The OCO Level 2 Algorithm User's Guide

Hari Nair (`Hari.Nair@jpl.nasa.gov`)
Hartmut Bösch (`Hartmut.Boesch@le.ac.uk`)
James McDuffie (`James.McDuffie@jpl.nasa.gov`)

April 6, 2010

Contents

1 Introduction

The Orbiting Carbon Observatory Mission (OCO) will make the first time-dependent, global measurements of atmospheric carbon dioxide with the precision and resolution needed to characterize its sources and sinks. These measurements will improve humankind's understanding of the processes that regulate atmospheric CO_2 and enable more reliable forecasts of climate change.

The OCO instrument consists of three boresighted high resolution grating spectrometers. Each of these spectrometers measures the intensity of radiation over one of three very narrow Near Infrared (NIR) bands that are sensitive to the presence of CO_2 and O_2.

The L2 Algorithm software takes spectra measured by the OCO instrument and derives the column integrated CO_2 volume mixing ratio (XCO_2). The L2 Algorithm software is also capable of retrieving XCO_2 from observations by the ground based Fourier Transform Spectrometer (FTS) network as well as from observations made by the SCIAMACHY instrument, currently in orbit aboard the ENVISAT satellite.

The code is written in Fortran 90 and runs on the OCO Linux cluster. It has also been run on the JPL institutional supercomputing cluster (cosmos), the JPL High Performance Computing group's Los Angeles cluster, and on the atmospheric science machines at Colorado State University.

2 Getting the code

The code and supporting files can be obtained from the subversion repository on nephthys. You must have an account on nephthys and be listed in the subversion users directory. If you are not in the users directory, please contact Jennifer Kesterson at JPL (`Jennifer.A.Kesterson@jpl.nasa.gov`, 818-393-2568).

The simplest thing to do is to use the same directory structure which exists in the subversion repository:

```
  mkdir -p ${HOME}/alg/L2_Contrib/
  mkdir -p ${HOME}/alg/L2_EXE/
  mkdir -p ${HOME}/alg/L2_Support/
  mkdir -p ${HOME}/alg/L2_Tests/
```

To check out the latest code:

```
  cd ${HOME}/alg/L2_EXE/
  svn checkout ${SVNROOT}/L2_EXE/trunk
```

This will create an `trunk` directory containing the latest code. The value of `$SVNROOT` depends on whether you are accessing the repository from JPL (`export SVNROOT=https://svn/oco/alg/`) or from outside using SSH tunneling (`export SVNROOT=https://localhost:20443/oco/alg/`). See Appendix A for instructions to set up SSH tunneling.

To update your copy of the code, simply enter the `trunk` directory and use the command:

```
  svn update
```

3 Building the code

Enter the `trunk/src` directory and type `make` for some brief instructions:

```
  To build the L2 PGE:
        make oco_l2

  The default compiler is g95.  This can be overridden by using FC= on the command line.
  Valid compilers are:
        f90 (Absoft)
        g95 (GNU)
ifort (Intel)
        f95 (NAG)

  Valid options are:
        BITS=(32/64)   (default 64)
        FTS=(t/f)      (default t)
        DEBUG=(t/f)    (default f)
        PARALLEL=(t/f) (default f)
        OPT=(t/f)      (default f)
        HDF=(t/f)      (default f)

  For example:
        make oco_l2 FC=f90 PARALLEL=t DEBUG=t
```

Some compilers (like NAG) choke on some of the FTS code, so use the `FTS=f` option if you run into this issue. Of course, the resulting executable will not be able to do FTS retrievals. HDF will download and compile the HDF5 libraries using the compiler you specify, but may not work with all compilers. Note that you will need to delete the .depend file and run 'make clean' when switching between the HDF to non-HDF version.

The makefile will create a binary with a name like `oco_l2.g95-1930`, which tells us the compiler used and revision number in the subversion repository.

4 Running the code

A number of testcases are stored under subversion in `$SVNROOT/L2_Tests`.

```
cd ${HOME}/alg/L2_Tests
svn co ${SVNROOT}/L2_Tests/trunk
```

4.1 A Nadir Case

4.1.1 Generating a simulated spectrum

There are a number of nadir test cases in `test_nadir`. Let's generate a simulated OCO spectrum for Park Falls, Wisconsin, in July, with an aerosol optical depth of 0.1.

```
cd ${HOME}/alg/L2_Tests/trunk/test_nadir/pf_Jul1_TROP_01/FM
rsync -a std_input/ oco_l2.g95-1930
```

This will copy the contents of `std_input` into a directory called `oco_l2.g95-1930`. Enter this directory and look at its contents.

```
drwxr-xr-x  5 hnair algorithm 4096 Apr 25 15:33 in/
-rw-r--r--  1 hnair algorithm 6060 Mar 27 15:47 oco_l2.inp
-rw-r--r--  1 hnair algorithm 3189 Mar 27 15:47 oco_l2.run
lrwxrwxrwx  1 hnair algorithm   40 May  1 09:08 oco_l2.win -> ../../../static_fts_locations/oco
drwxr-xr-x  4 hnair algorithm 4096 Apr 25 15:33 out/
```

The majority of input files are located in the `in` directory. The `out` directory stores the run output.

On startup, the code looks for a file in the current directory called `oco_l2.run`. This file contains run parameters that should be sounding independent. The idea is that the user would only need to update `oco_l2.inp` and the appropriate files in the `in` subdirectory for different test cases.

You can run the code in this directory:

```
${HOME}/alg/L2_EXE/trunk/bin/oco_l2.g95-1930 > stdout 2> stderr
```

This particular case takes about 6 minutes to run on coral. Upon completion, you will have `oco_l2.log`, `stderr`, and `stdout` files which contain a lot of diagnostic output.

The generated spectrum is `in/l1b/spec/OCO/OCO_1_00001.0001`. Even though it's an output file in this case, it is an input file for the retrieval, so that's why it's in the `in` directory.

4.1.2 Performing a retrieval

We can now use the spectrum we just generated to perform a retrieval. Edit the `oco_l2.run` file and change

```
run_mode       = FORWARD_MODEL
```

to

```
run_mode       = RETRIEVAL
```

You can now run `oco_l2.g95-1930` as you did before.

```
${HOME}/alg/L2_EXE/trunk/bin/oco_l2.g95-1930 > stdout 2> stderr
```

Although the calculation takes about 90 minutes on coral, the retrieval easily succeeds, as we're starting from the exact state that we used to generate the spectrum. If you want to use a different initial state, you will need to modify files in the `in` directory.

5 Output Files

5.1 Log files

5.1.1 `stdout` and `stderr`

FORTRAN units for standard error (unit 0) and standard output (unit 6) can be redirected to files, otherwise their messages will appear on the screen.

5.1.2 `oco_l2.log`

This file contains diagnostic information, much of which is also output to `stdout`.

5.2 The `out` directory

5.2.1 The `aggregator` directory

This directory contains files that will be used by the Ground Data System to create the L2 product for the DAAC. They may be of interest to other users.

`atm_levels.dat` This file contains retrieved parameters on the atmosphere grid.

`brdf_spec_dep.dat and brdf_spec_indep.dat` These files contain the retrieved BRDF parameters.

`dispersion.dat` This file contains the retrieved dispersion parameters and coefficients.

`scalar.dat` This file contains a number of scalar parameters that are needed for the error analysis.

`sv_names.dat` This file contains the name of each state vector element.

`sv_parameters.dat` This file contains the final state vector values and their uncertainties.

5.2.2 `ak_matrix.dat`

This is the averaging kernel.

5.2.3 `atmosphere.dat`

This contains temperature and gas profiles, updated at every iteration.

5.2.4 The `control1` directory

This directory contains output files used in the error analysis.

`a_col1.dat` Normalized column averaging kernel.

`a_col.dat` Column averaging kernel.

`a_col_sigma.dat` Column averaging kernel per standard deviation.

`aero_od_cov.dat` This is the aerosol covariance matrix, copied from the input directory.

`aer_pd_species.dat` This is the aerosol jacobian.

`alb_pd.dat` This is the albedo jacobian.

`a_targ_col.dat` Normalized column averaging kernel.

`a_targ.dat` Averaging kernel sub-matrix for target gas.

`co2_cov.dat` The is the CO_2 mixing ratio covariance matrix, copied from the input directory.

`control_file.dat`

`dispersion_cov.dat` This is the instrument dispersion covariance matrix, copied from the input directory.

`disp_pd.dat` This is the instrument dispersion jacobian.

`h2o_cov.dat` The is the H_2O mixing ratio covariance matrix, copied from the input directory.

`lambert_cov.dat` This is the surface albedo covariance matrix, copied from the input directory.

`mr_pd_species.dat` This is the gas absorber jacobian.

The `o_l` directory This directory contains diagnostic files for the off-line error analysis code.

`press_pd.dat` This is the surface pressure jacobian.

`pressure_levels.dat` These are the pressure levels used, copied from the file specified using the `pressure_file` keyword in `oco_l2.run`.

`psurf_cov.dat` This is the surface pressure covariance matrix, copied from the input directory.

`rad_meas.dat` This is the measured spectrum. This is the same file that is in the `out` directory.

`shat_diag.dat` Diagonal elements of a posteriori covariance matrix.

`shat_row.dat` Covariance of Xtarget with non-target gas elements.

`shat_targ.dat` A posteriori sub-covariance matrix for CO_2.

`statevector.dat` This file contains the final statevector, along with the *a priori* values.

`surface_pressure.dat` This file contains the final surface pressure.

`t_cov.dat` This is the temperature covariance matrix, copied from the input directory.

`temperature.dat` This file contains the final temperature profile.

`t_pd.dat` This is the temperature jacobian.

`x_target.dat` This contains the final solution for XCO_2 and its estimated error.

5.2.5 `correlation_cof.dat`

Correlation coefficient matrix.

5.2.6 `dof.dat`

Degrees of freedom.

5.2.7 `high_res.dat`

These files contain the high resolution spectrum computed by the forward model for each window in each spectrometer.

5.2.8 `out_info.dat`

Summary of retrieval diagnostics.

5.2.9 `rad_conv.dat`

This file contains the convolved radiance calculated by the forward model.

5.2.10 `rad_meas.dat`

This file contains the measured spectrum. It is identical to the spectrum in `in/l1b/spec/OCO`.

5.2.11 `residual.dat`

This file contains the residual between the measured and computed radiances.

5.2.12 `results.dat`

Summary of retrieval results.

5.2.13 `solar_all.dat`

This file contains the calculated solar spectrum for each spectrometer.

6 Input Files

6.1 The `in` directory

The `in` directory contains sounding specific files, such as the input spectrum, atmospheric state, instrument parameters, and so on. In this test case, the `in` directory contains three subdirectories:

```
drwxr-xr-x  4 hnair algorithm 4096 Apr 25 15:33 l1b/
drwxr-xr-x  5 hnair algorithm 4096 Apr 25 15:33 scene/
lrwxrwxrwx  1 hnair algorithm   35 May  1 09:08 static -> ../../../../static_fts_locations/in/
```

The `in/l1b` contains information that will be extracted from the HDF L1B output. For a forward model spectrum simulation, details of the viewing geometry must be present.

The `in/scene` directory contains atmospheric and surface pressure profiles.

The `in/static` directory contains databases of scattering parameters, ground types, etc. The specific file to use is specified in the `oco_l2.inp` file.

6.2 The `oco_l2.run` File

The `oco_l2.run` file contains parameters that are independent of any individual scene. It should not have to be updated very often. Please note that this name is hard-coded in the executable. The code will look for this file in the current directory when it starts.

This file is mostly a list of keyword/value pairs. The hash mark (#) is the comment marker. Any characters following the hash mark will be ignored. Keywords are case insensitive, and except for paths and filenames, values are also case insensitive.

An example is given in appendix D.2.

6.2.1 General Parameters

`input_file` The `input_file` keyword is the name of the file which contains sounding specific values (like climatology). ***** Maybe it's better to have this file contain a list of input filenames if we are going to have a sounding loop in the code *****

`log_file` The `log_file` keyword is set to the name of the log file for diagnostic output.

`constraint_log_file` The `constraint_log_file` contains information on when constraints were applied; e.g. if a state vector element goes negative. If not present, it defaults to "Positive_constraint.log"

`summary_file` The `summary_file` keyword is set to the name of the summary file for diagnostic output. The summary file contains the values of the initial guess and *a priori* state structures, as well as the state vector at each iteration and other diagnostics used to evaluate the retrieval.

`verbosity` The `verbosity` keyword controls how much output is sent to standard output, standard error, and the log file. It can take the following values:

`DEBUG`	Extensive debugging information
`INFO`	Informational messages that are probably unimportant
`WARNING`	Messages which may be cause for concern
`ERROR`	Messages which are likely errors, but not fatal
`FATAL`	Messages which lead to abnormal program termination

6.2.2 Run Flags

`append` If the `append` keyword is set to true, the results from this run will be appended to the results file. Otherwise, the results file will be created fresh for each sounding.

`control_flag` The `control_flag` keyword controls whether jacobians and covariances are written out to the control directory, defined in `control_path`.

`zero_azimuth` The `zero_azimuth` keyword controls if the azimuth in the sun-target-reflected plane is zero (this is true for nadir & glint, but not for target mode). ***** This will be removed at some point, since it can be determined from the viewing mode *****

6.2.3 Instrument Parameters

`noise_file` The `noise_file` keyword is the name of the noise parameter file. Floor gives a constant noise offset and gain defines an intensity dependent component. The format is

```
'Gain          ' 1.0d-16 1.687d-17 2.934d-16
'Floor         ' 17.37 17.62 26.78
```

Units are W $m^{-2}\mu m^{-1} sr^{-1}$

`num_spectrometers` The `num_spectrometers` keyword defines the number of spectrometers in the instrument. ***** This really belongs in a module in the code, and should not be user defined. *****

`num_channeling_parameters` The `num_channeling_parameters` defines the number of parameters used to describe the channeling. For example, 1 means a constant, 2 means linear, 3 quadratic, etc.

`num_continuum_parameters` number of continuum parameter

`num_dispersion_parameters` number of dispersion parameter

`num_ghost_parameters` number of ghost parameter(not implemented)

`num_stray_parameters` number of straylight paramter (not implemented)

`num_zero_level_parameters` number of zero level parameters

`num_ils_parameters` The `num_ils_parameters` keyword defines the number of polynomials used to describe the ils. The actual functional form of the ils is some combination of these polynomials defined in the code.

num_ils_wndepend The `num_ils_wndepend` keyword defines the number of polynomial coefficients in each ils parameter.

6.2.4 Spectral Windows

spectral_window_file The `spectral_window_file` keyword is the name of the file which contains a description of the spectral windows to be used. The actual id numbers for the windows to be used for this particular run are specified using the `spectral_windows` keyword. The format of the spectral windows file is described in the next section, and a sample file is given in Appendix D.4.

spectral_windows The `spectral_windows` keyword should be specified once for each spectrometer. The value is the list of window ids in this spectrometer. If there are no windows for a spectrometer, the keyword should still be given, with a blank value.

bin_windows The `bin_windows` keyword specifies the window ids for which spectral binning should be used.

6.2.5 Run Parameters

num_levels The `num_levels` keyword defines the number of vertical levels in the model atmosphere.

num_solar_parameters The `num_solar_parameters` keyword defines the number of parameters used in the functional form of the calculated effective temperature at each frequency. The code currently uses four solar parameters: $T_0, A, B,$ and ω.

$$T_{\text{eff}} = T_0 + \frac{A\omega^2}{((\lambda - B)^2 + \omega^2)}$$

target_species The `target_species` keyword defines the species used to compute x_target (mean column and error). This is normally CO_2.

pressure_file The `pressure_file` keyword names the file containing the values of the atmospheric pressure at each model level. The code will read the column labeled PRESSURE. See the sample atmosphere file later in this document for its format.

jacobian_mode The `jacobian_mode` keyword may be either ANALYTIC or FINITE_DIFFERENCE. For FINITE_DIFFERENCE mode, perturbation values need to be specified in the input file for each retrieved parameter.

`run_mode` The `run_mode` keyword may take any one of the following values:

`FORWARD_MODEL`	Run a single iteration of the forward model. Outputs a simulated radiance.
`JACOBIAN_ONLY`	As above, but also output jacobians for each species flagged for retrieval.
`RETRIEVAL`	Run a full retrieval.
`SOLAR`	Only retrieve solar parameters

***** SOLAR mode exists because? *****

6.2.6 Forward Model Parameters

`absco_path` The `absco_path` keyword defines the full path to the absorption coefficient tables.

`polarization` The `polarization` keyword can be TRUE or FALSE, indicating if the polarization correction should be done.

`spec_path` The `spec_path` keyword defines the path to the Level 1B spectrum file. In FORWARD_MODEL and JACOBIAN_ONLY mode, the spectrum file will be created in this directory.

`streams` The `streams` keyword defines the number of streams used in the radiative transfer calculation.

`single_scatter_correction` The `single_scatter_correction` can be TRUE or FALSE, indicating if the single scattering correction should be done.

`delta_m_correction` The `delta_m_correction` can be TRUE or FALSE, indicating if the Δ_m correction should be done.

`interpolation` The `interpolation` keyword sets the resolution of the resolution of fine grid to the calculated (***** Is this the same as convolved?**) grid.

`ils_cycle` The `ils_cycle` keyword

`apo_m` The `apo_m` keyword

`nlines` The `nlines` keyword specifies the number of lines in the solar line file.

`reclen` The `reclen` keyword specifies the number of characters in each record of the solar line file.

`points_sun` The `points_sun` keyword specifies the number of points

6.2.7 Retrieval Parameters

`max_divergence` The `max_divergence` keyword specifies the maximum number of diverging iterations that can be taken.

`max_iterations` The `max_iterations` keyword specifies the maximum number of iterations that can be taken.

`max_chi2` The `max_chi2` keyword specifies the maximum acceptable value of χ^2. A larger value means a failure to converge.

`lm_gamma` The `lm_gamma` keyword specifies the Levenberg-Marquardt gamma.

`scale_convergence` The `scale_convergence` keyword specifies the maximum value of $d\sigma^2/SV$, where SV is the length of the state vector. A larger value means a failure to converge.

`final_rad` ***** Note from Hartmut - final_rad and do_error should be hard coded and not options in the run file *****

The `final_rad` keyword specifies six logical values:

1.
2.
3.
4.
5.
6.

`do_error` The `do_error` keyword specifies five logical values:

1.
2.
3.
4.
5.

`num_diodes` The `num_diodes` keyword specifies the number of pixels in each spectrometer.

6.2.8 Output Files

`output_path` The `output_path` keyword specifies the directory for the output files. All files will be written inside this directory, with the exception of the L1B simulated spectrum file in FORWARD_MODEL run mode and the log file.

`result_file` The `result_file` is the name of the summary result file.

`out_info_file` The `out_info_file` contains additional output information.

`output_each_iteration` The `output_each_iteration` keyword specifies if radiance and jacobian files are to be saved for each iteration. If false, the radiance and jacobian files are overwritten at each iteration.

`control_path` The `control_path` directory is the location where jacobian and covariance files are written for off-line error analysis.

`controlsub_path` The `controlsub_path` directory is the directory within control_path where additional covariance files are written.

`control_file` The `control_file` keyword specifies the name of the file used as input to the off-line error analysis program.

`atmos_file` The `atmos_file` keyword specifies the output model atmosphere file.

`solar_transmittance` The `solar_transmittance` keyword specifies whether solar transmittance files should be written.

`summary_file` The `summary_file` keyword specifies the name of the file containing the state structure (first guess and a priori) and the state vector at each iteration. It will be placed in `output_path`.

`aggregator_dir` The `aggregator_dir` keyword specifies the path to write the aggregator files. These files contain a summary of the retrieval to be used by the GDS to create the L2 product.

6.2.9 To Be Implemented

`diagnostic_path` The `diagnostic_path` keyword specifies the directory underneath output_path where additional diagnostic files will be written.

`high_res_spectra` The `high_res_spectra` keyword specifies whether high resolution spectra files should be saved.

`save_jacobians` The `save_jacobians` keyword specifies whether jacobian files should be written.

6.3 The Spectral Windows File

This file is specified in the run file using the `spectral_window_file keyword`. It contains an arbitrary number of WINDOW blocks. The format is given below.

6.3.1 Non-binning parameters

`id` The `id` keyword is required for each block. It specifies a unique integer identifying this window, used with the `spectral_windows` and `bin_windows` keywords in the run file.

`name` The `name` keyword is a string describing this window. It is only used for diagnostic output.

`units` The `units` keyword can take the values “wavelength” or “wavenumber”.

`range` The `range` keyword specifies the spectral range of this window. It requires two values; the start and stop values.

`species` The `species` keyword specifies the absorbing species present in this spectral window. The list should be separated by spaces and is case insensitive.

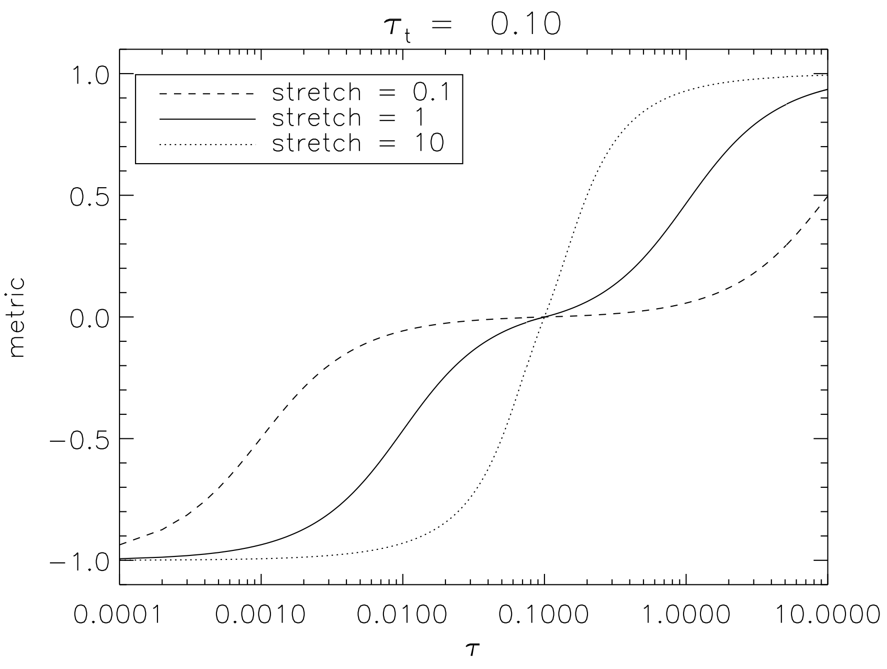
6.3.2 Binning Parameters

The following keywords are used for spectral binning. If a window id is listed with the `bin_windows` keyword in the run file, the corresponding block here must include these parameters.

The radiances are binned on an optical depth vs. scattering grid. The optical depth coordinate ranges from -1 to 1 and is determined by:

$$\begin{aligned} x &= \frac{2}{\pi}\tan^{-1}[S_1\tau_t(1-\frac{\tau_t}{\tau})] && (\tau < \tau_t) \\ &= \frac{2}{\pi}\tan^{-1}[S_2(\tau-\tau_t)] && (\tau \geq \tau_t) \end{aligned}$$

Here S_1 is `stretch1`, S_2 is `stretch2`, and τ_t is `tau_thres`. Figure 1 shows the effect of the stretch on the functional form. The bins are spaced evenly in “metric” space.



tau_thres The usage of `tau_thres` is described above. It should divide "high" and "low" optical depths. A typical value is 0.1.

stretch The `stretch` keyword assigns values for `stretch1` and `stretch2`. The usage is described above.

npoint The `npoint` keyword assigns values for `npoint1` and `npoint2`. There are `npoint1` optical depth bins for `x` $<$ `tau_thres` and `npoint2` optical depth bins for `x` $>$ `tau_thres`

nscatter The `nscatter` keyword specifies the number of scattering bins to use.

6.4 The Sounding Input File

This file is specified in the run file using the input_file keyword. It contains two blocks: the SOUNDING_INFO block and the PARAMETER_DEFINITION block.

6.4.1 The SOUNDING_INFO block

spectrum_file The `spectrum_file` specifies the spectrum data for the retrieval. This can refer either the the output of the forward model or an L1B HDF file.

`range_file` The `range_file` specifies which l1b files should be read.

`soundinginfo_file` The `soundinginfo_file` specifies the scene date and location.

`sounding_number` The `sounding_number` specifies the sounding within L1B and ECMWF HDF files that should be used. The index is 1-based, so if you use `hdfview` to view HDF files, remember that `hdfview` uses 0-based indicies. This means that if you want the data shown at index 4 via `hdfview`, you'll need to specify 5 here.

`frame_number` The `frame_number` specifies the frame within L1B and ECMWF HDF files that should be used. The index is 1-based, so if you use `hdfview` to view HDF files, remember that `hdfview` uses 0-based indicies. This means that if you want the data shown at index 4 via `hdfview`, you'll need to specify 5 here.

6.4.2 The PARAMETER_DEFINITION block

This block contains blocks for each element in the state vector. The `aerosol_types`, `albedo_types`, and `retrieval_vector` keywords should be defined outside of the sub-blocks.

Each of the blocks contained here may use the following keywords:

`name`	Name used for the column label in matrix files that are read in
`a_priori`	Filename containing *a priori* values.
`first_guess`	Filename containing first guess profiles. If not present, *a priori* values are used.
`covariance`	Covariance matrix
`retrieval_indices`	list of retrieval indices, given as individual indices or a range. 1 2 3:11 12 is equivalent to 1:12, which means 1 through 12.
`perturb`	Perturbations used for finite difference jacobians.

HDF Input Files `a_priori` and `first_guess` typically refert to matrix files. However, they may refer to either an L1B or ECMWF HDF file for certain variables as described below:

Block	Variable	HDF File type
GAS	H20	ECMWF
SURFACE_PRESSURE	PSURF	ECMWF
TEMPERATURE	T	ECMWF
INSTURMENT	DISP	L1B
INSTURMENT	ILS	L1B*

* Note that for the ILS, you must always have an ASCII .dat file because certain variables not available in L1B HDF files are are read from the headers of the ASCII file. To specify an L1B file for the ILS, add a 'hdf_file' header to the ASCII file that contains the path on an L1B HDF file.

`aerosol_types` The `aerosol_types` lists the aerosol types (can be more than one) to be used in this run. Each name here must have a matching name in an AEROSOL block later in the file.

`ground_type` The `ground_type` lists the ground type (only specify one) to be used in this run. Each name here must have a matching name in a BRDF block later in the file.

`retrieval_vector` The `retrieval_vector` keyword lists the elements of the state vector that are to be retrieved. The `name` field of each state vector element to be retrieved should be placed here. For the BRDF block, use the name for the whole BRDF block, and not the names for the individual sub-blocks.

AEROSOL block The AEROSOL block contains the following keywords in addition to the general ones specified above:

`mie_file`	Mie scattering parameters
`moment_file`	Aerosol phase function moments
`retrieval_mode`	can be LINEAR or LOGARITHMIC

BRDF block A BRDF type can have spectrally dependent and independent parameters. Each type is contained in either the SPECTRALLY_DEPENDENT or SPECTRALLY_INDEPENDENT sub-blocks.

`type`	BRDF type - currently either lambert or coxmunk. Specify this outside of the sub-blocks.
`num_parameters`	number of parameters used to fit the function
`num_coefficients`	number of polynomial coefficients for each parameter. Only used in the spectrally dependent sub-block.

GAS block The GAS block contains the following keywords in addition to the general ones specified above:

`hitran_index`	HITRAN index of the gas - not used
`isotope`	isotope label (e.g. 626 for $O^{16}C^{12}O^{16}$) - not used
`absco_file`	absorption coefficient file name
`scale_parameters`	Four values: *a priori*, initial guess, perturbation, covariance

The `scale_parameters` keyword is optional. If present, a single scaling factor will be retrieved. This scaling factor is multiplied by the profile to give the best fit.

INSTRUMENT block A separate INSTRUMENT block needs to be specified for each instrument parameter with the `type` keyword.

`type`	CHANNELING, CONTINUUM, DISPERSION, GHOST, ILS, STRAY_LIGHT, or ZERO_LEVEL

SOLAR block

`solar_linelist`	list of solar lines, output from GFIT

SURFACE_PRESSURE block The SURFACE_PRESSURE block has no keywords specific to it.

TEMPERATURE block

`scale_parameters`	Four values: *a priori*, initial guess, perturbation, covariance

As in the GAS block, the `scale_parameters` keyword is optional.

A SSH Tunneling

The JPL firewall restricts access to most of the services on nephthys. External users can use SSH to connect, but the web and subversion servers are not accessible. SSH tunneling can be used to "tunnel" the web and subversion connections through SSH.

On the Colorado State system, I have created a file called `config` in `/home/nair/.ssh` containing:

```
Host nephthys
   ForwardX11   yes
   HostName     nephthys.jpl.nasa.gov
   LocalForward 20000 137.78.162.28:80
   LocalForward 20443 137.78.162.28:443
   ServerAliveInterval 600
   User         hnair
```

This sets up ports on your local machine (e.g. coral.atmos.colostate.edu) that are redirected to ports on nephthys. For example, the web server on nephthys is active on port 80. The line

```
LocalForward 20000 137.78.162.28:80
```

redirects port 80 on IP address 137.78.162.28 (you can also use 127.0.0.1, since this is how a unix machine refers to itself) to port 20000 on coral. The subversion server is active on port 443 on nephthys. This gets mapped to port 20443 on coral.

So to access the OCO Wiki, create an `$HOME/.ssh/config` file like the one above (using your username, of course), and then `ssh nephthys`.

Now if you open up a browser and point it to `http://localhost:20000/~hnair/OCOWiki/`, you should see the wiki page.

If you close the ssh connection to nephthys and try to reload the wiki page, you will get an error.

To list the contents of the subversion repository:

```
svn ls https://localhost:20443/oco/alg
```

You should see:

```
L2_Contrib/
L2_EXE/
L2_Support/
L2_Tests/
```

B Using subversion

All of the OCO code, including that developed by the GDS, is managed using `subversion`. There are only a few commands you really need to know:

Command	Short form	Description
`svn checkout`	`svn co`	Checkout a working copy
`svn update`	`svn up`	Update working copy
`svn diff`	`svn di`	Show differences between two versions
`svn list`	`svn ls`	List directory contents
`svn log`	`svn log`	Print commit log messages
`svn revert`	`svn revert`	Revert working copy to original version
`svn status`	`svn st`	Print status of working copy

C Using screen

`screen` is an enormously useful program that is included on most modern Unix systems (including OS X). It creates a terminal session that can be detached and reattached at any time. What this means is that you can set a job running inside of a screen session, detach it and go home, and then reattach it from home to check on its progress.

It is also very useful for setting up ssh tunnels. On the Colorado State computers, I ssh to nephthys using a screen session, and then detach. When I don't need the tunnel anymore, I reattach the screen and log out.

`screen` creates a new screen. `screen -r` reattaches a detached screen. Basic commands within screen are:

Key sequence	Name	Description
`ctrl-a ''`	windowlist	Show the list of windows
`ctrl-a A`	title	Name this window
`ctrl-a c`	create	Create a new window
`ctrl-a d`	detach	Detach the session
`ctrl-a n`	next	Go to the next window
`ctrl-a p`	previous	Go to the previous window

D Sample Files

D.1 Matrix File

This is a file that can specified as an a_priori or first_guess file in a GAS parameter block, and/or as the pressure_file in the oco_l2.run file.

```
begin HEADER
  File_ID = "Atmospheric State - Lauder, July"
  File_Creation = "Tue Oct 10 15:35:10 2006"
  File_Type = "Matrix"
  Num_Rows =      12
  Num_Columns =      5
  Labels = "Pressure" "T" "H2O" "CO2" "O2"
  Units  = "Pa" "K" "VMR" "VMR" "VMR"
end HEADER

# Pressure T H2O CO2 O2
# (Pa)            (K)     (VMR)           (VMR)   (VMR)
      100.000      245.360  5.11594e-06  0.000368931 0.2095
      7000.00      221.815  3.23449e-06  0.000374329 0.2095
      10000.0      221.558  2.92339e-06  0.000375367 0.2095
      20000.0      219.754  5.56769e-06  0.000375895 0.2095
      35000.0      228.431  0.000170962  0.000376235 0.2095
      45000.0      240.849  0.000621730  0.000376108 0.2095
      55000.0      252.295   0.00155334  0.000375761 0.2095
      65000.0      261.206   0.00281429  0.000375366 0.2095
      75000.0      266.232   0.00401206  0.000375146 0.2095
      85000.0      270.859   0.00556349  0.000375098 0.2095
      95000.0      277.408   0.00669547  0.000374922 0.2095
      105000.      283.154   0.00687684  0.000374915 0.2095
```

D.2 Run File

```
# These are useful to keep at the top, as you probably want to catch
# error messages as this file is being parsed.  Choices for verbosity
# are are DEBUG, INFO, WARNING, ERROR, FATAL.
verbosity           = DEBUG
log_file            = oco_l2.log


### Input file for this run
input_file     = oco_l2.inp

### Run flags
append = True
control_flag = TRUE
zero_azimuth = TRUE

### Instrument parameters
noise_file                 = in/instrument/noise.dat
num_spectrometers          = 3
num_channeling_parameters = 0
num_continuum_parameters  = 0
num_dispersion_parameters = 3
num_ghost_parameters      = 0

num_ils_parameters        = 1

# describe each ils parameter by a polynomial with this many
# coefficients (1 means each parameter is constant, 2 linear, etc.)
num_ils_wndepend          = 1

num_stray_parameters      = 0
num_zero_level_parameters = 1

### Define Spectral Windows
spectral_window_file = oco_l2.win
spectral_windows = 1 # Spectrometer 1
spectral_windows = 2 # Spectrometer 2
spectral_windows = 3 # Spectrometer 3
#bin_windows = 1 2 3

### Run parameters
num_levels                = 12
num_solar_parameters      = 4

target_species            = CO2
pressure_file = in/scene/atmosphere/pressure.dat # Reads the column labeled PRESSURE

jacobian_mode  = ANALYTIC
#jacobian_mode  = FINITE_DIFFERENCE

#run_mode       = FORWARD_MODEL # Run a single iteration of the forward model
run_mode       = JACOBIAN_ONLY # Calculate jacobians, don't do a retrieval
#run_mode      = RETRIEVAL     # Run a full retrieval
#run_mode      = SOLAR         # Retrieve Solar model parameters

# Forward model parameters
absco_path    = /state/partition1/ABSCO/v2.0.1/
polarization  = false             # Turn on polarization
spec_path     = in/l1b/spec/OCO/  # Path of l1b spectrum file
streams       = 8                 # Number of streams that RADIANT will use

single_scatter_correction = false
delta_m_scaling = true

interpolation = 100 100 100       # resolution of fine grid/calculated grid
ils_cycle = 6 6 6
apo_m = 1
nlines = 20585
reclen = 101
points_sun = 10000

# retrieval parameters
max_divergence    = 30   # stop after this number of diverging steps
```

```
max_iterations    = 30   # stop after this many iterations
max_chi2          = 1.0  # fail convergence if chi2 > max_chi2
lm_gamma          = 0.0  # Levenberg-Marquardt gamma
scale_convergence = 1.0  # converged if d_sigma_sq < scale_convergence * len_sv
final_rad = t t t f f f
do_error = True tRue f false F

num_diodes = 1024 1024 1024

# output files

# all output files will go here, except log file
output_path   = out/
result_file   = results.dat     # state vector output file name
out_info_file = out_info.dat    # additional output file
output_each_iteration = FALSE   # save radiances & jacobians at each iteration

# Save jacobians and covariances in this dir for offline error
# analysis.  If not present, no files will be saved.
control_path      = control1/         # underneath output_path
controlsub_path   = o_l/              # underneath control_path
control_file      = control_file.dat  # in control_path

# Atmosphere file
atmos_file = atmosphere.dat

### Anything below this is not implemented yet

# All diagnostic files go here.  If not present, no files will be created.
diagnostic_path     = diagnostics/  # underneath output_path
high_res_spectra    = TRUE          # create high-resolution spectra files
save_jacobians      = TRUE          # save jacobian files
solar_transmittance = TRUE
```

D.3 Input File

```
# Lauder, July
# soundinginfo_file = in/l1b/soundinginfo_la_Jul1.dat

# A priori for gas/temperature, psurf, brdf, and aerosol type:
# a_priori          = in/scene/atmosphere/atmosphere_la_Jul1.dat
# a_priori          = in/scene/psurf/psurf_la_Jul1.dat
# a_priori          = in/scene/frost.dat
# mie_file          = ./in/scene/aerosol/vij_2.mie
# moment_file       = ./in/scene/aerosol/vij_2.mom
```

```
begin SOUNDING_INFO

  range_file   = in/l1b/range.dat
  soundinginfo_file = in/l1b/soundinginfo_la_Jul1.dat

end SOUNDING_INFO

### Define Initial State and Retrieval Parameters
begin PARAMETER_DEFINITION

  aerosol_types = TROP_01 STRAT
  ground_type  = frost

  retrieval_vector = ALBEDO CO2

# The following blocks define the retrieval vector.
  begin GAS
    name              = CO2
    hitran_index      = 2
    isotope           = 626
    a_priori          = in/scene/atmosphere/atmosphere_la_Jul1.dat
    absco_file        = co2_4700_6500_v21.abs
# If first guess is not specified, a priori will be used
#   first_guess       = ./in/scene/atmosphere/atmosphere_fg.dat
    perturb           = ./in/scene/atmosphere/atmosphere_perturb.dat
    covariance        = ./in/covariance/co2_cov.dat
    retrieval_indices = 1:12       # retrieve levels 1 through 12
# retrieve single scaling factor instead of a profile
# A priori, initial guess, perturbation, covariance
#    scale_parameters = 1.0 1.0 0.01 0.1
  end GAS

  begin GAS
    name              = H2O
    hitran_index      = 1
    isotope           = 161
    a_priori          = in/scene/atmosphere/atmosphere_la_Jul1.dat
    perturb           = ./in/scene/atmosphere/atmosphere_perturb.dat
    absco_file        = h2o_4700_6500_v21.abs
    covariance        = ./in/covariance/h2o_cov.dat
    retrieval_indices = 1:12       # retrieve levels 1 through 12
  end GAS

  begin GAS
    name              = O2
    hitran_index      = 7
    isotope           = 66
    a_priori          = in/scene/atmosphere/atmosphere_la_Jul1.dat
    absco_file        = o2_12745_13245_v21.abs
  end GAS

# Surface pressure is not being retrieved but there’s no harm in
# putting covariance and pertubation files here
  begin SURFACE_PRESSURE
    name              = PSURF
    a_priori          = in/scene/psurf/psurf_la_Jul1.dat
    covariance        = ./in/covariance/psurf_cov.dat
    perturb           = ./in/scene/psurf/psurf_perturb.dat
    retrieval_indices = 1
  end SURFACE_PRESSURE

  begin TEMPERATURE
    name              = T
    a_priori          = in/scene/atmosphere/atmosphere_la_Jul1.dat
    covariance        = ./in/covariance/t_cov.dat
    perturb           = ./in/scene/atmosphere/atmosphere_perturb.dat
    retrieval_indices = 1:12
# retrieve single additive factor instead of a profile
# A priori, initial guess, perturbation, covariance
#     scale             = 0 0 2 10
  end TEMPERATURE

  begin AEROSOL
    name              = TROP_01
    a_priori          = ./in/scene/aerosol/aerosol.dat
    covariance        = ./in/covariance/aero_od_01_cov.dat
    mie_file          = ./in/scene/aerosol/vij_2.mie
    moment_file       = ./in/scene/aerosol/vij_2.mom
    perturb           = ./in/scene/aerosol/aerosol_perturb.dat
    retrieval_indices = 1:12
  end AEROSOL

  begin AEROSOL
    name              = STRAT
    a_priori          = ./in/scene/aerosol/aerosol.dat
    mie_file          = ./in/scene/aerosol/strat.mie
    moment_file       = ./in/scene/aerosol/strat.mom
  end AEROSOL

 begin BRDF
    name        = frost
    type        = lambert
    begin SPECTRALLY_DEPENDENT
      num_parameters    = 1
      num_coefficients  = 2
      name              = albedo
      a_priori          = ./in/scene/ground/frost.dat
      covariance        = ./in/covariance/lambert_cov.dat
      perturb           = ./in/scene/ground/lambert_perturb.dat

# Don't comment these out; just remove the indices if you don't want
# to retrieve them.
      retrieval_indices = 1 2   # spectrometer 1, parameter 1 coefficients
      retrieval_indices = 1 2   # spectrometer 2, parameter 1 coefficients
      retrieval_indices = 1 2   # spectrometer 3, parameter 1 coefficients
    end SPECTRALLY_DEPENDENT
  end BRDF

  begin INSTRUMENT
    type              = DISPERSION
    name              = DISP
    a_priori          = ./in/instrument/dispersion.dat
    covariance        = ./in/instrument/dispersion_cov.dat
    perturb           = ./in/instrument/dispersion_perturb.dat

# Don't comment these out; just remove the indices if you don't want
# to retrieve them.
    retrieval_indices = 1 2 3       # spectrometer 1
    retrieval_indices = 1 2 3       # spectrometer 2
    retrieval_indices = 1 2 3       # spectrometer 3
  end INSTRUMENT

  begin INSTRUMENT
    type              = ZERO_LEVEL
    name              = ZERO
    a_priori          = ./in/instrument/zero_level.dat
    covariance        = ./in/instrument/zero_level_cov.dat
    perturb           = ./in/instrument/zero_level_perturb.dat

# Don't comment these out; just remove the indices if you don't want
# to retrieve them.
    retrieval_indices =       # spectrometer 1
    retrieval_indices =       # spectrometer 2
    retrieval_indices =       # spectrometer 3
  end INSTRUMENT

  begin INSTRUMENT
    type              = ILS
    name              = ILS
    a_priori          = ./in/instrument/ils.dat
    covariance        = ./in/instrument/ils_cov.dat
    perturb           = ./in/instrument/ils_perturb.dat

# Don't comment these out; just remove the indices if you don't want
# to retrieve them.  These lines are read in a loop like this:
# do i = 1, n_spec
#   do j = 1, num_ils_parameters
#     read num_ils_wndepend polynomial coefficients for each ils parameter
#   end do
# end do
    retrieval_indices =       # spectrometer 1, parameter 1 coefficients
    retrieval_indices =       # spectrometer 2, parameter 1 coefficients
    retrieval_indices =       # spectrometer 3, parameter 1 coefficients
  end INSTRUMENT

  begin SOLAR
    a_priori       = ./in/solar/solar.dat
    covariance     = ./in/solar/solar_cov.dat
    perturb        = ./in/solar/solar_perturb.dat
    solar_linelist = ./in/solar/solar_di.h92
#    retrieval_indices = 1:4
  end SOLAR

end PARAMETER_DEFINITION
```

D.4 Spectral Windows File

```
begin WINDOW
      id = 1
      name = O2 A Band
      units = wavelength
      range = 0.755 0.785
      species = O2

      # Binning parameters
      tau_thres = 0.2
      stretch   = 12 1
      npoint    = 20 7
      nscatter  = 3

end WINDOW

begin WINDOW
```

```
      id = 2
      name = Weak CO2
      units = wavelength
      range = 1.58 1.65
      species = CO2 H2O

      tau_thres = 0.1
      stretch   = 20 6
      npoint    = 20 5
      nscatter  = 3
end WINDOW

begin WINDOW
      id = 3
      name = Strong CO2
      units = wavelength
      range = 2.03 2.09
      species = CO2

      tau_thres = 0.2
      stretch   = 12 4
      npoint    = 20 5
      nscatter  = 3
end WINDOW
```