Qpack2 – atmospheric retrievals by OEM in Matlab

Patrick Eriksson
(patrick.eriksson@chalmers.se)

April 27, 2012

1 Scope

Qpack2 is a retrieval system, implemented in Matlab, for performing inversions of atmospheric observations inside the framework of "optimal estimation" (OEM). So far, only cases where the aim is to retrieve vertical profiles of atmospheric quantities (1D cases) are handled. Accordingly, the present main application areas of Qpack2 are ground-based observations and inversion of individual satellite spectra/scans. Inversion of satellite scanning sequences where atmospheric fields to be retrieved have also a latitude or longitude extension (2D or 3D) could be considered as part of future development.

The package should be general and flexible for covered measurements. For example, there is basically no limitations regarding observation geometry. 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 more general 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 (not part of Atmlab). The communication with ARTS-2 is made through the Qarts system. OEM inversions are performed by the function `oem.m`. Climatology data are stored and interpolated through the atmdata format.

This can be seen as a direct successor of Qpack (*Eriksson et al.*, 2005), despite all code is written from scratch. However, 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 are removed as it was hardly used.

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`, `FMODEL_VERBOSITY`, `VERBOSITY` 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( '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 Qpack2 (`atmlab/retrieval/qpack2/ChangeLog`). The versions of Qpack2 are numbered as Qpack2.x.y, where x is increased when a change can not be accommodated with backward compatibility and y is increased for each change/commit.

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 (dummy) 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 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.

A full example is found in the file `qpack2_demo` (placed in the folder `demos`). The example works with simulated data, created by `qpack2` itself. The 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 inside 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. Maybe most importantly, 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. If any field is added to `Q`, it shall not change the result of older setting files. If a needed addition can not be accommodated with backward compatibility, this generates a new version of Qarts.

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_PROP_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 `tests`. 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.ATMOSPHERE_DIM` must be set to 1.
- `Q.R_GEOID` must be set.
- `Q.RAW_ATMOSPHERE` must be unset (`{}`).
- The fields `T_FIELD`, `VMR_FIELD` and `Z_FIELD` are ignored. These fields are set by Qpack2 by the corresponding "atmdata"-fields.
- `Q.ABS_SPECIES.ATMDATA` must be set.
- `Q.T.ATMDATA` must be set.
- `Q.Z_ATMDATA` is optional. If not set, geometrical altitudes of the pressure levels (the WSV `z_field`) are found by applying hydrostatic equilibrium using `Y.HSE_P` and `Y.HSE_Z` (Sec. 6.3). This later option demands that `Q.HSE.ON` is true.
- `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.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 (spectra or scans) 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`.

`P.year`, `P.month`, `P.day`, `P.hour`, `P.minute`, and `P.second`:
The time of the measurement.

`P.latitude` and `P.longitude`:
The geographical position of the measurement.

`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).

`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.

`y`:
Measured spectrum.

`yf`:
Fitted spectrum. That is, the spectrum matching retrieved state vector.

`baseline`:
Retrieved "baseline". That is, retrieved spectrum distortion through e.g. polynomial fitting. A scalar 0 if no baseline retrieval is performed.

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

`dx`:
Change in the state vector x between each iteration. See further `oem.m`.

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.)

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`.

`species1_e`:
Total retrieval (observation + smoothing) error for the species. Unit depends on `Q.SPECIES(i).UNIT`.

`species1_eo`:
Observation error for the species. Unit depends on `Q.SPECIES(i).UNIT`.

`species1_es`:
Smoothing error for the species. Unit depends on `Q.SPECIES(i).UNIT`.

`species1_mr`:
Measurement response for the species. This is the sum of the rows of `species_A` (see below).

`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.

`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 is so far only a single main approach for baseline fits, `POLYFIT`. 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.).

6.4.5 Frequency fit variables

If only a "frequency shift" retrieval is performed, the fields listed below hold scalars. If also a "stretch fit" is made, the fields are vectors of length 2. The set of fields are:

`P.ffit_x`, `P.ffit_xa`, `P.ffit_e`, `P.ffit_eo`, `P.ffit_es` and
`P.ffit_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' );
```

data of atmdata type are obtained:

```
>> 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).

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.

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 intrnal 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*.
%
% FORMAT   L2 = qpack2_demo
%
% OUT      L2   L2 data output from *qpack2*.

% 2010-05-12   Created by Patrick Eriksson.

function L2 = qpack2_demo


%- 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( 'Some mismatch between Q.F_BACKEND and frequencies of spectra.' );
end
%
[Y.F] = deal( {} );      % The field F is now obselete. {} is used inside
                         % qarts and qpack2 to flag undefined fields


%- OEM variables
%
O = qp2_l2( Q );         % This ensures that OEM returns the varibles 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
  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]' );
  legend( 'Retrieval 1', 'Retrieval 2', 'True and a priori' );
end
return
%-------------------------------------------------------------------------
%-------------------------------------------------------------------------
%-------------------------------------------------------------------------




function Q = q_demo

%- Atmlab settings
%
arts_xmldata_path = atmlab( 'ARTS_XMLDATA_PATH' );
arts_includes     = atmlab( 'ARTS_INCLUDES' );
if isnan( arts_xmldata_path )
  error( 'You need to set ARTS_XMLDATA_PATH to run this exmaple.' );
end
if isnan( arts_includes )
  error( 'You need to ARTS_INCLUDES to run this example.' );
end
%
fascod = fullfile( arts_xmldata_path, 'atmosphere', 'fascod' );


%- Init Q
%
```

```
Q = qarts;


%- General
%
Q.INCLUDES            = { fullfile( arts_includes, 'general.arts' ), ...
                          fullfile( arts_includes, 'continua.arts' ) };
Q.ATMOSPHERE_DIM      = 1;
Q.STOKES_DIM          = 1;
Q.J_DO                = true;
Q.CLOUDBOX_DO         = false;


%- Radiative transfer
%
Q.Y_UNIT              = 'RJBT';
Q.YCALC_WSMS          = { 'yCalc' };
%
Q.PPATH_LMAX          = 250;
Q.PPATH_STEP_AGENDA   = { 'ppath_stepGeometric' };


%- Surface
%
Q.R_GEOID            = constants( 'EARTH_RADIUS' );
Q.Z_SURFACE          = 5e3;          % 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    = 'Arts';
%
Q.ABSORPTION          = 'OnTheFly';
Q.ABS_NLS             = [];

%- Pressure grid
%
z_toa                 = 95e3;
%
Q.P_GRID              = z2p_simple( Q.Z_SURFACE-1e3 : 250 : 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_ZA_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, 'climatology', ...
                        'msis90', '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    = { z2p_simple(Q.Z_SURFACE+1e3:2e3:z_toa)', [], [] };
Q.ABS_SPECIES(1).ATMDATA  = gf_artsxml( fullfile( fascod, ...
                            'midlatitude-winter.O3.xml' ), 'O3', 'vmr_field' );
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 );
 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 );
 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 );

 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 );
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.H2O.xml' ), 'H2O', 'vmr_field' );

% O2 and N2 not active, to make example somewhat faster
if 0
%- Oxygen
%
Q.ABS_SPECIES(3).TAG      = { 'O2-PWR93' };
Q.ABS_SPECIES(3).RETRIEVE = false;
Q.ABS_SPECIES(3).ATMDATA  = gf_artsxml( fullfile( fascod, ...
                            'midlatitude-winter.O2.xml' ), 'O2', 'vmr_field' );

%- Nitrogen
%
Q.ABS_SPECIES(4).TAG      = { 'N2-SelfContStandardType' };
Q.ABS_SPECIES(4).RETRIEVE = false;
Q.ABS_SPECIES(4).ATMDATA  = gf_artsxml( fullfile( fascod, ...
                            'midlatitude-winter.N2.xml' ), 'N2', 'vmr_field' );
end



%- Frequency (shift retrieval here, shift+stretch also possible)
%
Q.FFIT.RETRIEVE           = true;
Q.FFIT.DF                 = 50e3;
Q.FFIT.ORDER              = 0;
Q.FFIT.SX                 = 50e3^2;
Q.FFIT.L2                 = true;

%Q.FFIT.RETRIEVE           = true;
%Q.FFIT.DF                 = 50e3;
%Q.FFIT.ORDER              = 1;
%Q.FFIT.SX                 = [300e3^2 0;0 100e3^2];
%Q.FFIT.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;

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         = 70;

% 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 = 0.05;
% 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;

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

% 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

ABSORPTION:
String describing how absorption shall be calculated/obtained.
Existing options are:
    {}   : Absorption variables defined by include files. All absorption fields below 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*.

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. As an array of line data. This option is only allowed when ABS_LINES_FORMAT is set to 'Arts'.

ABS_LINES_FORMAT:
The format of spectroscopic data.
   Possible line formats are 'Arts', 'Hitran', 'Jpl' and 'Mytran2'. Note that these strings must be given exactly as stated here (first upper case etc.).
   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_lineshapeDefine" 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 -1 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_lineshapeDefine" 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 XML file, or as a matching Matlab variable.

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 XML file, or as a matching Matlab variable.

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 XML file, or as a matching Matlab variable. 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 XML file, or as a matching Matlab variable.

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*.

<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.

<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.

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 XML file, or as a matching Matlab variable.

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 XML file, or as a matching Matlab variable. Empty ([]) signifies no perturbations.

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 XML file, or as a matching Matlab variable.

ABS_WSMS:
Workspace method calls to include just before calculation of absorption.

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_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*.

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

EMISSION_AGENDA:
As the arts WSV with the same name. WSMs are listed as an array of strings, with possible arguments.

F_GRID:
Has the same functionality as the arts WSV with the same name.
   Can be given either as the name of a XML file, or as a matching variable.

FFIT:
Structure for specification of frequency fit retrievals. No retrieval is made if field is empty or sub-field RETRIEVAL is set to false. The WSM used is *jacobianAddFreqShiftAndStretch*. These sub-fields are defined:

<RETRIEVE> Flag to activate frequency retrieval.

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

<ORDER> Order of frequency fit, max value is 1 (0=shift, 1=stretch).

<SX> Covariance matrix of a priori knowledge for frequency shift and stretch. A 1x1 or 2x2 matrix, depending on ORDER.

<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 controled 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.
   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_CLEARSKY_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, with possible arguments. See PPATH_STEP_AGENDA for a
practical example on format to use.

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:
Has the same functionality as the arts WSV with the same name,
with the difference that a scalar value is accepted when
ATMOSPHERE_DIM = 1. It could be needed to set such a scalar value
when *qarts_vmr_field* and similar functions are used.
   Can be given either as the name of a XML file, or as a matching
Matlab variable.

LON_GRID:
Has the same functionality as the arts WSV with the same name,
with the difference that a scalar value is accepted when
ATMOSPHERE_DIM = 1. It could be needed to set such a scalar value
when *qarts_vmr_field* and similar functions are used.
   Can be given either as the name of a XML file, or as a matching
Matlab variable.

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.

MBLOCK_AA_GRID:
Has the same functionality as the arts WSV with the same name. This field can be ignored, depending on setting of SENSOR_RESPONSE.

If the antenna dimension is 1, this variable can be left as {}. For 2D antennas it has to be set even if the sensor does not include an antenna.

Can be given either as the name of a XML file, or as a Matlab variable.

MBLOCK_ZA_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 XML file, or as a Matlab variable.

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.

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 *jacobianAddPointing*. 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 pointing errors.

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

<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.

<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_LMAX:
As the arts WSV with the same name .
   Can be given either as the name of a XML file, or as a Matlab variable.

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

PPATH_STEP_AGENDA:
As the arts WSV with the same name.WSMs are listed as an array of strings. An example showing how an agenda definition looks like:
   Q.PPATH_STEP_AGENDA={'ppath_stepGeometric'}

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

R_GEOID:
Geoid radius. As the arts WSV with the same name .

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 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.

REFR_INDEX_AGENDA:
As the arts WSV with the same name. If PPATH_STEP_AGENDA does not involve refraction, this field can be left as {}.
    WSMs are listed as an array of strings. See PPATH_STEP_AGENDA for a practical example on format to use.

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_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_ZA/AA_GRID and SENSOR_RESPONSE_F/ZA/AA/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_ZA/AA_GRID.

SENSOR_RESPONSE_AA:

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 XML file, or as a matching
Matlab variable.

SENSOR_RESPONSE_AA_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 XML file, or as a matching
Matlab variable.

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 XML file, or as a matching
Matlab variable.

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 XML file, or as a matching
Matlab variable.

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 XML file, or as a Matlab
variable.

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 XML file, or as a Matlab
variable.

SENSOR_RESPONSE_ZA:
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 XML file, or as a matching
Matlab variable.

SENSOR_RESPONSE_ZA_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 XML file, or as a matching
Matlab variable.

STOKES_DIM:
As the arts WSV with the same name.

SURFACE_PROP_AGENDA:
As the arts WSV with the same name. WSMs are listed as an array of strings. See PPATH_STEP_AGENDA for a practical example on format to use.

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_t_or_z_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*.

<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.

<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.

<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 controled by Q.HSE.

<METHOD> As the argument to *jacobianAddTemperature* with same name. Default here is "analytical".

<DT> As the argument to *jacobianAddTemperature* with same name. Default here is 1 K.

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 XML file, or as a matching Matlab variable.

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.

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 XML file, or as a matching Matlab variable.
   For retrievals, the a priori state for *ABS_SPECIES* is determined by this field.

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

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

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

WSMS_AT_END:
Workspace method calls to include at the far end of the control file. As WSMS_AT_START beside position in control file.

WSMS_AT_START:
Workspace method calls to include at the top of the control file. These calls will be executed before inclusion of files specified in 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' }

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 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 as for WSMS_AT_START.

Y_UNIT:
As the arts WSV with the same name. That is, the radiance unit. The following options exist:

'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 y_unit'.

Z_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_t_or_z_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*.

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 XML file, or as a matching Matlab variable.

Z_SURFACE:
Surface altitude (above reference ellispoid). As the arts WSV with the same name.

C The fields of O

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

sxnorm:
Flag to internally perform a normalisation of x, based on the diagonal elements of Sx. Numerical problems can occur when the retrieved values differ strongly in magnitude (due to poor condition number for matrix inversions). This flag can be used to overcome this problem.
The inverse of Sx must be calculated internally if this option is used and there is no use in pre-calculating *Sxinv*.

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

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 required 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 presssure (HSE_P).

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

LATITUDE:
The geographical position of the measurement.

LONGITUDE:
The geographical position of the measurement.

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 devation. A scalar or a vector having the same length as y. If a scalar, the value is applied for all spectrometer channels.

Y:
The spectrum.

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.

Z_PLATFORM:
Altitude (above geoid) of observation platform.