partial \mathbf{k}^{p}$ equals $\partial \sigma/\partial S^{\mathbf{p}}$. These two terms are discussed in Section 7.4.

Here it is assumed that S equals the Planck function, B (Equation 3.2), and the derivative of the source function with respect to the temperature is (see also Equation 44 of *Eriksson et al.* [2000])

$$\frac{\partial S}{\partial T} = \frac{h\nu}{k_B T^2}\left(1 - e^{-h\nu/k_B T}\right)^{-1} B(\nu, T) \tag{7.54}$$

The term $\partial \mathbf{S}^{p}/\partial \mathbf{t}^{p}$ is calculated using Equation 7.54 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.

7.7.2 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. The pressure in the atmosphere changes as

$$\Delta P = -\rho g \Delta z \tag{7.55}$$

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:

$$z_2 - z_1 = \frac{R_d \bar{T}_v}{g} ln\left(\frac{P_1}{P_2}\right) \tag{7.56}$$

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

$$T_v = T\left(1 + 0.379\frac{x_{H2O}}{1 - x_{H2O}}\right) \tag{7.57}$$

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

The calculations take into account that the gravitational acceleration and the average molecular weight changes with altitude. ...(To be written!!)

The temperature WFs with hydrostatic equilibrium are basically calculated by perturbations (Eq. 7.1). The temperature at each pressure level is changed 1 K. When considering hydrostatic equilibrium, the ground pressure is kept constant, i.e. the vertical altitudes of the pressure levels below the point of concern are not changed. (Finish after implementation!! Smart tricks as to calculate the absorption for +1K (effect of vertical changes?)?)

7.8 WF for ground emission factor

This WF is not yet implemented but this can easily be done.

Chapter 8

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.

8.1 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

$$\mathbf{K}_{\mathbf{x}}^p = \frac{\partial \mathbf{y}}{\partial \mathbf{x}^p} = \frac{\partial \varepsilon}{\partial \mathbf{x}^p} = \mathbf{H}_d\frac{\partial \varepsilon'}{\partial \mathbf{x}^p} \tag{8.1}$$

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.

8.2 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. Instead of providing weighting functions, the forward model gives the covariance matrix for the thermal noise.

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.

8.2.1 Measurement thermal noise

As mentioned above, measurement thermal noise is here defined to be totally uncorrelated between the different viewing angles. The magnitude of the thermal noise, expressed in brightness temperatures, is described by the radiometer noise formula

$$\sigma_{tn}^{i} = \frac{q\left(T_{rec} + T_{a}^{i}\right)}{\sqrt{\Delta\nu^{i}\tau}} \tag{8.2}$$

where σ_{tn}^{i} is the standard deviation of the thermal noise for channel i, q a compensation factor T_{rec} the receiver noise temperature, T_a the antenna temperature, $\Delta\nu$ the channel bandwidth and τ the integration time.

The factor q is used to compensate for extra noise introduced by the calibration, losses in the spectrometer etc. It is important to define q and τ consistently. Let us take an ordinary load switching instrument as example, where one half of the time is used to measure the atmosphere and the other half is used to observe a reference load. If then τ gives the total integration time, q should be (about) a factor $\sqrt{2}$ higher than when τ gives only the integration time for the atmospheric observations.

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

$$\mathbf{S}_{tn}^{ii} = \left(\sigma_{tn}^{i}\right)^{2} \tag{8.3}$$

where $\mathbf{S}^{ii}$ is element (i, i) of the matrix.

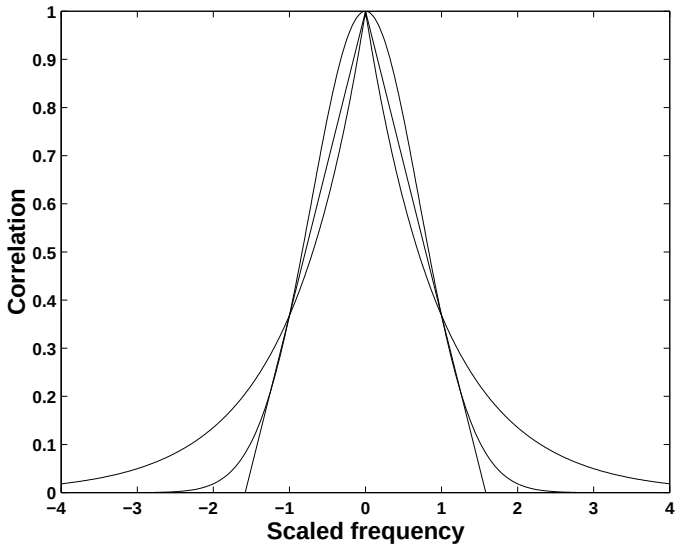

Figure 8.1: The frequency correlation functions. The frequency is scaled to the correlation length as $(\nu_i - \nu_j)/\nu_c$.

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 treated in the forward model by three different correlation functions: (1) gaussian

$$c^{ij} = exp\left(-\left(\frac{\nu_i - \nu_j}{f_c}\right)^2\right) \tag{8.4}$$

(2) exponential

$$c^{ij} = exp\left(-\frac{|\nu_i - \nu_j|}{f_c}\right) \tag{8.5}$$

and (3) tenth

$$\begin{aligned} c^{ij} &= 1 - \frac{|\nu_i - \nu_j|(1 - e^{-1})}{\nu_c}, \quad |\nu_i - \nu_j| < \frac{\nu_c}{(1 - e^{-1})} \\ c^{ij} &= 0, \quad |\nu_i - \nu_j| \geq \frac{\nu_c}{(1 - e^{-1})} \end{aligned} \tag{8.6}$$

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. 8.1). 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

$$\mathbf{S}_{tn}^{ij} = c^{ij}\sigma_{tn}^{i}\sigma_{tn}^{j} \tag{8.7}$$

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

$$\mathbf{S}_{tn} = \mathbf{H}_d\mathbf{S}'_{tn}\mathbf{H}_d^T \tag{8.8}$$

where $\mathbf{S}'_{tn}$ is the covariance matrix before data reduction.

8.2.2 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 and a simplified expression is used:

$$\sigma_{tn}^{i} = \frac{T_{cal}}{\sqrt{\Delta\nu^{i}\tau_{cal}}} \tag{8.9}$$

where T_{cal} is an effective noise temperature covering all relevant effects and τ_{cal} the calibration integration time.

The correlation functions used for the measurement thermal noise can also be applied for the calibration thermal noise.

Data reduction is considered by Equation 8.8.

8.3 Sinusoidal baseline ripple

Reflections inside the receiver give theoretically rise to a sinusoidal baseline ripple. The relationship between the period length in the spectrum, $\Delta\nu_{2\pi}$, and the physical distance between the reflecting objects, l, is [*Rohlfs*, 1986]

$$\Delta\nu_{2\pi} = \frac{c}{2l} \tag{8.10}$$

where c is the speed of light.

This type of baseline ripple is retrieved by expressing the sine functions, with unknown amplitude and phase, as a sum of sine and cosine functions [*Kuntz et al.*, 1997]

$$\varepsilon_{sin} = \sum_{i=1}^{n}\left(x_i sin\left(2\pi\frac{\nu-\bar{\nu}}{\Delta\nu_{2\pi}^{i}}\right) + x_{i+n} cos\left(2\pi\frac{\nu-\bar{\nu}}{\Delta\nu_{2\pi}^{i}}\right)\right) \tag{8.11}$$

where n is the number of ripple terms, $\bar{\nu}$ the mean frequency, $\nu_{2\pi}^{i}$ the period length of ripple i and x_i are the amplitude of the sine and cosine functions to be determined. The length of the part of $\mathbf{x}$ used to fit sinusoidal baseline ripples is accordingly $2n$. The mean frequency is defined below by Equation 8.19.

Using Equation 8.1, the WFs for the sine and cosine terms can be determined to be

$$\mathbf{K}_{\mathbf{x}}^{p} = \mathbf{H}_d\mathbf{a}_p \tag{8.12}$$

and

$$\mathbf{K}_{\mathbf{x}}^{p} = \mathbf{H}_d\mathbf{b}_p \tag{8.13}$$

respectively, where the elements of the vectors $\mathbf{a}_p$ and $\mathbf{b}_p$ are

$$\mathbf{a}_p^{i} = sin\left(2\pi\frac{\nu^{i}-\bar{\nu}}{\Delta\nu_{2\pi}^{p}}\right) \tag{8.14}$$

and

$$\mathbf{b}_p^{i} = cos\left(2\pi\frac{\nu^{i}-\bar{\nu}}{\Delta\nu_{2\pi}^{p}}\right), \tag{8.15}$$

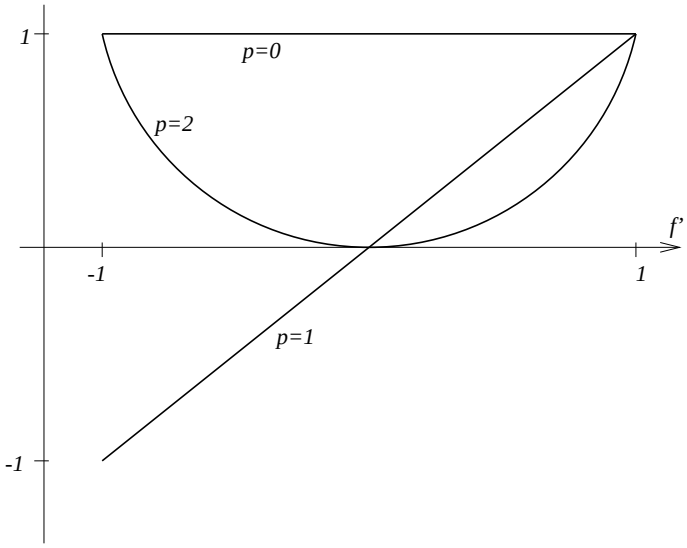

Figure 8.2: Polynomial WFs of order 0, 1 and 2. The scaled frequency is $f' = (\nu - \bar{\nu})/\Delta\nu$.

where ν^i is the frequency for channel i.

It should be noted that the treatment of baseline ripple neglects the effect of the spectrometer and Equation 8.11 assumes that the widths of the spectrometer channels are much smaller than the period length of the ripple. However, this should be the situation found for most practical situations.

8.4 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 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

$$\varepsilon_{pol} = x_0 + \sum_{i=1}^{n_{pol}} x_i \left(\frac{\nu - \bar{\nu}}{\Delta\nu} \right)^i \tag{8.16}$$

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

$$\mathbf{x} = \begin{bmatrix} \vdots \\ x_0 \\ x_1 \\ \vdots \\ x_{n_{pol}} \\ \vdots \end{bmatrix} \tag{8.17}$$

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

$$\bar{\nu} = \frac{\nu_{min} + \nu_{max}}{2} \tag{8.18}$$

$$\Delta\nu = \frac{\nu_{max} - \nu_{min}}{2} \tag{8.19}$$

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

$$\mathbf{K}_{\mathbf{x}}^{p} = \mathbf{H}_d \mathbf{a}_p \tag{8.20}$$

where the elements of $\mathbf{a}_p$ are

$$\mathbf{a}_p^i = \left(\frac{\nu^i - \bar{\nu}}{\Delta\nu} \right)^p \tag{8.21}$$

Note that for $p = 0$, $\mathbf{a}_p = 1$.

Examples on polynomial weighting functions are shown is Figure 8.2.

8.5 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 8.3 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 8.21, while for remaining frequencies the WFs are 0.

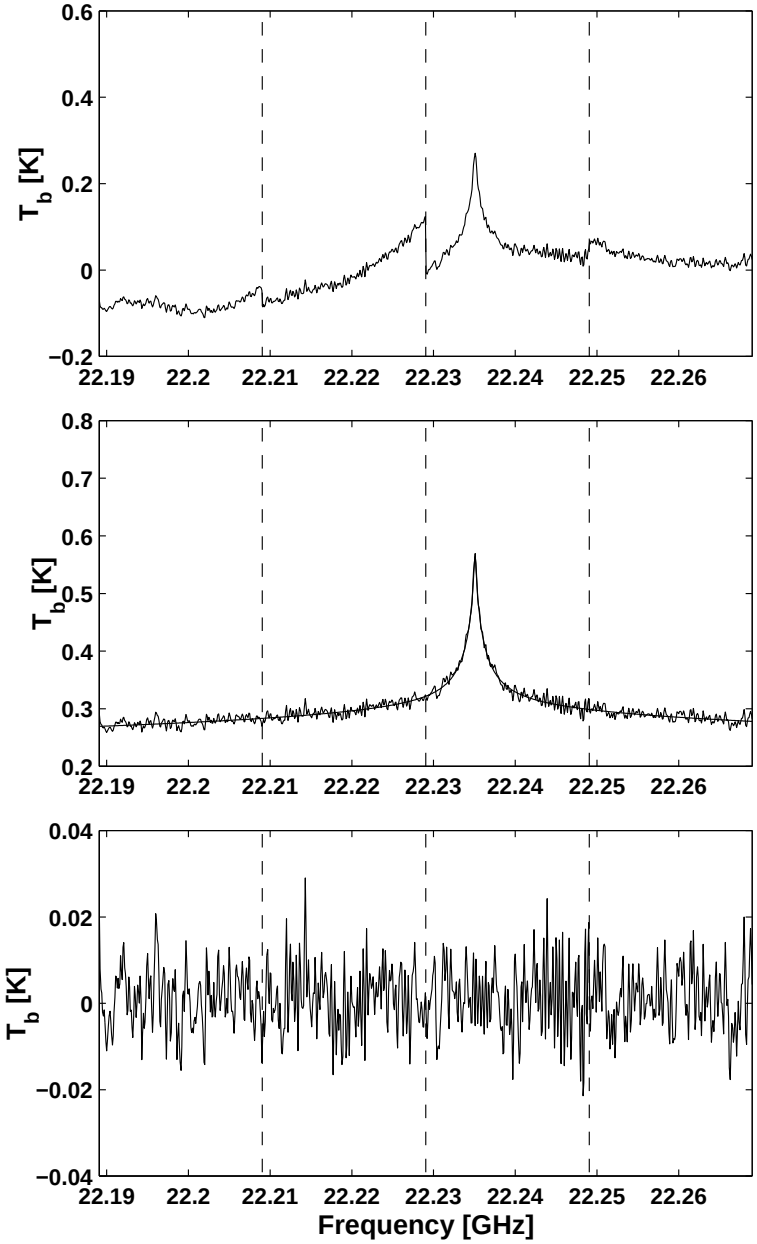

Figure 8.3: Example on fit of baseline with piecewise polynomials. The top figure shows a (poor!) test measurement with the 22.2 GHz water vapor radiometer at Onsala Space Observatory, Sweden. The spectrum was recorded by an auto-correlator spectrometer having four 20 MHz wide individual parts, clearly 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.

Chapter 9

Sensor variables weighting functions

This section presents weighting functions for sensor variables, beside the ones treated as measurement errors. The covered features are calibration, pointing and frequency instability.

9.1 Calibration weighting functions

9.1.1 Proportional calibration errors

This section gives the WF for situations where a calibration uncertainty gives an error that is directly proportinal to the noise free spectrum. Such a calibration uncertainty can be encountered for e.g. ground-based observations of altitudes above the tropoapuse, where a compensation of the tropospheric attenuation must be made, as an error of the assumed tropospheric opacity gives rise to a proportional calibration error.

A measurement with a proportional calibration uncertainty can be expressed as

$$\mathbf{y} = \mathbf{H}_d \left((1 + x_{cal}) \mathbf{H}_s \mathbf{i} + \varepsilon' \right) \tag{9.1}$$

See Equation 2.11 for definition of the variables. The WF for this case is easily obtained

$$\mathbf{K_x} = \mathbf{H}_d \mathbf{H}_s \mathbf{i} = \mathbf{H} \mathbf{i} = \mathbf{y} - \varepsilon \tag{9.2}$$

that is, the WF is identical to the (noise free) spectrum given by the forward model.

9.1.2 Calibration load temperatures

The calibration of a Dicke switched radiometer is often performed by observing two loads with known intensity. The calibration formula is then (neglecting data reduction)

$$\mathbf{y}^i = I_1^i + (I_2^i - I_1^i) \frac{V_{atm}^i - V_1^i}{V_2^i - V_1^i} \tag{9.3}$$

History

000320 Created and written by Patrick Eriksson.

Here	In ARTS	Description
–	–	–

Table 9.1: Symbols used in this chapter and the corresponding ARTS notation.

where $\mathbf{y}^i$ is the calibrated value for channel i, I_1 and I_2 are the assumed intensities of the two loads, V_{atm}, V_1 and V_2 are the voltage recorded when observing the atmosphere, load 1 and load 2, respectively.

The load temperature WFs are obatined by differenting Equation 9.3. For example, we have that [*Eriksson and Merino*, 1997]

$$\frac{\partial \mathbf{y}^i}{\partial I_1^i} = 1 - \frac{V_{atm}^i - V_1^i}{V_2^i - V_1^i} = \frac{I_2^i - \mathbf{y}^i}{I_2^i - I_1^i} \tag{9.4}$$

The WF for load temperature 1 is then

$$\mathbf{K_x} = \mathbf{H}_d \mathbf{a} \tag{9.5}$$

where the elements of the vector $\mathbf{a}$ are

$$\mathbf{a}^i = \frac{I_2^i - \mathbf{y}^i}{I_2^i - I_1^i} \tag{9.6}$$

The corresponding expression for load 2 is

$$\mathbf{a}^i = \frac{\mathbf{y}^i - I_1^i}{I_2^i - I_1^i} \tag{9.7}$$

Hence, these WFs are easily calculated if the spectrum (before data reduction) is at hand.

Chapter 10

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.

10.1 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 `data` for auxiliary data (such as model atmospheres). 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 `arts/INSTALL` for installation instructions. We are developing with the GNU C++ compiler, no other compilers have been tried so far. FIXME: Update this also.

History	
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`.

10.2 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 10.5.4. 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 I had to use some tricks to make sure that the Doxygen automatic documentation and the User Guide work as they should. I've set it up so that these documents need not to be built by the installer of the program, since he or she might not have the necessary programs (Doxygen and LaTeX). However, this could (and hopefully will) be done much more nicely in the future. For example, there should be an automatik check for Doxygen and LaTeX, with apropriate actions taken depending on if the programs are found or not.

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 if the things you added are really there.

10.3 Conventions

Here are some general rules for ARTS programming:

10.3.1

Never use `float` or `double` explicitly, use the type `Numeric` instead. This is set in `arts.h` (to `double` by default). Thus, it is possible to compile the program for `float` by simply changing the `typedef` in `arts.h`.

10.3.2

Use `VECTOR` and `MATRIX` for mathematical vectors and matrices (with elements of type `Numeric`). Use `ARRAY<string>` for example to create an array of strings (and likewise for any other type). This should work for everything except `bool` (dunno why not for `bool`, strange things happen). ARRAYs can be used just like VECTORs. In particular, you can use round braces to get 1-based indexing, and they do range checks.

10.3.3 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.

10.3.4 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.

10.3.5 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.)

10.3.6 Version numbers

The package version number is set in file `configure.in` in the top level ARTS directory. Always increase this when you make a new distribution. The minor version number is set in `src/version.cc`. Always increase this before you do a CVS commit, even for small changes.

10.3.7 Global header file

The global header file `arts.h` *must* be included by every file, for example because it turns on or off assertions (see also Section 10.7).

10.3.8 Documentation

Doxygen is used to generate automatic 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.** If you make changes to a function or file, add some descriptive text and another `\author` `\date` block (see example below).

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

   Made some modifications to illustrate how histories should be handled.
   \author Stefan Buehler
   \date   2000-09-14
*/
int dummy(int& a, int b);
```

Generic comment:

```
/**
   This is a dummy comment. You can write as much as you want here...
*/
```

10.4 Extending ARTS

10.4.1 How to add a workspace variable

1. Create a record entry in file `workspace.cc`. (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.

2. That's it!

10.4.2 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)

10.4.3 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! It should contain line breaks and even double line breaks (= blank lines) in appropriate places.

   For the future, maybe also an `author` field would be nice here. (FIXME: Update this.)

2. Run: `make`.

3. Look in `md.h`. There is a new function prototype
   `void <YourNewMethod>(...)`
   Check that everything looks nice. If necessary, change the documentation string.

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. Thats it!

10.4.4 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. Unfortunately, at the moment you also have to register your file in a similar way in the file `doc/doxygen/Makefile.am`, if the Doxygen documentation should work.

4. Then go to the top level arts directory and run: `reconf`.

5. In the same directory, run: `configure`. This will create new makefiles which take your new file into account.

6. Go to `src` and run: `cvs add <my_file>` to make your file known to CVS.

10.5 CVS issues

The arts project is controlled by CVS. This section describes some basic CVS commands. For more information see the extensive CVS documentation.

10.5.1 How to check out arts

1. Go to a temporary directory.

2. Run: `cvs co -P arts`.

10.5.2 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.

10.5.3 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: `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 `src/version.cc`.

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.

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. (Guess we should set up a mailing list. FIXME: Update this.)

10.5.4 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`.)

2. Commit your changes (see other howto). However, the following is different now:

   - Set the subversion number in file `src/version.cc` to 0.

3. 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.

4. 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.

5. Check the release carefully by trying to build and install the program.

10.5.5 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).

10.6 Configuration

Here are some interesting options for `configure`:

–disable-warnings: Compile without `-Wall` on g++ compilers (by default warnings are on).

–disable-assert: Include `#define NDEBUG 1` in `config.h`. This is (will be FIXME: Implement this) the central switch to turn off all debugging features (index range checking for vectors, the trace facility, assertions,...) **Not yet implemented.**

10.7 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/index.html`.

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.

In ARTS, assertions are turned on and off with the global NDEBUG preprocessor macro, which can be set or unset in file `arts.h`. In the future there will be also a configure option to achieve this (FIXME: Update this).

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 TNT (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.

Bibliography

Balluch, M., and D. Lary, Refraction and atmospheric photochemistry, *J. of Geophys. Res.*, *102*, 8845–8854, 1997.

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.

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

Kyle, T., *Atmospheric transmission, emission and scaterring*, Pergamon Press, 1991.

Press, W., S. Teukolsky, W. Vetterling, and B. Flannery, *Numerical recipes in FORTRAN*, 2nd ed., Cambridge University Press, 1992.

Rodgers, C., Characterization and error analysis of profiles retrieved from remote sounding measurements, *J. of Geophys. Res.*, *95*, 5587–5595, 1990.

Rohlfs, K., *Tools of radio astronomy*, Springer-Verlag, Berlin, 1986.

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.