# Qpack2 – a Matlab tool for performing atmospheric retrievals of OEM type

Patrick Eriksson
(patrick.eriksson@chalmers.se)

August 17, 2020

## 1 Scope

Qpack2 is a retrieval system, implemented in Matlab, for performing inversions of atmospheric observations inside the framework of "optimal estimation" (OEM). With good knowledge of the background systems a wide range of retrievals can be handled, but the primary aim is to provide relatively straightforward retrievals of vertical profiles of atmospheric quantities. That is, the atmosphere is treated to be of "1D type" during the retrieval. Accordingly, a main application areas of Qpack2 is ground-based observations.

The package should be general and flexible for covered measurements. For example, there is basically no limitations regarding observation geometry. Each measurement can consist of several spectra, obtained by some scanning procedure or different instruments. See Sec. 8 for a discussion of possibilities and limitations when it comes to handle more complex measurements. Further, Qpack2 has been developed to be suitable for operational inversions, as long as not most extreme calculation speed is required. Some important aspects here are that batch calculations are allowed and atmospheric a priori profiles can automatically be extracted from climatology data.

## 2 Background and introduction

Qpack2 is part of the Atmlab package of Matlab functions. In fact, it is largely a merge of some of the systems in Atmlab. The computational engine, i.e. the forward model, of Qpack2 is ARTS-2 (*Eriksson et al.*, 2011). ARTS is a C++ program, downloaded separately (i.e. not part of Atmlab). The communication with ARTS-2 is made through a systen denoted as Qarts. OEM inversions are performed by the function `oem.m`. Climatology data are stored and interpolated through a data format called "atmdata".

This can be seen as a direct successor of Qpack (*Eriksson et al.*, 2005), despite all code is written from scratch. For both Qpack versions the computations are controlled by a set of settings fields packed into a structure denoted as Q (though Qpack2 has also a structure O). Qpack was built around ARTS-1, this new Qpack version is mainly a consequence of that ARTS-2 is the maintained version of ARTS. (ARTS-2 is below just denoted as ARTS.) This gave also an opportunity to make a more stringent implementation and to add some features. This should hopefully make it easier to maintain and extend Qpack2. Data formats are now more clearly defined and the ambition level around documentation is higher (but still modest). A main improvement compared to Qpack is that extraction of a priori data from climatology data is now an integrated part. On the other hand,

the feature of classifying errors into different categories (0-3 in Qpack) and the associated plotting features were removed as these features are dificult to implement in a generic way.

Quality checks are not handled by Qpack2. For example, there is no default check that the frequencies of the measurements are as expected. This for efficiency reasons and the fact that such checks are hard to make in a totally general manner. Checks are left for accompanying dedicated functions, such as `qp2_check_f`.

# 3    Software and installation

The needed software packages are ARTS and Atmlab. The simplest is to obtain and update these packages through svn. Download instructions for ARTS are found at `www.sat.ltu.se/arts/getarts`. The installation details are here not repeated. A plain installation of ARTS suffices.

Download options for Atmlab are found at `www.sat.ltu.se/arts/tools`. The file CONFIGURE (found in the top folder) gives instructions for how to get started with Atmlab. Qarts2 demands that several "atmlab settings" are activated. This is handled by the function `atmlab`. A description of all atmlab settings is obtained by:

```
>> help atmlab
```

At least the following atmlab settings are used: `ARTS_PATH`, `ARTS_INCLUDES`, `VERBOSITY`, `FMODEL_VERBOSITY`, `STRICT_ASSERT` and `WORK_AREA`. Some example settings:

```
atmlab( 'ARTS_PATH',         fullfile(homedir,'ARTS/arts/src/arts') );
atmlab( 'ARTS_INCLUDES',     fullfile(homedir,'ARTS/arts/includes') );
atmlab( 'FMODEL_VERBOSITY',  0                                      );
atmlab( 'VERBOSITY',         1                                      );
atmlab( 'STRICT_ASSERT',     true                                   );
atmlab( 'WORK_AREA',         '/tmp'                                 );
```

The standard choice is to place these calls of `atmlab` in `atmlab_conf.m`, as described in CONFIGURE.

# 4    Documentation

The overall documentation is found in this document. The detailed information is mainly stored in the implementation files. Information on individual functions is obtained by the standard Matlab help command. For example:

```
>> help oem
```

Some of the used data structures are documented matching the requirements of `qinfo.m`. For example, to list all fields and the documentation text of Qarts' `Q`:

```
>> qinfo(@qarts)
```

The documentation for a specific field is obtained as

```
>> qinfo(@qarts,'F_BACKEND')
```

Wildcards are allowed when defining fields:

```
>> qinfo(@qarts,'ABS_*')
```

Details around development and bug fixes are described in the ChangeLog file of Atmlab (`atmlab/ChangeLog`).

# 5  Overview

## 5.1  Practicalities

A retrieval by Qpack2 has three main steps:

```
% 1: Define forward model and retrieval settings
[Q,O] = my_q_fun;

% 2: Import the measurement data to be inverted
Y = my_y_fun;

% 3: Perform the inversion
L2 = qpack2( Q, O, Y );
```

The variables `Q`, `O`, `Y` and `L2` are all structures, and the fields of these structures are described in Section 6.

The main part of the settings defined in step 1 are found in `Q`, that match directly the `Q` of the Qarts interface to ARTS. All forward model variables are part of `Q`, and retrieval variables such as grids and a priori data are also defined in `Q`. The exceptions are settings directly associated with OEM, that are stored in `O`. This division reflects the fact that the OEM operations are made by a stand-alone function. This function, `oem`, is implemented in a general manner, to be applicable for any retrieval case as long as an interface to a "forward model" is provided.

The measurement data structure `Y` can be an array. That is, a series of observations can be inverted in a single call of `qpack2`, on the condition that a `Q` and `O` are valid for all observations. This implies e.g. that basic sensor characteristics such as backend frequencies must be common for all the data cases in `Y`. On the other hand, both observation position and direction can differ between the observations, and the data from e.g. an aircraft flight could be inverted in a single run.

The usage of the functions `my_q_fun` and `my_y_fun` in steps 1 and 2 is just a suggestion for how to organise the definition phase of the run, but it is probably a good idea to separate the inversion settings and the task of importing measurement data.

When `Q`, `O` and `Y` are created, it is just to call `qpack2`, and everything should work automatically. The input of `qpack2` is fixed (throughout `Q`, `O` and `Y`). The output is normally the result of the retrieval, packed as a "level 2" data structure (`L2`). The function can also provide simulated measurements. This option is triggered by setting the spectrum field to be empty (if `Y` is an array, this must be done for all elements):

```
...
Ye.Y = [];
Ysim = qpack2( Q, O, Ye );
```

The output is a copy of `Ye`, with the field `Ysim.Y` set to the result of the forward model run. The simulated data match the a priori state of the retrieval set-up.

Several full examples are found in Atmlab's folder `demos`, such as `qpack2_demo.m`. The examples work with simulated data, created by `qpack2` itself. The `qpack2_demo.m` example file is also appended to this document as Appendix A. A related example is `arts_oem_demo`. It does not use Qpack2, but shows how an OEM inversion is performed with ARTS as the forward model, and the calculation steps in this example gives a good view of the interior of Qpack2.

## 5.2 Units

In most cases SI units are used. For example, frequency off-sets are retrieved in Hz. For retrieval of species profiles several "units" can be used. First of all, both volume mixing ratio and number density retrievals can be made. These options are selected by setting the species unit to `vmr` and `nd`, respectively.

Further, the species unit can be also be set to `rel`. In this case, the retrieved profile is expressed in fractions of the a priori profile. For example, the a priori state corresponds to 1, and $x = 2$ signifies that the species amount is double as high as the a priori. The remaining species unit is `logrel`, that is defined as the natural logarithm of the `rel` unit. The main reason for using this unit is to introduce a constrain of positive species values. For example, if z is a retrieved `logrel` value and $v_a$ is the a priori VMR, the retrieved volume mixing ratio is $v_a \exp(z)$, which always is greater than 0.

The prescribed or selected unit is used for all related variables. For example, a priori covariance matrices must be specified correspondingly. For example, if the selected species unit is `rel`, the 1 standard deviation a priori uncertainty should be in the order of 0.5 (that corresponds to a 50% uncertainty). The same applies to output data. No automatic unit conversions are performed (in contrast to the earlier Qpack version), e.g. the returned a priori profile for `logrel` retrievals is a vector of zeros. The basic idea is to return data that are fully consistent with the assumptions of the retrieval.

Data retrieved using the `rel` and `logrel` units can be converted to volume mixing ratio data by the function `qp2_rel2vmr`. However, this function handles only data directly associated with the species, it does not cover, for example, the Jacobian matrix. See further the help text of the function.

# 6 Main data structures

## 6.1 Forward model and retrieval settings

### 6.1.1 General features

Forward model parameters and most retrieval variables are joined into a single structure, denoted as `Q`. This structure has a fixed set of fields. That is, the structure must always contain exactly this set of fields. However, some fields are used only in particularly circumstances and not all fields must be set. A field of `Q` is flagged as undefined by setting it to {}, and e.g. [] is taken as an active selection. This applies also to structures `O` and `Y`.

The structure `Q` of Qpack2 is identical to the `Q` of Qarts. Qpack2 makes only slight modifications of `Q` before it is given to Qarts. The Qarts system is primarily an interface to the ARTS forward model, on the same time as retrieval variables can be defined in a relatively general manner. However, Qarts is not a complete retrieval system, it just allows definition of retrieval variables in parallel with the forward model data. The ambition is just to provide the basis for retrieval systems, such as Qpack2 or streamlined software dedicated to a single sensor.

Qarts is documented through the qinfo feature:

```
>> qinfo(@qarts)
```

The call of `qinfo` above provides a description of all defined fields (Sec. 3) and this information is also found here as Appendix B. It is recommended to initialise `Q` as

```
Q = qarts;
```

This ensures that Q contains all defined fields, also ones introduced more lately.

All features of ARTS can be accessed through Qarts. Calculations of standard type can largely be handled by defining the fields of Qarts that have a direct matching ARTS workspace variables (WSV). That is, the name is the same in Qarts and ARTS, beside that Qarts use uppercase for its fields and ARTS lowercase for its variables (e.g. `Z_SURFACE` and `z_surface`, respectively). Qarts has also several fields that allow inclusion of workspace method (WSM) calls. Some of these fields match ARTS agendas and have then a dedicated purpose (e.g. `SURFACE_RTPROP_AGENDA`). There are also fields where you are free to include any set of WSM calls (e.g. `WSMS_BEFORE_RTE`). You can also use ARTS's include feature through the field `INCLUDES`.

Accordingly, Qarts is a relatively clean and open interface to ARTS, and to set-up a calculation demands primarily an understanding of ARTS. Built-in documentation of ARTS WSMs and WSVs is obtained in a terminal as

```
$ arts -d z_surface
```

or from within Matlab as

```
>> arts('-d z_surface');
```

See the documentation of ARTS for more in-depth documentation.

The best introduction to the practical usage of ARTS is obtained through the control file examples found in ARTS's folder `controlfiles`. For an introduction of the usage of ARTS through Qarts, there are several example scripts in Atmlab's folder `demos`, such as `qarts_demo`.

### 6.1.2 Qpack2 specifics

Most fields of `Q` can be unspecified (`{}`), but Qpack2 handles some fields in special way:

- `Q.RAW_ATMOSPHERE` must be unset (`{}`).

- All existing "atmdata"-fields are considered and are given highest priority. That is, if such a field is set, any setting of the corresponding field will be overwritten. For example, if `Q.T.ATMDATA` is set, any existing values in `T_FIELD` will have no effect.

- The fields `T_FIELD`, `VMR_FIELD` and `Z_FIELD` must be defined directly in `Q`, possibly through the corresponding atmdata fields (see point above). That is, these workspace variables can not be set by an include file.

- In the case of `Z_FIELD` there is another option, this field can be set by applying hydrostatic equilibrium, then activated by setting `Q.HSE.ON` to true. The reference point for hydrostatic equilibrium is specified by `Y.HSE_P` and `Y.HSE_Z` (Sec. 6.3). If `Q.HSE.ON` is true, the derived `Z_FIELD` has highest priority, and any settings in the `Z_FIELD` and `Z.ATMDATA` have no effect.

- `Q.HSE_P` is set to `Y.HSE_P`. Other fields of `Q.HSE` must be set by the user, thus including an active choice for `Q.HSE.ON`.

- `Q.ABSORPTION` must be set. The option 'CalcTable' is not allowed (as it is difficult to ensure generation of good absorption tables in a general manner). You must either use 'OnTheFly' or create an absorption table as a pre-calculation (the 'LoadTable' option).

- `Q.SENSOR_POS` and `Q.SENSOR_LOS` are set based on data in the `Y` structure. User specified data in these fields have no impact.

- `Q.TNOISE_C` must be set.

- The a priori knowledge covariance matrix (Sx) is created by `arts_sx` and the `SX` sub-fields of the retrieval quantities must be set.

## 6.2 OEM settings

Settings directly associated with OEM are put into the structure `O`. Also this setting structure is documented through `qinfo` (see also Appendix C):

```
>> qinfo(@oem)
```

Many of the fields of `O` controls the output of the `oem` function. For Qpack2 you do not need to set these fields manually. The output required to provide the specified L2 data (Section 6.4) is ensured by initialising `O` as

```
O = qp2_l2( Q );
```

Some fields of O are also set by Qpack2, and user settings are overwritten. These fields are:

- `O.msg`

- `O.sxnorm`

For linear inversion, a single active setting is needed:

```
O.linear = true;
```

## 6.3 Measurement data

The series of measurements to be treated is packed into a data structure denoted as `Y`. The data fields of this structure are described in `qp2_y`, following the `qinfo` format:

```
>> qinfo(@qp2_y)
```

The same information is appended to this document in Appendix D. An example:

```
>> Y(3)

ans =

          DAY: 3
            F: [800x1 double]
         HOUR: 11
        HSE_P: 10000
        HSE_Z: 1.5677e+04
     LATITUDE: 57.4000
    LONGITUDE: 11.9300
       MINUTE: 57
        MONTH: 3
       SECOND: 58.0000
```

```
        TNOISE: 0.0243
             Y: [800x1 double]
          YEAR: 2002
            ZA: 65
    Z_PLATFORM: 5
```

The simplest way to initialise Y is to use `qp2_y` as

```
for i = 1 : n
  Y(i) = qp2_y;
  load( file{i} )
  Y(i).Y = ...
end
```

## 6.4   Retrieved data

Retrieved data are returned as a structure array L2. The fields of L2 are not fixed, the set of returned fields depend on settings in Q. This makes it impossible to use qinfo here (as for Q and Y), and the fields are instead described below in this document.

The only mandatory field of L2 is `converged`. The remaining content of L2 is controlled by the Qarts field `L2_EXTRA` and the L2 sub-field for the retrieval quantities (e.g. `Q.POLYFIT.L2`). As an example on a complete L2, the first element of the output of `qpack2_demo` is (the polyfit part is truncated):

```
>> L2(1)

ans =

          year: 2008
         month: 2
           day: 25
          hour: {}
        minute: {}
        second: {}
     converged: 1
            dx: 4.4374e-06
          cost: 1.0748
        cost_x: 5.8390e-04
        cost_y: 1.0742
             f: [1279x1 double]
             y: [1279x1 double]
            yf: [1279x1 double]
            bl: [1279x1 double]
        p_grid: [45x1 double]
       t_field: [45x1 double]
       z_field: [45x1 double]
 species1_name: 'O3'
    species1_p: [45x1 double]
   species1_xa: [45x1 double]
    species1_x: [45x1 double]
    species1_e: [45x1 double]
```

```
    species1_eo: [45x1 double]
    species1_es: [45x1 double]
    species1_mr: [45x1 double]
        ffit_xa: 0
         ffit_x: -3.8104e+03
         ffit_e: 8.7798e+03
        ffit_eo: 8.6418e+03
        ffit_es: 1.5504e+03
        ffit_mr: 0.9692
    polyfit0_xa: 0
     polyfit0_x: 0.0516
     polyfit0_e: 0.4748
    polyfit0_eo: 0.0967
    polyfit0_es: 0.4649
    polyfit0_mr: 0.7745
    polyfit1_xa: 0
     polyfit1_x: -0.0022
     polyfit1_e: 0.0097
    polyfit1_eo: 0.0094
    polyfit1_es: 0.0023
    polyfit1_mr: 0.9996
        ...
```

A description of the options for `Q.L2_EXTRA` is obtained by:

```
>> help qp2_l2
```

Further comments and description of fields directly associated with specific retrieval quantities are found below (no ambition of completeness here).

### 6.4.1  Various general fields

`cost`, `cost_x` and `cost_y`:
Cost values of retrieved state and fit to measurement. See further `oem.m`. All obtained by including the string 'cost' in `L2_EXTRA`.

`P.year`, `P.month`, `P.day`, `P.hour`, `P.minute`, and `P.second`:
The time of the measurement. All obtained by including the string 'date' in `L2_EXTRA`.

`tnoise`:
Mean of assumed thermal noise. This is simple average, no weighting with channel widths is applied. This value has no direct physical meaning, but can be used as a mean to e.g. remove the most noisy measurements (without having `Y` at hand). Obtained by including the string 'tnoise' in `L2_EXTRA`.

`f`:
Frequency vector of measurement. Equals `Y.F` if set. Otherwise set `Q.SENSOR_RESPONSE_F` or `Q.F_GRID`, depending on if a sensor is applied or not. Only included together with `y`.

`y`:
Measured spectrum. Obtained by including the string 'y' in `L2_EXTRA`.

`yf`:
Fitted spectrum. That is, the spectrum matching retrieved state vector. Obtained by including the string 'yf' in `L2_EXTRA`.

`bl`:
Retrieved "baseline". That is, retrieved spectrum distortion through e.g. polynomial fitting. A scalar 0 if no baseline retrieval is performed. Obtained by including the string 'bl' in `L2_EXTRA`.

`converged`:
Convergence flag. See further `oem.m`. Always included.

`dx`:
Change in the state vector x between each iteration. See further `oem.m`. Obtained by including the string 'dx' in `L2_EXTRA`.

`J`, `G` and `A`:
The (full) Jacobian, gain and averaging kernel matrix, respectively. Obtained by including the string 'J', 'G' and 'Afull', respectively, in `L2_EXTRA`. Note that the string ' A' only triggers inclusion of partial averaging kernels (see below).

`S`, `So` and `Ss`:
The covariance matrix of total, observation and smoothing error, respectively. Obtained by including the string 'S', 'So' and 'Ss', respectively, in `L2_EXTRA`.

`jq` and `ji`:
Information describing the different retrieval quantities. The structure `jq` is basically matching the user settings. One example:

```
>> L2(1).jq{1}

ans =


        maintag: 'Absorption species'
         subtag: 'O3'
           mode: 'vmr'
     analytical: 1
    perturbation: 0
          grids: {[48x1 double]}
```

The second structure, `ji`, gives the start and end index in the state vector for each retrieval quantity. The data are stored in a manner matching the data type used in ARTS for this information, an ArrayOfIndex. For example, if retrieval quantity two holds position 49 in 96 in the state vector, we have that

```
>> L2(1).ji{2}

ans =

   [49]    [96]
```

These two structures are included automatically when `A`, `S` or any data matching the complete state vector is included in the output, and, hence, the structures are needed match vector and matrix elements to the different retrieval quantities.


### 6.4.2  Background atmospheric state

`p_grid`, (`lat_grid`, `lon_grid`), `t_field` and `z_field`:
As the ARTS workspace variables with the same name. The data are identical to the ones

that are provided to the forward model. (If temperatures are retrieved, this information is used. Otherwise returned data match climatology temperatures.) Obtained by including the string 'ptz' in `L2_EXTRA`.

### 6.4.3 Retrieved species

The retrieved species are indexed after the order they are specified in `Q` and named as `species1`, `species2` ... A number of fields is returned for each species. The field names below are valid for species 1. If two species are retrieved there is also a field named as `species2_name` etc. The unit of output follows `Q.SPECIES(i).UNIT`, see further Section 5.2. These set of fields are available:

`species1_name:`
Tag name of species.

`species1_p:`
Vertical/pressure retrieval grid for the species.

`species1_x:`
Retrieved profile. Unit depends on `Q.SPECIES(i).UNIT`.

`species1_xa:`
A priori state for the species. Unit depends on `Q.SPECIES(i).UNIT`. Obtained by including the string 'xa' in `L2_EXTRA`.

`species1_e:`
Total retrieval (observation + smoothing) error for the species. Unit depends on `Q.SPECIES(i).UNIT`. Obtained by including the string 'e' in `L2_EXTRA`.

`species1_eo:`
Observation error for the species. Unit depends on `Q.SPECIES(i).UNIT`. Obtained by including the string 'eo' in `L2_EXTRA`.

`species1_es:`
Smoothing error for the species. Unit depends on `Q.SPECIES(i).UNIT`. Obtained by including the string 'es' in `L2_EXTRA`.

`species1_mr:`
Measurement response for the species. This is the sum of the rows of `species_A` (see below). Obtained by including the string 'mresp' in `L2_EXTRA`.

`species1_A:`
The species specific averaging kernels. That is, this is the diagonal centred sub-matrix covering only the resolution between changes of the species at different altitudes. The resolution between variables of different retrieval quantities is thus not covered. Obtained by including the string 'A' in `L2_EXTRA`.

`species1_vmr0:`
Added automatically if the retrieval unit is `rel` or `logrel`, to be used to convert data to VMR. For example, the retrieved profile in VMR for the `rel` case is `x.*vmr0`.

### 6.4.4 Baseline fit variables

There are two approaches for performing baseline fits, denoted as `POLYFIT` and `SINEFIT`. The two approaches can be used seperately, or be combined.

A set of fields is added for each polynomial coefficient, and for the first polynomial coefficient they are

`P.polyfit0_x`, `P.polyfit0_xa`, `P.polyfit0_e`, `P.polyfit0_eo`, `P.polyfit0_es` and
`P.polyfit0_mr`:
As corresponding fields above for species (`species1_x` etc.).

Fields are added in a similar manner for `SINEFIT`. For the first selected ripple period the
fields are

`P.sinefit1_x`, `P.sinefit1_xa`, `P.sinefit1_e`, `P.sinefit1_eo`, `P.sinefit1_es` and
`P.sinefit1_mr`:
These fields have length 2, where the two values correspond to the cosine and sine terms,
respectively.

### 6.4.5 Frequency fit variables

Both a shift and a stretch factor can be retrieved:

`P.fshift_x`, `P.fshift_xa`, `P.fshift_e`, `P.fshift_eo`, `P.fshift_es` and
`P.fshift_mr`:
As corresponding fields above for species (`species1_x` etc.).

`P.fstretch_x`, `P.fstretch_xa`, `P.fstretch_e`, `P.fstretch_eo`, `P.fstretch_es` and
`P.fstretch_mr`:
As corresponding fields above for species (`species1_x` etc.).

# 7 Data types and file formats

## 7.1 Qpack2 variables

The data types in Qpack2 (and in Qarts) follow largely ARTS. Regarding basic data types,
there are two main things to consider. Firstly, a distinction is made between the logical
0 and 1, and the integers 0 and 1. Accordingly, booleans, sometime also denoted as flags,
must be set to `true` or `false`. Secondly, vectors are demanded to be columns.

A notable feature of Qpack2 and Qarts is that data fields, booleans and scalars excluded, can either be set directly or be set to the name of a file (including data of the
expected type). Both these settings are possible

```
Q.F_GRID = [ 501.18e9 : 1e6 : 501.58e9 ]';
Q.F_GRID = 'f_odinsmr_ac2a.xml;
```

Note the transpose in the first case (to create a column vector).

The fields `Q.ABS_SPECIES.ATMDATA`, `Q.T.ATMDATA` and `Q.Z_ATMDATA` are intended for
data of climatology character. The data format used here is called atmdata. The format
is similar to the GriddedField format in ARTS, but has also fields for e.g. units. The
atmdata format is based on the general gformat:

```
>> help gformat
```

For a description of the specialities of atmdata:

```
>> help isatmdata
```

For example, the MSIS temperature climatology found in arts-xml-data is stored as GriddedField4 but imported as

```
Q.T_ATMDATA = gf_artsxml( fullfile( arts_xmldata_path, 'climatology', ...
                          'msis90', 'msis90.t.xml' ), 'Temperature', 't_field' );
```

This produces data of atmdata type:

```
>> Q.T_ATMDATA

ans =

         TYPE: 'atmdata'
         NAME: 'Temperature'
       SOURCE: [1x65 char]
          DIM: 4
         DATA: [4-D double]
    DATA_NAME: 'Temperature'
    DATA_UNIT: 'K'
        GRID1: [141x1 double]
   GRID1_NAME: 'Pressure'
   GRID1_UNIT: 'Pa'
        GRID2: [19x1 double]
   GRID2_NAME: 'Latitude'
   GRID2_UNIT: 'deg'
        GRID3: 0
   GRID3_NAME: 'Longitude'
   GRID3_UNIT: 'deg'
        GRID4: [13x1 double]
   GRID4_NAME: 'doy'
   GRID4_UNIT: ''
```

## 7.2  File formats

The standard format for data input files is "ARTS xml". These files are hardly created by hand, but stored from Matlab (through `xmlStore`) or from ARTS (through `WriteXML`). If ARTS is compiled with support for NetCDF, that format can also be used (`arts_nc_write_`*datatype* and `WriteNetCDF`, respectively).

# 8  Measurements consisting of multiple spectra

As long as each measurement consists of a single spectrum, Qpack2 should be quite flexible in treating sensor aspects. For some conditions, it also possible to handle measurements consisting of multiple spectra.

## 8.1  Common sensor characteristics

If the sensor characteristics are common for all spectra involved (not including thermal noise) it is relatively straightforward to apply Qpack. For example, a sensor that is moved vertically, or rotated to measure at different zenith angles, can be treated. Requirements and considerations to handle such multiple-spectra measurements include:

- The fields `Y.Z_PLATFORM` and `Y.ZA` shall give the altitude and zenith angle of each spectrum, and are accordingly (column) vectors instead of scalar values for single-spectra cases.

- If `Y.LAT_PLATFORM` is used, this must be a vector of same length as `Y.ZA`. Note that `Y.LATITUDE`, as well as `Y.LONGITUDE`, is always scalar values, and can not be used to descibe a horizontally moving sensor.

- `Y.T_NOISE` must be extended where each column describes the noise of the corresponding spectrum. Hence, the noise level can vary between the spectra.

- If a baseline fit is performed, the corresponding `SX`-fields must be extended. These fields are scalar values for single-spectrum observations. With multiple spectra, the covariance matrices can include an expected correlation between the baseline coefficient for different spectra. For example, if the zero-order `POLYFIT` coefficient is expected to have a standard deviation of 2 K and has a correlation of 0.9 between two spectra, these assumptions are included as

  ```
  Q.POLYFIT.SX0 = 2.0^2 * [ 1 0.9; 0.9 1 ];
  ```

It is also possible to describe the scanning in a 3D atmosphere, that should be of interest if the azimuth angle matters. This option should be of interest for cases including a Zeeman influence or wind retrievals. Additional considerations are then:

- `Q.ATMOSPHERE_DIM` shall be set to 3.

- `Q.LAT_GRID` and `Q.LON_GRID` must be specified.

- Atmospheric fields, such as `Q.T_FIELD`, must be 3D. The standard choice should be to set the fields to be constant in latitude and longitude directions. Using the `ATMDATA` feature with an input 1D atmosphere handles this automatically.

- The field `Y.AA` shall be set, and match `Y.ZA` in length.

- If `Y.LON_PLATFORM` is used, this vector must match `Y.ZA` in length.

- Latitude and longitude retrieval grids, such in `Q.ABS_SPECIES.GRIDS`, must be specified, but must have length one. Any value should work. One example:

  ```
  Q.ABS_SPECIES(1).GRIDS = { Q.P_GRID, mean(Q.LAT_GRID), mean(Q.LON_GRID)};
  ```

  Retrieved atmospheric data will effectively have no latitude and longitude dimension and be of 1D type.

The file `qpack2_wind3d_demo` exemplifies retrievals of this type.

## 8.2 More complex set-ups

It can be considered to append spectra from different instruments to perform a joint retrieval. In a few cases this can be handled as in the sub-section above, but would normally require a more complex set-up. Another more complex situation is when the sensor characteristics change during e.g. a limb scanning sequence. Both these examples can be handled by ARTS by making use of the `yCalcAppend` workspace method.

Using Qpack together with `yCalcAppend` is not impossible but requires detailed knowledge of how both Qpack and ARTS work and far from all possible measurement scenarios can be treated. Hence, it is best to contact Patrick for advice before start trying to implement a set-up of this type.

# References

Eriksson, P., C. Jiménez, and S. A. Buehler, Qpack, a general tool for instrument simulation and retrieval work, *J. Quant. Spectrosc. Radiat. Transfer*, 91, 47-64, 2005.

Eriksson, P., S. A. Buehler and C. P. Davis and C. Emde and O. Lemke, ARTS, the atmospheric radiative transfer simulator, Version 2, *J. Quant. Spectrosc. Radiat. Transfer*, 112, 1551-1558, 2011.

# Appendices

# A  An example script

```
% QPACK2_DEMO   Demonstration of the Qpack2 retrieval system
%
%    The main features of Qpack2 are demonstrated. The example case is airborne
%    measurements of ozone at 110.8 GHz. Synthetic measurement data are
%    generated internally. See the code and internal comments for details.
%
%    Everything is here put into a single file. For practical retrievals it
%    is probably better to put the definitions of Q (together with O?) in a
%    separate function (i.e. [Q,O] = q_mycase). A function to import
%    measurement data into the "Y format" (see *qp2_y*) is needed. The
%    retrieval result is returned in the L2 format produced by *qp2_l2*.
%
%    This script focuses on giving an introduction, indicating different
%    retrieval units and retrieval of other variables. See also *qpack2_demo2*.
%
% FORMAT   L2 = qpack2_demo
%
% OUT      L2   L2 data output from *qpack2*.

% 2010-05-12   Created by Patrick Eriksson.

function L2 = qpack2_demo

errid = ['atmlab:' mfilename];

%- Qarts settings
%
Q    = q_demo;              % Local file, found below



%- Measurement data
%
Y = y_demo( Q );           % Local file, found below



%- Check that all frequencies are OK
%
if ~qp2_check_f( Q, Y, 1e3 );
  error( errid, ...
           'Some mismatch between Q.F_BACKEND and frequencies of spectra.' );
end



%- OEM variables
%
O = qp2_l2( Q );           % This ensures that OEM returns the variables needed
%                            to fill the L2 structure, as defined in Q
```

```
O.linear = false;
%
if ~O.linear
  O.itermethod = 'GN';
  O.stop_dx    = 0.01;
  O.maxiter    = 5;
end



%- Make inversion
%
L2 = qpack2( Q, O, Y );



%- Plot, if no output argument
%
if ~nargout

  % Profiles
  figure(1),clf
  plot( L2(1).species1_x*1e6, p2z_simple(L2(1).species1_p)/1e3, 'b', ...
        L2(2).species1_x*1e6, p2z_simple(L2(2).species1_p)/1e3, 'r', ...
        L2(1).species1_xa*1e6, p2z_simple(L2(1).species1_p)/1e3, 'k-' );
  xlabel( 'Ozone [VMR]' );
  ylabel( 'Approximate altitude [km]' );
  axis( [ 0 8 10 90 ] )
  legend( 'Retrieval 1', 'Retrieval 2', 'True and a priori' );

  % Spectra
  figure(2),clf
  h=plot( L2(1).f/1e9, L2(1).y, 'k.', L2(2).f/1e9, L2(2).y, 'k.', ...
          L2(1).f/1e9, L2(1).yf, 'b-', L2(2).f/1e9, L2(2).yf, 'r-', ...
          L2(1).f/1e9, L2(1).bl, 'b-.', L2(2).f/1e9, L2(2).bl, 'r-.' );
  xlabel( 'Frequency [GHz]' );
  ylabel( 'Tb [K]' );
  axis( [ min(L2(1).f/1e9) max(L2(1).f/1e9) 0 18 ] )
  legend( h(3:end), 'Fitted 1', 'Fitted 2', 'Baseline 1', 'Baseline 2' );
end

return
%-----------------------------------------------------------------------------
%-----------------------------------------------------------------------------
%-----------------------------------------------------------------------------




function Q = q_demo
```

```
errid = ['atmlab:' mfilename];
%- Atmlab settings
%
arts_xmldata_path = atmlab( 'ARTS_XMLDATA_PATH' );
arts_includes     = atmlab( 'ARTS_INCLUDES' );
if isnan( arts_xmldata_path )
  error( errid,'You need to set ARTS_XMLDATA_PATH to run this exmaple.' );
end
if isnan( arts_includes )
  error( erird,'You need to ARTS_INCLUDES to run this example.' );
end
%
fascod = fullfile( arts_xmldata_path, 'planets', 'Earth', 'Fascod' );


%- Init Q
%
Q = qarts;
%
Q.INCLUDES             = { fullfile( 'ARTS_INCLUDES', 'general.arts' ), ...
                           fullfile( 'ARTS_INCLUDES', 'agendas.arts' ), ...
                           fullfile( 'ARTS_INCLUDES', 'continua.arts' ), ...
                           fullfile( 'ARTS_INCLUDES', 'planet_earth.arts' ) };
Q.ATMOSPHERE_DIM       = 1;
Q.STOKES_DIM           = 1;
Q.J_DO                 = true;
Q.CLOUDBOX_DO          = false;


%= Define agendas
%
% Here we do it by using the predefined agenda templates
%    (found in arts/controlfiles/general/agendas.arts)
% This works only if the pre-defined agenda is names following the pattern:
%    name_of_agenda__(Something)
%
Q.PPATH_AGENDA              = { 'ppath_agenda__FollowSensorLosPath'   };
Q.PPATH_STEP_AGENDA         = { 'ppath_step_agenda__GeometricPath'    };
Q.IY_SPACE_AGENDA           = { 'iy_space_agenda__CosmicBackground'   };
Q.IY_SURFACE_AGENDA         = { 'iy_surface_agenda__UseSurfaceRtprop' };
Q.IY_MAIN_AGENDA            = { 'iy_main_agenda__Emission'            };


%- Radiative transfer
%
Q.IY_UNIT          = 'RJBT';
Q.YCALC_WSMS       = { 'yCalc' };
%
```

```
Q.PPATH_LMAX            = 250;


%- Surface
%
Q.Z_SURFACE             = 1e3;            % Just a dummy value. A 10 km
                                          % observation altitude is assumed here


%- Absorption
%
Q.ABS_LINES             = fullfile( atmlab_example_data, 'o3line111ghz' );
Q.ABS_LINES_FORMAT      = 'ARTSCAT';
%
Q.ABSORPTION            = 'OnTheFly';
Q.ABS_NLS               = [];

%- Pressure grid
%
z_toa                   = 95e3;
%
Q.P_GRID                = z2p_simple( Q.Z_SURFACE-1e3 : 2e3 : z_toa )';



%- Frequency, spectrometer and pencil beam antenna
%
% The hypothetical spectrometer has rectangular response functions
%
Q.F_GRID                = qarts_get( fullfile( atmlab_example_data , ...
                                              'f_grid_111ghz.xml' ) );
%
H                       = qartsSensor;
%
H.SENSOR_NORM           = true;
%
H.BACKEND_DO            = true;
df                      = 0.5e6;
H.F_BACKEND             = ( min(Q.F_GRID)+df : df : max(Q.F_GRID)-df )';
%
B.name                  = 'Spectrometer channel response function';
B.gridnames             = { 'Frequency' };
B.grids                 = { [-df/2 df/2] };
B.dataname              = 'Response';
B.data                  = [1 1];
%
H.BACKEND_CHANNEL_RESPONSE{1} = B;
clear B
%
```

```
Q.SENSOR_DO          = true;
Q.SENSOR_RESPONSE    = H;
%
Q.ANTENNA_DIM        = 1;
Q.MBLOCK_DLOS_GRID   = 0;




%- Correlation of thermal noise
%
f               = H.F_BACKEND;
cl              = 1.4 * ( f(2) - f(1) );
cfun            = 'gau';
cco             = 0.05;
%
Q.TNOISE_C      = covmat1d_from_cfun( f, [], cfun, cl, cco );
%
clear H f



%- Define L2 structure (beside retrieval quantities below)
%
Q.L2_EXTRA      = { 'cost', 'dx', 'xa', 'y', 'yf', 'bl', 'ptz', ...
                    'mresp', 'A', 'e', 'eo', 'es', 'date' };



%- Temperature
%
Q.T.RETRIEVE    = false;
Q.T.ATMDATA     = gf_artsxml( fullfile( arts_xmldata_path, ...
                      'planets', 'Earth', 'MSIS90', 'climatology', ...
                      'msis90.t.xml' ), 'Temperature', 't_field' );

%- Determine altitudes through HSE
%
Q.HSE.ON        = true;
Q.HSE.P         = Q.P_GRID(1);
Q.HSE.ACCURACY = 0.1;



%- Species

% Ozone, only species is retrieved here
Q.ABS_SPECIES(1).TAG      = { 'O3' };
Q.ABS_SPECIES(1).RETRIEVE = true;
Q.ABS_SPECIES(1).L2       = true;
Q.ABS_SPECIES(1).GRIDS    = { Q.P_GRID, [], [] };
Q.ABS_SPECIES(1).ATMDATA = gf_artsxml( fullfile( fascod, ...
      'midlatitude-winter', 'midlatitude-winter.O3.xml' ), 'O3', 'vmr_field' );
```

```
%
% If you don't apply a min value (by MINMAX), you could need to active this:
%Q.VMR_NEGATIVE_OK = true;
%
% For demonstration, setting for several units are provided:
switch 1
 case 1  % Constant VMR
  Q.ABS_SPECIES(1).UNIT     = 'vmr';
  Q.ABS_SPECIES(1).SX       = ...
                     covmat1d_from_cfun( Q.ABS_SPECIES(1).GRIDS{1}, 1.5e-6, ...
                                         'lin', 0.2, 0.00, @log10 ) + ...
                     covmat1d_from_cfun( Q.ABS_SPECIES(1).GRIDS{1}, 0.3e-6, ...
                                         'lin', 0.5, 0.00, @log10 );
   Q.ABS_SPECIES(1).MINMAX  = 1e-12;
 case 2 % Constant rel
  Q.ABS_SPECIES(1).UNIT     = 'rel';
  Q.ABS_SPECIES(1).SX       = ...
                     covmat1d_from_cfun( Q.ABS_SPECIES(1).GRIDS{1}, 0.5, ...
                                         'lin', 0.2, 0.00, @log10 ) + ...
                     covmat1d_from_cfun( Q.ABS_SPECIES(1).GRIDS{1}, 0.1, ...
                                         'lin', 0.5, 0.00, @log10 );
  Q.ABS_SPECIES(1).MINMAX  = 1e-6;
 case 3 % Mimic case 2 in vmr
  Q.ABS_SPECIES(1).UNIT     = 'vmr';
  Q.ABS_SPECIES(1).SX       = ...
                     covmat1d_from_cfun( Q.ABS_SPECIES(1).GRIDS{1}, ...
                                    [ Q.ABS_SPECIES(1).ATMDATA.GRID1,...
                                 0.5 * Q.ABS_SPECIES(1).ATMDATA.DATA ],...
                                         'lin', 0.2, 0.00, @log10 ) + ...
                     covmat1d_from_cfun( Q.ABS_SPECIES(1).GRIDS{1}, ...
                                    [ Q.ABS_SPECIES(1).ATMDATA.GRID1, ...
                                 0.1 * Q.ABS_SPECIES(1).ATMDATA.DATA ],...
                                         'lin', 0.5, 0.00, @log10 );
  Q.ABS_SPECIES(1).MINMAX  = 1e-12;
 case 4 % Constant logrel
  Q.ABS_SPECIES(1).UNIT     = 'logrel';
  Q.ABS_SPECIES(1).SX       = ...
                     covmat1d_from_cfun( Q.ABS_SPECIES(1).GRIDS{1}, 0.5, ...
                                         'lin', 0.2, 0.00, @log10 ) + ...
                     covmat1d_from_cfun( Q.ABS_SPECIES(1).GRIDS{1}, 0.1, ...
                                         'lin', 0.5, 0.00, @log10 );
  Q.ABS_SPECIES(1).MINMAX  = 1e-6;
end



%- Water
%
% This generates no absorption, as linefile has no H2O lines
```

```
%
Q.ABS_SPECIES(2).TAG       = { 'H2O' };
Q.ABS_SPECIES(2).RETRIEVE = false;
Q.ABS_SPECIES(2).ATMDATA  = gf_artsxml( fullfile( fascod, ...
      'midlatitude-winter', 'midlatitude-winter.H2O.xml' ), 'H2O', 'vmr_field' );




%- Wind
%
% Here demonstrated for v-component, that should the component of main
% concern for ground-based instruments. This component can be seen as the
% part of the horizontal wind going along the viewing direction, where
% positive values mean a movement away from the sensor.
%
Q.WIND_V_FIELD    = [];       % This is a short-cut for zero wind.
%
Q.WIND_V.RETRIEVE = false;    % Set to true to activate retrieval
Q.WIND_V.L2       = true;
Q.WIND_V.GRIDS    = { Q.P_GRID(1:2:end), [], [] };
Q.WIND_V.SX       = covmat1d_from_cfun( Q.WIND_V(1).GRIDS{1}, 40, ...
                                         'lin', 0.5, 0.00, @log10 );
if Q.WIND_V.RETRIEVE
  Q.WSMS_AT_START{end+1} = 'IndexSet(abs_f_interp_order,3)';
end




%- Pointing
%
% Here just included for testing purposes, of little interest for ground-based
% spectrometers.
%
Q.POINTING.RETRIEVE        = false;
Q.POINTING.DZA             = 0.1;
Q.POINTING.POLY_ORDER      = 0;
Q.POINTING.CALCMODE        = 'recalc';
Q.POINTING.SX              = 1;
Q.POINTING.L2              = true;




%- Frequency shift
%
Q.FSHIFTFIT.RETRIEVE       = true;
Q.FSHIFTFIT.DF             = 25e3;
Q.FSHIFTFIT.SX             = 50e3^2;
Q.FSHIFTFIT.L2             = true;


%- Frequency stretch
```

```
%
Q.FSTRETCHFIT.RETRIEVE    = false;   % Set to true to activate retrieval
Q.FSTRETCHFIT.DF          = 25e3;
Q.FSTRETCHFIT.SX          = 50e3^2;
Q.FSTRETCHFIT.L2          = true;


%- Polyfit
%
% A polynomial of order 3 is used for "baseline fit".
%
Q.POLYFIT.RETRIEVE        = true;
Q.POLYFIT.ORDER           = 3;
Q.POLYFIT.L2              = true;
Q.POLYFIT.SX0             = 1^2;
Q.POLYFIT.SX1             = 0.5^2;
Q.POLYFIT.SX2             = 0.2^2;
Q.POLYFIT.SX3             = 0.1^2;


%- Sinefit
%
% Here demonstrated with two period lengths
%
Q.SINEFIT.RETRIEVE        = false;   % Set to true to activate retrieval
Q.SINEFIT.PERIODS         = [ 75e3 200e3 ]';
Q.SINEFIT.L2              = true;
Q.SINEFIT.SX1             = 0.2^2;
Q.SINEFIT.SX2             = 0.4^2;

return
%------------------------------------------------------------------------------
%------------------------------------------------------------------------------
%------------------------------------------------------------------------------




function Y = y_demo(Q)

% The data should be loaded from one or several files, but are here genereted
% by a forward model call to show how qpack2 can also be used to generate
% simulaled measurements (matching a priori assumptions).

% The simulated data model airborn measurements at two different zenith
% angles, from two nearby positions.

% Init Y
```

```
%
Y = qp2_y;

% Set a date
%
Y.YEAR  = 2008;
Y.MONTH = 2;
Y.DAY   = 25;

% Lat / lon
%
Y.LATITUDE  = 45;
Y.LONGITUDE = exp(1);

% An airborn measurement assumed here
%
Y.Z_PLATFORM = 10.5e3;
Y.ZA         = 50;

% Reference point for hydrostatic equilibrium
%
Y.HSE_P = 100e2;
Y.HSE_Z = 16e3;

% Set backend frequencies
%
Y.F = Q.SENSOR_RESPONSE.F_BACKEND;

% Thermal noise standard deviation
%
Y.TNOISE = 0.1;
% To test varying noise
%Y.TNOISE = linspace( 0.03, 0.07, length(Y.F) )';

% Simulate a measurement
%
Y.Y = [];                           % A flag to tell qpack2 to calculate the
%                                     spectrum ({} signifies undefined!).

% Add a second measurement
%
Y(2) = Y(1);
%
Y(2).LONGITUDE = pi;
Y(2).ZA        = 45;

% Add frequency error to measurement 2
%
df = 100e3;
```

```
Y(2).F            = Y(2).F + df;


% Calculate simulated spectra
%
Y = qpack2( Q, oem, Y );              % Dummy oem structure OK here

% Move back to expected frequencies in Y(2)
%
Y(2).F = Y(2).F - df;

% Add thermal noise
%
% The correlation specified in Q is included
%
for i = 1 : length(Y)
  Y(i).Y = Y(i).Y + Y(i).TNOISE .* make_noise(1,Q.TNOISE_C);
end

% Add a constant "baseline shift" for measurement 2
%
Y(2).Y = Y(2).Y + 1;
```

# B   The fields of Q

The information below is obtained in Atmlab by typing:

```
>> qinfo( @qarts )
```

ABS_LINES:
The data on spectroscopic lines to use. See *ABS_LINES_FORMAT* for
how to handle cases without lines. Can be given in two ways:
1. As a file name. All formats handled by arts can then be used.
See ABS_LINES_FORMAT. The reading is not restricted to any
frequency range.
2. For the XML option, several file names can be given (as string
array) and the line data in the files are appended.
3. As an array of line data. This option is only allowed when
ABS_LINES_FORMAT is set to 'XML'.

ABS_LINES_FLIMS:
Frequency limits when reading from input spectroscopic file.
   When *ABS_LINES_FORMAT* is set and is not equal to 'None', this
field matches the arguments *fmin* and *fmax* in ARTS reading
routines for *abs_lines*. See e.g. *abs_linesReadFromJpl*. This
field shall be a vector of length 2. If not set, default is
[0,1e99].
   Can be given either as the name of a file, as a matching Matlab
variable or control file text.

ABS_LINES_FORMAT:
The format of spectroscopic data.
   Possible line formats are 'XML', 'Artscat', 'Hitran', 'Jpl' and
'Mytran2'. The first option (XML) expects the ABS_LINES is line
dataincluding meta data (i.e. the system inroduced 2020).
   The field can further be set to 'None' which indicates that no
line data shall be included (the WSV *abs_lines_per_species* is
set to be empty).

ABS_LINESHAPE:
The line shape to use. The same line shape is used for all tag
groups. Shall be a string. Do "arts -d abs_linesSetLineShapeType"
to list valid options.

ABS_LINESHAPE_CUTOFF:
The line shape cut-off to apply. The same cut-off is used for all
tag groups. A value of <0 means no cut-off. Do "arts -d
abslineshapeDefine" for further information.

ABS_LINESHAPE_FACTOR:
The line shape normalisation factor to apply. The same factor is
used for all tag groups. Shall be a string. Do "arts -d
```

abs_linesSetNormalization" for valid options.

ABS_LINESHAPE_MIRRORING:
Option for mirroring of lines.Do "arts -d abs_linesSetMirroring"
for valid options.

ABS_LOOKUP:
Has the same functionality as the arts WSV with the same name.
   Can be given either as the name of a file, as a matching Matlab
variable or control file text.

ABS_MODELS:
Control files including descriptions of absorption models to use
(called continua in arts). This shall be a series of calls of
*abs_cont_descriptionAppend* (call of *abs_cont_descriptionInit*
shall be included).
   Given as an array of strings. The atmlab setting ARTS_INCLUDES
is recognised. A file containing standard choices is:
ARTS_INCLUDES/continua.arts

ABS_NLS:
As the arts WSV with the same name. Only needed if an absorption
look-up shall be calculated. Can be set by *qarts_abs_species*.
   Can be given either as the name of a file, as a matching Matlab
variable or control file text.

ABS_NLS_PERT:
As the arts WSV with the same name. Only needed if an absorption
look-up shall be calculated. See *qarts_set_abs_lookup* for simple
setting of this field.
   Can be given either as the name of a file, as a matching Matlab
variable or control file text. Empty ([]) signifies no
perturbations.

ABS_P:
As the arts WSV with the same name. Only needed if an absorption
look-up shall be calculated. See *qarts_set_abs_lookup* for simple
setting of this field.
   Can be given either as the name of a file, as a matching Matlab
variable or control file text.

ABS_SPECIES:
Specification of absorption species. Arts requires in many cases
that H2O and N2 are included. This is a structure array defining
species and associated retrieval variables. These sub-fields are
defined:

<TAG> Tag group data, following *arts_tgs_cnvrt*.

<ATMDATA> VMR data for the species. The input shall follow the atmdata format, defined in *isatmdata*. The data specified here are not directly given to arts, it can only be used to set-up *VMR_FIELD*. This step is handled by *qarts_vmr_field*. The field is accordingly not mandatory. The main usage of this field should be to import data of climatology character. Can be given as a variable or a file saved through *gf_save*. This field is recognised and used automatically by Qpack2.

<RETRIEVE> If set to true the species is retrieved. Otherwise the species is not retrieved (what a surprise!).

<UNIT> Retrieval unit. Allowed choices are 'rel', 'vmr', 'nd', and 'logrel'. For some more information:
arts -d jacobianAddAbsSpecies.

<GRIDS> Retrieval grids for the species. An array of vectors of length 3: {p_grid,lat_grid,lon_grid}. Grids for dimensions not used shall be empty.

<SX> Covariance matrix of a priori knowledge for the species. Size must match the grid field. Data must match <UNIT>. A matrix, that can be sparse.

<SXINV> The inverse of SX. An optional field. If set, a matrix of same size as SX. Otherwise the inverse is calculated internally. A reason to set this field is if the inverse can be determined by an analytical espression.

<L2> Flag for any function creating L2 data. See comments for *L2_EXTRA*. If set to true, the retrieved state for the species will be included in the L2 output. Otherwise not. This field has no importance if RETRIEVE is false.

<L2_RANGE> Fine tuning of L2 data ranges. The default is to store data directly matching the retrieval grids. This is done if the field is non-existing. This field allows you to either crop or extend range of the L2 data. It is then a vector of length, at least, twice the atmospheric dimensionality: [p_min p_max lat_min lat_max lon_min lon_max] where lat_min gives the minimum latitude to include etc.
 If a given limit is inside the range covered by the corresponding retrieval grid, this results in that the data are cropped. This feature can be used to remove parts that are of no interest for L2 data (such as parts always having a low measurement response).
 The default L2 output is not reflecting the fact that values at end points of the retrieval grids are valid all the way to the matching atmospheric limit. This field can also be used to incorporate this part. If a limit is outside the retrieval grid,

the L2 data will also include the retrieved state at the specified
or the atmospheric limit (which is the closest to the retrieval
grid). In practice this signifies a duplication of end point
data.
 Only retrieved profile and errors are modified. Averaging kernel,
covariance and gain matrices are kept consistent with the original
retrieval grids.


<MINMAX> Min and max allowed value for retrievals. This field has
no impact on linear retrievals. These limits are applied during
the iterations of a non-linear inversion. The limits apply also to
the final result for inversions that converge, but not always for
non-converging inversions,
   A single min and max value is applied for each species. The
field can be left undefined, be empty, be a scalar or vector of
length 2. A single value as taken as min, and is then set to Inf.
The min value can be set to -Inf. The unitfollows the field
*UNIT*.
 The primary usage of the field should be to avoid negative vmr
values.


ABS_T:
As the arts WSV with the same name. Only needed if an absorption
look-up shall be calculated. See *qarts_set_abs_lookup* for simple
setting of this field.
   Can be given either as the name of a file, as a matching Matlab
variable or control file text.

ABS_T_PERT:
As the arts WSV with the same name. Only needed if an absorption
look-up shall be calculated. See *qarts_set_abs_lookup* for simple
setting of this field.
   Can be given either as the name of a file, as a matching Matlab
variable or control file text.

ABS_VMRS:
As the arts WSV with the same name. Only needed if an absorption
look-up shall be calculated. See *qarts_set_abs_lookup* for simple
setting of this field.
   Can be given either as the name of a file, as a matching Matlab
variable or control file text.

ABS_WSMS:
Workspace method calls to include just before calculation of
absorption. Format and options as WSMS_AT_START.

ABS_XSEC_AGENDA:
As the arts WSV with the same name. WSMs are listed as an array of

strings.
   If the field *ABSORPTION* is set, this agenda is set to its
most common choice. Accordingly, this field is only needed if your
calculations require a deviating agenda definition.

ABSORPTION:
String describing how absorption shall be calculated/obtained.
Existing options are:
    {}         : Absorption variables defined by include files. All
absorption fields above will be ignored.
   'OnTheFly' : Absorption is calculated for each propagation path
point. The agenda *abs_scalar_gas_agenda* must here be set by an
include file. No other absorption variables need to be set.
   'CalcTable': Calculate and use an absorption look-up table. See
*qarts_abstable* for simple setting of needed variables.
   'LoadTable': Load and use pre-calculated absorption look-up
table, defined by field *ABS_LOOKUP*.
If this field is set, *ABS_XSEC_AGENDA* and
*PROPMAT_CLEARSKY_AGENDA* are set matching calculations of
standard type.

ANTENNA_DIM:
As the arts WSV with the same name.
   This field can be ignored, depending on SENSOR_RESPONSE.

ATMOSPHERE_DIM:
As the arts WSV with the same name.

BATCH:
Batch calculations. Defined by a structure, described in
*qartsBatch*. Type 'qinfo(@qartsBatch);' for further information.
Batch calculations are not started automatically (in e.g.
*arts_y*), but must be selected specifically (most easily done by
using *arts_batch*).

CLOUDBOX:
Handling of cloudbox/scattering. Defined by a structure, described
in *qartsCloudbox*. Type 'qinfo(@qartsCloudbox);' for further
information.

CLOUDBOX_DO:
Boolean to activate the cloud box (scattering calculations), or
not.
   {}: Nothing is done. Relevant data are assumed to be specified
by inclusion files.
    0: Call of *cloudboxOff* is included.
    1: Action follows setting of field *CLOUDBOX*.

F_GRID:

Has the same functionality as the arts WSV with the same name.
   Can be given either as the name of a file, as a matching Matlab
variable or control file text.

FSHIFTFIT:
Structure for specification of frequency shift retrievals. No
retrievalis made if field is empty or sub-field RETRIEVAL is set
to false. The WSM used is *jacobianAddFreqShift*. The polynomial
order is hard-coded to zero. These sub-fields are defined:

<RETRIEVE> Flag to activate frequency retrieval.

<DF> Size for numerical perturbation.

<SX> Variance of a priori knowledge on the variable.

<SXINV> The inverse of SX. An optional field, see further
ABS_SPECIES.

<L2> Flag for any function creating L2 data. See comments for
*L2_EXTRA*. If set to true, the retrieved frequency fit variables
will be included in the L2 output. Otherwise not. This field has
no importance if RETRIEVE is false.

FSTRETCHFIT:
Structure for specification of frequency stretch retrievals. No
retrieval is made if field is empty or sub-field RETRIEVAL is set
to false. The WSM used is *jacobianAddFreqStretch*. The polynomial
order is hard-coded to zero. These sub-fields are defined:

<RETRIEVE> Flag to activate frequency retrieval.

<DF> Size for numerical perturbation.

<SX> Variance of a priori knowledge on the variable.

<SXINV> The inverse of SX. An optional field, see further
ABS_SPECIES.

<L2> Flag for any function creating L2 data. See comments for
*L2_EXTRA*. If set to true, the retrieved frequency fit variables
will be included in the L2 output. Otherwise not. This field has
no importance if RETRIEVE is false.

HSE:
Variables associated with hydrostatic equilibrium. A structure
with the sub-fields:

<ON> Flag to enforce hydrostatic equilibrium. If true, the WSV

*z_field*is recalculated by *z_fieldFromHSE* (note that a "first guess" *z_field* must be provided).

<P> Pressure for reference point. Matches the WSV *p_hse*.

<ACCURACY> Calculation accuracy. Matches the WSV *z_hse_accuracy*.

If the field is unset, no action is taken. This equals to set Q.HSE.ON to false. Note the setting Q.T.HSE which controls the calculation of t-jacobians. The two settings are treated as independent, but should be set to the same value (true/false). If the HSE is maintained during the iterations of a temperature retrieval is controlled by THIS field.

INCLUDES:
Paths to control files to be included. These files will be included at the top of the control file, only preceded by WSMS_AT_START. Given as an array of strings. Format and options as for WSMS_AT_START.
   The atmlab setting ARTS_INCLUDES is recognised. To include the standard arts general definition file, select:{'ARTS_INCLUDES/general.arts'}

INPUT_FILE_FORMAT:
The file format for arts input files. That is, the format of the files created in Matlab to be read by arts. Possible options are 'binary', and 'double'. The binary option, that is default, should be most efficient. Use 'double' if you want to visually inspect the files generated. (The option 'float' is not allowed, as this is not always sufficient for frequency data.)

IY_AUX_VARS:
As the arts WSV with the same name.

IY_MAIN_AGENDA:
As the arts WSV with the same name. WSMs are listed as an array of strings.

IY_SPACE_AGENDA:
As the arts WSV with the same name. WSMs are listed as an array of strings.

IY_SUB_AGENDA:
As the arts WSV with the same name. WSMs are listed as an array of strings.

IY_SURFACE_AGENDA:
As the arts WSV with the same name. WSMs are listed as an array of

strings.

IY_TRANSMITTER_AGENDA:
As the arts WSV with the same name. WSMs are listed as an array of
strings.

IY_UNIT:
As the arts WSV with the same name. That is, the radiance unit.
The options include:
    '1'      : Basic radiances [W/m2/Hz/sr]
    'RJBT'   : Conversion to brightness temperature by the
Rayleigh-Jeans approximation of the Planck function.
    'PlanckBT': Conversion to brightness temperature by the Planck
function.
For further information: 'arts -d iy_unit'.

J_DO:
Boolean to include calculation of jacobians.
    {}: Nothing is done. Relevant data are assumed to be specified
by inclusion files.
     0: Call of *jacobianOff* is included.
     1: Action follows setting of field *J*.

LAT_GRID:
As the arts WSV with the same name. For example, shall be empty
for 1D simulations.
    Can be given either as the name of a file, as a matching Matlab
variable or control file text.

LAT_TRUE:
As the arts WSV with the same name.
    Can be given either as the name of a file, as a matching Matlab
variable or control file text.

LON_GRID:
As the arts WSV with the same name. For example, shall be empty
for 1D and 2D simulations.
    Can be given either as the name of a file, as a matching Matlab
variable or control file text.

LON_TRUE:
As the arts WSV with the same name.
    Can be given either as the name of a file, as a matching Matlab
variable or control file text.

L2_EXTRA:
Information to functions repacking retrieval information. Such a
product is normally denoted as L2 data. This field lists data to
be included beside the direct retrieval quantities, for which

their L2 field has the same functionality. The information is given as string array, e.g.:

    Q.L2_EXTRA = {'cost','mresp','ptz'}

There is no general L2 function and the possible data output (and coding) differs between the functions.

MAG_U_FIELD:
As the arts WSV with the same name. Can be set to [].
   Can be given either as the name of a file, as a matching Matlab variable or control file text.

MAG_U:
Data and settings associated with MAG_U_FIELD. Has the same role as T with respect to T_FIELD, but as there are no retrievals associated with MAG_U there is only one sub-field here:

<ATMDATA> The input shall follow the atmdata format, defined in *isatmdata*. The data specified here are not directly given to arts, it can only be used to set-up *MAG_U_FIELD*. This step is handled by *qarts_atm_field*. The main usage of this field should be to import data of climatology character. Can be given as a variable or a file saved through *gf_save*. This field is recognised and used automatically by Qpack2.

<RETRIEVE> If set to true, this magnetic field component is retrieved.

<GRIDS> Retrieval grids for the component. An array of vectors of length 3: {p_grid,lat_grid,lon_grid}. Grids for dimensions not used shall be empty.

<SX> Covariance matrix of a priori knowledge. Size must match the grid field. Data must match <UNIT>. A matrix, that can be sparse.

<SXINV> The inverse of SX. An optional field, see further ABS_SPECIES.

<L2> Flag for any function creating L2 data. See comments for *L2_EXTRA*. If set to true, the retrieved state for this component will be included in the L2 output. Otherwise not.

<L2_RANGE>  As the same field for ABS_SPECIES.

<MINMAX> Min and max allowed value for retrievals. Works exactly as for ABS_SPECIES.

None of the fields are mandatory for pure forward calculations. If RETRIEVAL is set to false, the other retrieval related fields are ignored.

```
MAG_V_FIELD:
As the arts WSV with the same name. Can be set to [].
   Can be given either as the name of a file, as a matching Matlab
variable or control file text.


MAG_V:
Data and settings associated with the v-component of the magnetic
field. Defined as MAG_U, and the details are nor repeated here.


MAG_W_FIELD:
As the arts WSV with the same name. Can be set to [].
   Can be given either as the name of a file, as a matching Matlab
variable or control file text.


MAG_W:
Data and settings associated with the w-component of the magnetic
field. Defined as MAG_U, and the details are nor repeated here.


MBLOCK_DLOS_GRID:
Has the same functionality as the arts WSV with the same name.
This has to be set even if the sensor does not include an
antenna.
   Can be given either as the name of a file, as a matching Matlab
variable or control file text.


OUTPUT_FILE_FORMAT:
As the arts WSV with the same name. That is, the format for files
created by arts. Possible options are 'binary' and 'ascii'. The
binary option should in general be most efficient.


P_GRID:
Has the same functionality as the arts WSV with the same name.
   Can be given either as the name of a file, as a matching Matlab
variable or control file text.


POINTING:
Structure for specification of pointing fit retrievals. No
retrieval ismade if field is empty or sub-field RETRIEVAL is set
to false. The WSM used is *jacobianAddPointingZa*. These
sub-fields are defined:


<RETRIEVE> Flag to activate pointing retrieval.


<DZA> Size for numerical perturbation. See used WSM.


<POLY_ORDER> Order of polynomial to describe the time variation of
pointing errors.
```

<CALCMODE> The manner in which the jacobian is determined. See used WSM.

<SX> Covariance matrix of a priori knowledge for pointing fit variables. A square matrix of size POLY_ORDER+1.

<SXINV> The inverse of SX. An optional field, see further ABS_SPECIES.

<L2> Flag for any function creating L2 data. See comments for *L2_EXTRA*. If set to true, the retrieved pointing variables will be included in the L2 output. Otherwise not. This field has no importance if RETRIEVE is false.

POLYFIT:
Structure for specification of polynomial baseline fits. No retrieval ismade if field is empty or sub-field RETRIEVAL is set to false. The WSM used is *jacobianAddPolyfit*. All no_xxx_variation arguments of the WSM are set to default value. The different polynomial coefficients are assumed to be uncorrelated. These sub-fields are defined:

<RETRIEVE> Flag to activate pointing retrieval

<ORDER> Order of polynomial fit.

<SX0> Covariance matrix of a priori knowledge for coefficient of order zero. There shall be such a matrix for each coefficient order until:

<SXn> Covariance matrix of a priori knowledge for coefficient of order POLY_ORDER. The last required covariance matrix.

<SXINVn> The inverse of SXn. An optional field, see further ABS_SPECIES.

<L2> Flag for any function creating L2 data. See comments for *L2_EXTRA*. If set to true, the retrieved baseline variables will be included in the L2 output. Otherwise not. This field has no importance if RETRIEVE is false.

PPATH_AGENDA:
As the arts WSV with the same name. WSMs are listed as an array of strings.

PPATH_LMAX:
As the arts WSV with the same name .
   Can be given either as the name of a file, as a matching Matlab variable or control file text.

PPATH_LRAYTRACE:
As the arts WSV with the same name .
   Can be given either as the name of a file, as a matching Matlab
variable or control file text.

PPATH_STEP_AGENDA:
As the arts WSV with the same name. WSMs are listed as an array of
strings.

PROPMAT_CLEARSKY_AGENDA:
As the arts WSV with the same name. WSMs are listed as an array of
strings.
   If the field *ABSORPTION* is set, this agenda is set to its
most common choice (for the selected ABSORPTION option).
Accordingly, this field is only needed if your calculations
require a deviating agenda definition.

RAW_ATMOSPHERE:
If set, a "raw atmosphere" in read. This is the main file name for
the raw atmosphere. See further 'arts -d AtmRawRead'.
Data in the fields T_FIELD, Z_FIELD or VMR_FIELD are included at a
later stage and will then overwrite the data from the raw
atmosphere. This gives thus some flexibility to mix data from
different sources.

RAW_ATM_EXPAND_1D:
Boolean to expand an 1D raw atmosphere to set ATMOSPHERE_DIM. If
set to 1, the WSM *AtmFieldsCalcExpand1D* is used instead of
*AtmFieldsCalc*.This variable is only used if RAW_ATMOSPHERE is
set.

REFELLIPSOID:
Reference ellipsoid. As the arts WSV with the same name .

REFR_INDEX_AIR_AGENDA:
As the arts WSV with the same name. WSMs are listed as an array of
strings.

RTE_LOS:
As the arts WSV with the same name.

RTE_POS:
As the arts WSV with the same name.

RTE_POS2:
As the arts WSV with the same name.

SENSOR_DO:

Boolean to include sensor characteristics. Otherwise monochromatic
pencil beam calculations are performed. *SENSOR_POS/LOS* are used
in both cases.
   {}: Nothing is done. Relevant data are assumed to be specified
by inclusion files.
    0: Call of *sensorOff* is included.
    1: Action follows setting of field *SENSOR_RESPONSE*.

SENSOR_LOS:
As the arts WSV with the same name.

SENSOR_POL:
As the arts WSV with the same name.

SENSOR_POS:
As the arts WSV with the same name.

SENSOR_RESPONSE:
As the arts WSV with the same name. Three main options exist:
1: The first option is to specify each variable associated with
the sensor individually. This field can then be the name of a XML
file or a Matlab sparse matrix. Other fields that must be
specified for this option include: ANTENNA_DIM, MBLOCK_DLOS_GRID
and SENSOR_RESPONSE_F/DLOS/POL.
2: The name of control files to include (ARTS_INCLUDES
recognised). Given as an array of strings.
3: Calculate the response from basic data. This field shall then
be a structure, following the definitions in *qartsSensor*. Type
'qinfo(@qartsSensor);' for definition of required data
fields.Other fields that must/may be specified here include
ANTENNA_DIM and MBLOCK_DLOS_GRID.

SENSOR_RESPONSE_DLOS:
As the arts WSV with the same name. This field can be ignored,
depending on setting of SENSOR_RESPONSE.
   Can be given either as the name of a file, as a matching Matlab
variable or control file text.

SENSOR_RESPONSE_DLOS_GRID:
As the arts WSV with the same name. This field can be ignored,
depending on setting of SENSOR_RESPONSE.
   Can be given either as the name of a file, as a matching Matlab
variable or control file text.

SENSOR_RESPONSE_F:
As the arts WSV with the same name. This field can be ignored,
depending on setting of SENSOR_RESPONSE.
   Can be given either as the name of a file, as a matching Matlab
variable or control file text.

SENSOR_RESPONSE_F_GRID:
As the arts WSV with the same name. This field can be ignored,
depending on setting of SENSOR_RESPONSE.
    Can be given either as the name of a file, as a matching Matlab
variable or control file text.

SENSOR_RESPONSE_POL:
As the arts WSV with the same name. This field can be ignored,
depending on setting of SENSOR_RESPONSE.
    Can be given either as the name of a file, as a matching Matlab
variable or control file text.

SENSOR_RESPONSE_POL_GRID:
As the arts WSV with the same name. This field can be ignored,
depending on setting of SENSOR_RESPONSE.
    Can be given either as the name of a file, as a matching Matlab
variable or control file text.

SENSOR_TIME:
As the arts WSV with the same name.

SINEFIT:
Structure for specification of sinusoidal baseline fits. No
retrieval ismade if field is empty or sub-field RETRIEVAL is set
to false. The WSM used is *jacobianAddSinefit*. All
no_xxx_variation arguments of the WSM are set to default value.
The different period lengths are assumed to be uncorrelated. These
sub-fields are defined:

<RETRIEVE> Flag to activate retrieval

<PERIODS> Period lengths of the fit.

<SX1> Covariance matrix of a priori knowledge for coefficients of
first period length. This is applied as the covariance for both
the sine and cosine term, assuming that the two terms are
uncorrelated. That is, for single spectra retrievals, SX1 is a
scalar, giving the variance for both a and b, where the baseline
is modelled as a*sin(...)+b*cos(...). There shall be such a matrix
for each period length until:

<SXn> Covariance matrix of a priori knowledge for last period
length. The last required covariance matrix.

<SXINVn> The inverse of SX. An optional field, see further
ABS_SPECIES.

<L2> Flag for any function creating L2 data. See comments for

*L2_EXTRA*. If set to true, the retrieved baseline variables will
be included in the L2 output. Otherwise not. This field has no
importance if RETRIEVE is false.

STOKES_DIM:
As the arts WSV with the same name.

SURFACE_RTPROP_AGENDA:
As the arts WSV with the same name. WSMs are listed as an array of
strings.

T:
Data and setting associated with atmospheric temperatures.
Normally used when retrieving temperature. Otherwise, *T_FIELD*
can be set directly.

<ATMDATA> The input shall follow the atmdata format, defined in
*isatmdata*. The data specified here are not directly given to
arts, it can only be used to set-up *T_FIELD*. This step is
handled by *qarts_atm_field*. The main usage of this field should
be to import data of climatology character. Can be given as a
variable or a file saved through *gf_save*. This field is
recognised and used automatically by Qpack2.

<RETRIEVE> If set to true, atmospheric temperatures are
retrieved.

<GRIDS> Retrieval grids for temperature. An array of vectors of
length 3: {p_grid,lat_grid,lon_grid}. Grids for dimensions not
used shall be empty.

<SX> Covariance matrix of a priori knowledge for temperature. Size
must match the grid field. Data must match <UNIT>. A matrix, that
can be sparse.

<SXINV> The inverse of SX. An optional field, see further
ABS_SPECIES.

<L2> Flag for any function creating L2 data. See comments for
*L2_EXTRA*. If set to true, the retrieved state for temperature
will be included in the L2 output. Otherwise not.

<L2_RANGE>  As the same field for ABS_SPECIES.

<MINMAX> Min and max allowed value for retrievals. Works exactly
as for ABS_SPECIES.

<HSE> As the argument to *jacobianAddTemperature* with same name.
Default here is "on". This refers only to the actual jacobian

calculation. If the atmosphere itself fulfils HSE or not is
controlled by Q.HSE.

None of the fields are mandatory for pure forward calculations. If
RETRIEVAL is set to false, the other retrieval related fields are
ignored.

T_FIELD:
As the arts WSV with the same name. Will replace temperature data
inserted through RAW_ATMOSPHERE.
    Can be given either as the name of a file, as a matching Matlab
variable or control file text.

TNOISE_C:
Thermal noise correlation. A matrix (sparse preferably) giving the
correlation of thermal noise between channels. Hence, the diagonal
elements shall all be one. If the thermal noise is the same for
all channels and has a standard deviation of s, the covariance
matrix is s*s*TNOISE_C.

TRANSMITTER_POS:
As the arts WSV with the same name.
    Can be given either as the name of a file, as a matching Matlab
variable or control file text.

VMR_NEGATIVE_OK:
Corresponds directly to the argument *negative_vmr_ok* of
*atmfields_checkedCalc*. Default is false, and only needed if you
want to the flag to true.

VMR_FIELD:
As the arts WSV with the same name. Will replace VMR data inserted
through RAW_ATMOSPHERE.
    Can be given either as the name of a file, as a matching Matlab
variable or control file text.
    For retrievals, the a priori state for *ABS_SPECIES* is
determined by this field.

WIND_U:
Data and settings associated with the u-component of winds.
Normally used when retrieving winds. Otherwise, *WIND_U_FIELD* can
be set directly.

<ATMDATA> The input shall follow the atmdata format, defined in
*isatmdata*. The data specified here are not directly given to
arts, it can only be used to set-up *WIND_U_FIELD*. This step is
handled by *qarts_atm_field*. The main usage of this field should
be to import data of climatology character. Can be given as a
variable or a file saved through *gf_save*. This field is

recognised and used automatically by Qpack2.

<RETRIEVE> If set to true, this atmospheric wind component is
retrieved.

<GRIDS> Retrieval grids for the wind. An array of vectors of
length 3: {p_grid,lat_grid,lon_grid}. Grids for dimensions not
used shall be empty.

<SX> Covariance matrix of a priori knowledge. Size must match the
grid field. Data must match <UNIT>. A matrix, that can be sparse.

<SXINV> The inverse of SX. An optional field, see further
ABS_SPECIES.

<L2> Flag for any function creating L2 data. See comments for
*L2_EXTRA*. If set to true, the retrieved state for this wind
component will be included in the L2 output. Otherwise not.

<L2_RANGE>  As the same field for ABS_SPECIES.

<MINMAX> Min and max allowed value for retrievals. Works exactly
as for ABS_SPECIES.

None of the fields are mandatory for pure forward calculations. If
RETRIEVAL is set to false, the other retrieval related fields are
ignored.

WIND_U_FIELD:
As the arts WSV with the same name. Can be set to [].
   Can be given either as the name of a file, as a matching Matlab
variable or control file text.

WIND_V:
Data and settings associated with the v-component of winds.
Defined as WIND_U, and the details are nor repeated here.

WIND_V_FIELD:
As the arts WSV with the same name. Can be set to [].
   Can be given either as the name of a file, as a matching Matlab
variable or control file text.

WIND_W:
Data and settings associated with the w-component of winds.
Defined as WIND_U, and the details are nor repeated here.

WIND_W_FIELD:
As the arts WSV with the same name. Can be set to [].
   Can be given either as the name of a file, as a matching Matlab

variable or control file text.

WSMS_AT_END:
Workspace method calls to include at the far end of the control
file. Format and options as for WSMS_AT_START.

WSMS_AT_START:
Workspace method calls to include at the top of the control file.
These calls are only preceded by possible settings set by
INCLUDES.
 Method calls are specified as an cell array of strings. For
example, if no inclusion files are used, these calls could be
useful to include
    { 'VectorSet(abs_n2,[0.7808])', 'abs_cont_descriptionInit' }
A simple if/else/end feature is supported, with the format shown
below. The argument to the if-statement must be a field of Q that
is a boolean. Nested if-statements are not allowed. An example:
    <IF> J_DO
      (something)
    <ELSE>
      (something else)
    <END>
That is, <IF>, <ELSE> and <END> are used as identifiers. The
identifiers must start the line. For <ELSE> and <END>, the rest of
the line must be empty.

WSMS_BEFORE_ATMSURF:
Workspace method calls to include just before setting up the
atmosphere and the surface. That is, before consider e.g.
Q.P_GRID, Q.T_FIELD, Q.Z_SURFACE. Format and options as for
WSMS_AT_START.

WSMS_BEFORE_RTE:
Workspace method calls to include just before execution of
*YCALC_WSMS* or batch core part. Accordingly, this field is
considered only if spectra are calculated (ignored for e.g. pure
absorption calculations). Format and options as for WSMS_AT_START.

YCALC_WSMS:
Workspace method calls for performing radiative transfer
calculations. The standard choice should be that this fields
includes a call of *yCalc*. There is no clear division between
this field and *WSMS_BEFORE_RTE*, but for best flexibility this
field should start with *yCalc* (or corresponding method),
followed by post-processing not handled by other fields.
    Accordingly, this field is considered only if spectra are
calculated (ignored for e.g. pure absorption calculations). Format
and options as for WSMS_AT_START.

Z:
Data and settings associated with Z_FIELD. Has the same role as T
with respect to T_FIELD, but as there are no retrievals associated
with Z there is only one sub-field here:

<ATMDATA> The input shall follow the atmdata format, defined in
*isatmdata*. The data specified here are not directly given to
arts, it can only be used to set-up *Z_FIELD*. This step is
handled by *qarts_atm_field*. The main usage of this field should
be to import data of climatology character. Can be given as a
variable or a file saved through *gf_save*. This field is
recognised and used automatically by Qpack2.

Z_FIELD:
As the arts WSV with the same name. Will replace altitude data
inserted through RAW_ATMOSPHERE.
    Can be given either as the name of a file, as a matching Matlab
variable or control file text.

Z_SURFACE:
Surface altitude (above reference ellipsoid). As the arts WSV with
the same name.
    Can be given either as the name of a file, as a matching Matlab
variable or control file text.

## C The fields of O

The information below is obtained in Atmlab by typing:

```
>> qinfo( @oem )
```

A:
Flag to include averaging kernel matrix in X. Default is 0.

cost:
Flag to include cost values in X and to calculate cost even if
solution method does not require cost values. This affects also
the output to *outfids*.

dx:
Flag to include the sequence of convergence values (defined as for
*stop_dx*).

e:
Flag to include in X the estimate of total retrieval error
(square root of diagonal elements of S). That is, the standard
deviation for the sum of observation and smoothing errors.

eo:
Flag to include in X the estimate of observation error (square
root of diagonal elements of So).

es:
Flag to include in X the estimate of smootning error (square root
of diagonal elements of Ss).

ex:
Flag to include in X the a priori uncertainty (square root of
diagonal elements of Sx).

G:
Flag to include gain matrix in X (alse denoted as Dy).

ga:
Flag to include Marquardt-Levenberg parameter in X. Default is 0.

ga_factor_not_ok:
The factor with which the Marquardt-Levenberg factor is increased
when not a lower cost value is obtained. This starts a new
sub-iteration. This value must be > 1.

ga_factor_ok:
The factor with which the Marquardt-Levenberg factor is decreased
after a lower cost values has been reached. This value must be >

1.

ga_max:
Maximum value for gamma factor for the Marquardt-Levenberg method.
The stops if this value is reached and cost value is still not
decreased. This value must be > 0.

ga_start:
Start value for gamma factor for the Marquardt-Levenberg method.
Type:
    help oem
for a definition of the gamma factor. This value must be >= 0.

itermethod:
Iteration method. Choices are 'GN' for Gauss-Newton and 'ML'
or'LM' for Marquardt-Levenberg.

J:
Flag to include weighting function matrix in X.

jexact:
Flag to select recalculation of J after last iteration. If not set
to 1, J will correspond to x before last iteration. Also used for
the linear case.

jfast:
Flag to always calculate the Jacobian in parallel to the spectrum.
This field is only used for the Marquardt-Levenberg case. This
option can save time if the calculation of the Jacobian is very
fast and the convergence is smooth (few cases where ga has to be
increased). The advanatge of this option is that the next
iteration can be started without a call of the forward model.

linear:
Flag to trigger a linear inversion. Fields like itermethod are
ignored if this option is selected. Default is non-linear (0).

maxiter:
Maximum number of iterations.

msg:
Message to put at the start of output messages. Can include e.g.
number of retrieval case.

outfids:
File identifiers for output messages. Inlcude 1 for the screen.
Set to [] for no output att all.

S:

Flag to include covariance matrix for total error in X. That is,
the sum of So and Ss.

So:
Flag to include covariance matrix for observation error in X.

Ss:
Flag to include covariance matrix for smoothing error in X.

stop_dx:
Stop criterion. The iteration is halted when the change in x is <
stop_dx (see Eq. 5.29 in Rodgers' book). A normalisation to the
length of x is applied.

yf:
Flag to include "fitted spectrum" in X. That is, the simulated
measurement matching retrieved state.

Xiter:
Flag to include all iteration states in X.

# D   The fields of Y

The information below is obtained in Atmlab by typing:

```
>> qinfo( @qp2_y )
```

AA:
Line-of-sight azimuth angle for the measurement. Only considered
if ATMOSPERE_DIM set to 3. A scalar value, or a vector. See
further *ZA*.

DAY:
Measurement time information. All these fields (YEAR, MONTH, ...)
are numeric scalars. This information is primarily used to extract
data from the climatology databases.

F:
Frequency for each value of Y. Not essential information. Qpack2
uses this field only for consistency checks.

HOUR:
Optional data. Allows a more detailed specification of measurement
time. Otherwise as field DAY.

HSE_P:
The reference point when enforcing hydrostatic equilibrium. The
geometrical altitude (HSE_Z) is given for one pressure (HSE_P).

HSE_Z:
The reference point when enforcing hydrostatic equilibrium. The
geometrical altitude (HSE_Z) is given for one press-sure (HSE_P).

LATITUDE:
The geographical position of the measurement. This value is when
interpolating input of ATMDATA type. It also specifies the sensor
position when *LAT_PLATFORM* is not defined (and 2D or 3D).

LAT_PLATFORM:
The latitude of the observation platform. If not given, this
position is set to *LATITUDE*, i.e. the position of measurement.
At least for satellite limb sounding it should be needed to set
this field. It is fully up to the user to make sure that the
specified observation geometry and *LATITUDE* and *LONGITUDE* are
consistent.
   If this field is set, it shall be a vector, following the size
requirement of ZA. The field is only considered for 2D and 3D.

LONGITUDE:
The geographical position of the measurement. This value is when

interpolating input of ATMDATA type. It also specifies the sensor
position when *LON_PLATFORM* is not defined (and 3D).

LON_PLATFORM:
The longitude of the observation platform. If not given, this
position is set to *LONGITUDE*, i.e. the position of measurement.
At least for satellite limb sounding it should be needed to set
this field. It is fully up to the user to make sure that the
specified observation geometry and *LATITUDE* and *LONGITUDE* are
consistent.
    If this field is set, it shall be a vector, following the size
requirement of ZA. The field is only considered for 3D.

MINUTE:
Optional data. Allows a more detailed specification of measurement
time. Otherwise as field DAY.

MONTH:
Measurement time information. All these fields (YEAR, MONTH, ...)
are numeric scalars. This information is primarily used to extract
data from the climatology databases.

SECOND:
Optional data. Allows a more detailed specification of measurement
time. Otherwise as field DAY.

TNOISE:
Magnitude of thermal noise, given as 1 standard deviation. A
scalar or a column vector having the same length as y. If a
scalar, the value is applied for all spectrometer channels. If the
measurement consists of several spectra (see *ZA*), this field
must be a matrix (or row vector) where the columns give the noise
for each spectrum.

Y:
The spectrum. If multiple spectra, all data appended into a column
vector.

YEAR:
Measurement time information. All these fields (YEAR, MONTH, ...)
are numeric scalars. This information is primarily used to extract
data from the climatology databases.

ZA:
Line-of-sight zenith angle for the measurement. A scalar value, or
a vector. The fields *Z_PLATFORM* and *ZA*, as well as *AA* for 3D
case, must have the same size. The same applied to *LAT_PLATFORM
and *LON_PLATFORM* is these fields are used. If *ZA* has a length
> 1, the measurement will consist of several spectra. In ARTS,

these spectra are simulated as separate measurement blocks.
Further, *TNOISE* must have a matching size, where the numbers of
columns shall match length of *ZA*.

Z_PLATFORM:
Altitude (above geoid) of observation platform. A scalar value, or
a vector. See further *ZA*.