NWPSAF ECMWF IASI PCA-Based Compression Package Manual

Andrew Collard, ECMWF, Reading, UK

NWPSAF-EC-UD-011

Version 1.0: 30th January 2008

This software was developed within the context of the EUMETSAT Satellite Application Facility on Numerical Weather Prediction (NWP SAF), under the Cooperation Agreement dated 1st December 2006, between EUMETSAT and the Met Office, UK, by one or more partners within the NWP SAF. The partners in the NWP SAF are the Met Office, ECMWF, KNMI and Météo-France.

Please note that this package requires LAPACK to be installed on your system.

Click here for a pdf version of this file (better for printing).

Contents

1. Introduction

This software package contains the basic routines for producing *reconstructed radiances* from spectra obtained from advanced infrared sounders.

Reconstructed radiances are radiances that have been intelligently smoothed such that the atmospheric signal is retained while the instrument noise is suppressed. While the information content of the entire spectrum cannot be increased through the reconstruction process, it allows for the efficient compression of the information from the entire spectrum into a reduced number of channels.

Reconstructed radiances (Antonelli *et al.*, 2004) are formed through the evaluation of the amplitudes, **p**, of the principal components, **L**, of the observed spectrum. Here, **L** is the set of N_p leading eigenvectors of the covariance matrix of a representative set of thousands of spectra. **p** is related to **y** (the noise normalised radiances with the mean radiance subtracted) through

$$\mathbf{p} = \mathbf{L}^{\mathbf{T}}\mathbf{y}$$

The reconstructed radiances, $\tilde{\mathbf{y}}$, are then calculated from:

$$\tilde{\mathbf{y}} = \mathbf{L}\mathbf{p} = \mathbf{L}\mathbf{L}^{\mathbf{T}}\mathbf{y}$$

If we restrict $\tilde{\mathbf{y}}$ to a subset of N_R channels, by replacing the first **L** above with $\mathbf{L}_{\mathrm{NR}}$, those channels will contain all of the information present in the N_p principal components provided $\mathbf{L}_{\mathrm{NR}}$ has $\geq N_p$ positive singular values. The minimum criterion for this is that $N_R \geq N_p$ and in practice this criterion is usually sufficient.

1.1. Package overview

The supplied package is split into two parts:

- Create_EOFs is a stand-alone program to create the eigenvectors, **L**, from a set of spectra.
- RR_Filter is a subroutine that uses the eigenvectors produced by Create_EOFs to produce the reconstructed radiances from supplied spectra. RR_Filter also optionally returns the amplitudes of the principal components if desired. In addition, two further subroutines, Calculate_PCScores and PCScores2Spectrum are available that allow reconstructed radiances to be calculated from the intermediate principal component scores.

1.2. Units and Noise

The EOFs, **L**, used in this process are usually produced from radiance (rather than brightness temperature) spectra normalised by the expected (diagonal) noise. The

assumed noise is read into the Create_EOFs from a file and then written to the resulting eigenvector file for use by RR_Filter. If one does not wish to noise normalise, one may simply set the values in the assumed noise file to be all unity.

The radiance units used in the input spectra for Create_EOFs must, of course, be consistent with the units used in the noise file and also with the input spectra used by RR_Filter. If one wants to, say, convert IASI apodised radiances to IASI unapodised radiances or convert brightness temperatures to radiances, these transformations must be made before the radiances are presented to the packages.

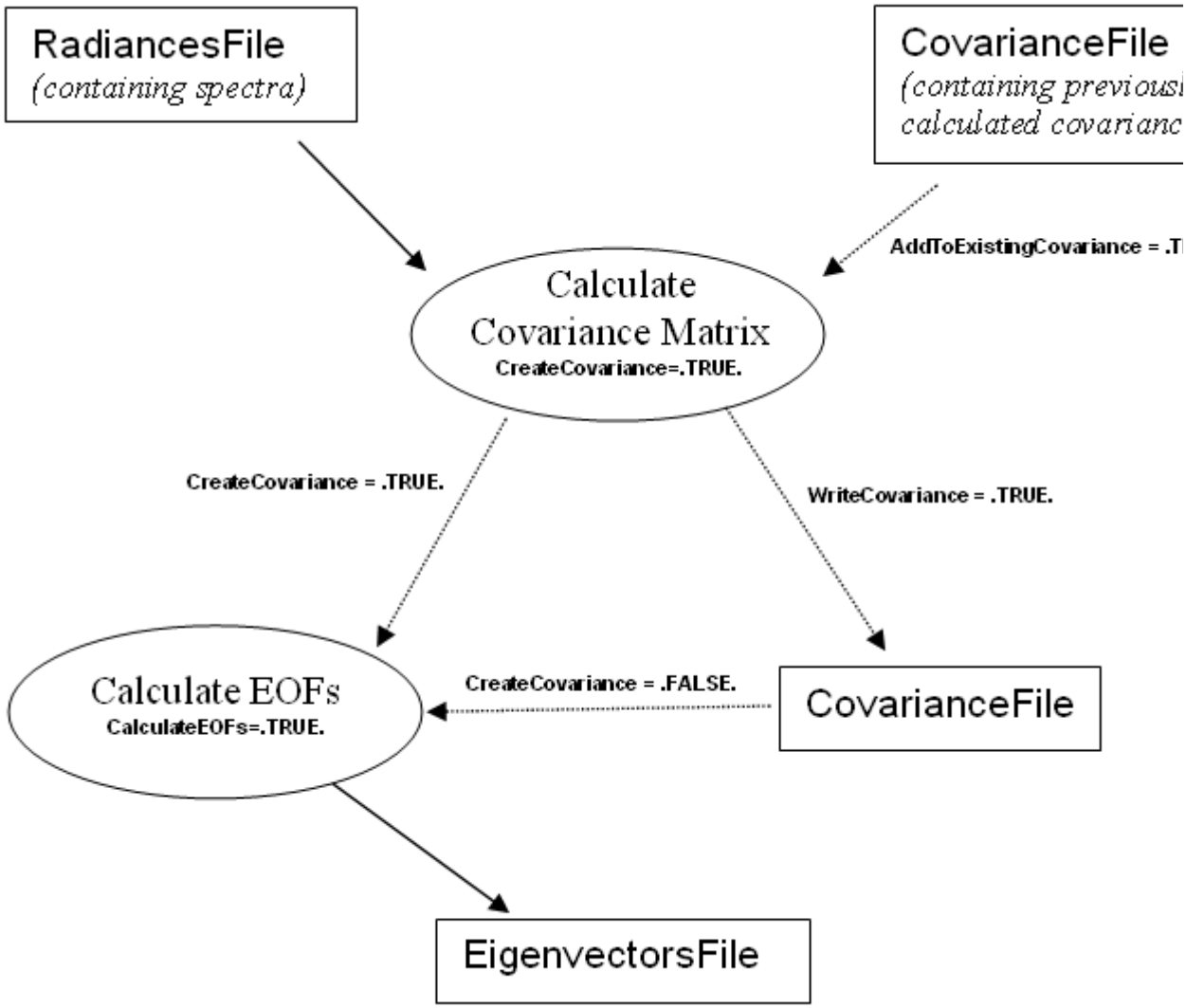
Two noise files are provided with the package: one derived before launch (IASI_NOISE_8461_PreLaunch.dat) and one from after the instrument became operational (IASI_NOISE_8461_Nov07.dat). The latter is used in the test scripts.

2. Create_EOFs

Create_EOFs is a stand-alone program which creates the eigenvectors used by RR_Filter. The eigenvectors are created from a covariance matrix which describes that variabilty of a training set of input spectra that are read in from a supplied file. These training spectra may be either simulated or observed spectra depending on what is required.

There are two strategies for computing EOFs from a number of spectra. The one used here (computation of a covariance matrix from which the EOFs are calculated) or direct calculation from a matrix of each of the spectra via Singlular Value Decomposition. The former has been chosen here to allow large numbers of spectra (many tens of thousands) to be used in the most efficient manner as the singular value decomposition method is limited by the typically available computer memory once tens of thousands of spectra are beig processed.

Create_EOFs will optionally produce the covariance matrix from input spectra; produce the EOFs from a pre-computed covariance matrix; or both. The data flow for the Create_EOFs program is illustrated below:



2.1. NAMELIST control

The program is controlled via a namelist file, Create_EOFs.NL. Each of the namelist variables has a default value hardcoded in the program. The namelist is called CreateEOFS and its variables are:

2.1.1. Switches:

- CreateCovariance (default=.TRUE.): create a covariance matrix (from which eigenvectors are to be derived) from provided spectra. If this is .FALSE. it is assumed that a file exists (with name provided by the CovarianceFile namelist variable) which contains this covariance.
- AddToExistingCovariance (default=.FALSE.): if .TRUE. it is assumed that CovarianceFile already exists and the current spectra are added to that file. CreateCovariance is forced to be .TRUE. if AddToExistingCovariance is .TRUE.
- WriteCovariance (default=.TRUE.): after the covariance is calculated write to the file CovarianceFile.
- AddNoise (default=.FALSE.): if .TRUE. random noise with standard deviation taken from the assumed noise (read in from the files provided by the NoiseFile

namelist variable) is added to the input spectra used to calculate the covariance matrix. This is useful if trying to simulate the use of real observations when only simulated spectra are available.

- CalculateEOFs (default=.TRUE.): calculate the eigenvalues from the calculated or read-in covariance matrix.
- UseExpert (default=.FALSE.): if .TRUE. the LAPACK "Expert" eigenvector finding algorithm is used, the default value is the LAPACK "Really Robust Algorithm". The differences between the performances of these algorithms has been found to be small. LAPACK generally recommends the latter, but some older implementations of LAPACK may only have the former.
- VerboseMode (default=.TRUE.): get some helpful output while the program is running.

2.1.2. Files:

Note that some of these files are described further in Section 2.2.

- RadiancesFile (Input file): The file containing the spectra to be processed. The number of channels per spectrum in this file must be MaxChans. This file is used if and only if CreateCovariance=.TRUE.
- ChannelFile (Input file): A file listing the channels to be included in the covariance and eigenvector calculations. The channel numbers are the positions of the required channels in the input spectra in the RadiancesFile. This file is used if and only if CreateCovariance=.TRUE.
- NoiseFile (Input file): The file containing the assumed noise. The number of elements in this file must match the number of channels in the RadiancesFile (i.e, MaxChans) and the units must be in agreement also. This file is used if and only if CreateCovariance=.TRUE.
- CovarianceFile (Input/Output file): This is a binary file containing the covariance matrix for the spectra in the training set. It is an input file if CreateCovariance=.FALSE. or AddToExistingCovariance=.TRUE., it is an output file if WriteCovariance=.TRUE. and CreateCovariance=.TRUE.
WARNING: The size of this file can be as large as 600Mb if all IASI channels are used.
- EigenvectorFile (Output file): This is an ASCII file containing the calculated eigenvectors along with the channel selection, assumed noise and mean noise-normalised radiances. It is produced if CalculateEOFs=.TRUE.
WARNING: The size of this file can be very large; e.g., 1000 eigenvectors for 8461 channels produces a 226Mb file.

2.1.3. Parameters:

- MaxChans, (default=8461): The number of channels in the input file RadiancesFile and in the noise file AddNoise.
- MaxSpec, (default=10000000): The largest number of spectra that may be read in at one time.
- NumEOFs, (default=1000): The number of EOFs to be calculated.

2.2 Files

2.2.1. Covariance File:

The covariance file is an unformatted binary file (to allow swifter input/output and because it is not expected that one might want to inspect it very often). It actually contains the number of spectra used, *n*; the sum of the radiances for each channel:

$$\overline{x}_i' = \sum_{n=1}^{N} x_{ni}$$

and the sum of the radiances product for each combination of channels:

$$\mathrm{Cov}'_{ij} = \sum_{n=1}^{N} x_{ni} x_{nj}$$

Here x_{ni} refers to the *i*th channel of the *n*th spectrum.

The true mean and covariance of the input spectra can then be calculated simply while allowing the covariance to be easily updated with additional spectra:

$$\overline{x}_i = \frac{1}{n} \overline{x}_i' \qquad \mathrm{Cov}_{ij} = \frac{1}{n} \mathrm{Cov}'_{ij} - \overline{x}_i \overline{x}_j$$

and .

Only the lower triangle of the covariance matrix is recorded as the matrix is symmetrical and the eigenvalue/eigenvector calculation routines only require this. The upper triangle simply contains zeroes.

The files supplied with this package are big-endian, so a suitable compiler flag should be used if the machine being used is little-endian.

2.2.2. Radiances File:

The input radiances file is also a binary, direct-access fortran file (and the supplied example is big-endian). It is possible that the user will want to change the form of the input file to match the particular form of the data available.

The supplied file contains simulated IASI radiances in $mW/m^2/sr/(cm^{-1})$ = $erg/s/cm^2/sr/(cm^{-1})$.

2.2.3. Eigenvectors File:

The eigenvectors file is stored as ASCII (with double-precision (8-byte) reals). It contains all of the information required to produce reconstructed radiances from an input spectrum. The entries in this file are as follows:

- Number of channels
- Indices of channels in the file [Number of channels]
- Assumed noise [Number of channels]
- Mean spectrum (in noise-normalised radiance units) [Number of channels]

- the number of eigenvectors
- The eigenvectors themselves, starting with the eigenvector corresponding to the largest eigenvalue - i.e., the component that explains most of the variance). [Number of channels * Number of eigenvectors]
- The eigenvalues corresponding to each of the eigenvectors in the file. This is for information only. [Number of eigenvectors]

This file is used by the RR_Filter, Calculate_PCScores and PCScores2Spectrum subroutines.

2.3. Compiling and Running the code:

The program is contained in a single file except for the LAPACK routines, so it may be compiled with one line provided LAPACK and BLAS libraries are available. e.g.,

```
pgf90 -Mbyteswapio -fast -o Create_EOFs Create_EOFs.f90 -llapack -
lblas
```

is the command to compile the code with a Portland Group Fortran 90 compiler run on a little-endian machine.

`-Mbyteswapio` is the flag used to swap from big- to little- endian. This, (or the equivalent for other machines) is needed if one wishes to use the binary data files (which are big-endian) supplied with this package and one is using a little-endian machine. If one is running on a little-endian machine and using a compiler that does not support byte-swapping, one may request a big-endian dataset from the NWPSAF at http://www.metoffice.gov.uk/research/interproj/nwpsaf/feedback.html

The program may then be run simply by executing Create_EOFs.

LAPACK (Anderson *et al.*, 2002) and BLAS (Lawson *et al.*, 1979; Dongarra *et al.*, 1988a, 1988b, 1990; Blackford *et al.*, 2002; Dongarra, 2002) are often installed by default on scientific computing systems but if not available already they may by installed from http://www.netlib.org/lapack and http://www.netlib.org/blas. Both are freely-available software packages which may be incorporated into commercial software packages.

Run time and memory requirements vary according to the number of channels being considered. For large numbers of channels, the dominant step in the eigenvector calculation is the conversion of the covariance matrix to tridiagonal form before deriving the eigenvectors. The CPU time for this step scales as the cube of the number of channels used. The following are guideline timings for 1000 and 8461 channels on an IBM Thinkcentre with a 3.40Ghz Intel Pentium 4 CPU:

	1000 Channels	8461 Channels
Covariance calculation: Time per 10000 spectra	1 minute	74 minutes
Eigenvector calculation (1000 EOFs)	10seconds	37 minutes
Eigenvector calculation (100 EOFs)	4 seconds	26 minutes
Memory usage	10Mb	548Mb

*The above timings exclude reading and writing of the covariance and eigenvectors files but include reading in the spectra from disk.

If one runs on a IBM Cluster 1600 supercomputer, the timings for the 8461 channels case are reduced to 22, 14 and 10 minutes.

Note that as the purpose of reconstructed radiances is to compress the information available in the *whole spectrum* into a subset of channels, the input channels should comprise a large fraction of the total spectrum. Reasons for excluding channels might be the poor quality of certain individual channels (e.g., the "popping" channels of AIRS) or excessive oversampling of the spectrum produced from an interferometer.

3. Producing Reconstructed Radiances from EOFs

The three subroutines presented here are used to produced filtered radiances from input unfiltered radiances. RR_Filter does this in one step while Calculate_PCScores and PCScores2Spectrum achieve the same result in two steps through the calculation of the principal component scores (which may be stored more efficiently that either the input or output spectra).

A test program, RR_Filter_Test, that calls these subroutines is supplied. This program may be compiled by modifying and executing the file *Make_RR_Filter_Test*. On running the resultant executable file, *RR_Filter_Test*, the test program will process 100 test spectra, reporting the QC index for each and outputting the ASCII file *RR_Filter_Test.out* which contains the means and standard deviations for the input and output spectra (for the 300 output channels specified in the program).

3.1. RR_Filter

RR_Filter is a subroutine that takes input spectra and outputs the spectrum reconstructed from the leading eigenvectors as calculated by Create_EOFs. The input spectrum must be in the same units as the spectra used to generate the EigenvectorsFile being used (in the examples provided these units are $mW/m^2/sr/(cm^{-1}) = erg/s/cm^2/sr/(cm^{-1})$).

The subroutine arguments are as follows:

Spectrum_In(NumChans_In)	REAL Array (IN)	The input spectrum
Chans_In(NumChans_In)	INTEGER Array (IN)	The channel numbers for the input spectrum. These channels must include *all* the channels in the Eigenvector file.
NumChans_In	INTEGER (IN)	The number of input channels
Num_EOFs	INTEGER (IN)	The number of EOFs to be used
Chans_Out(NumChans_Out)	INTEGER Array (IN)	The required channel numbers for the output spectrum
NumChans_Out	INTEGER (IN)	The required number of output

		channels
Fixed_Channels	LOGICAL (IN)	Set .TRUE. if and only if the input and output channels are fixed. It is recommended for reasons of efficiency that this is set to .TRUE. if at all possible.
Last_Call	LOGICAL (IN)	Set to .TRUE. to DEALLOCATE allocated arrays (the spectrum is not processed if LastCall=.TRUE.)
Spectrum_Out(NumChans_Out)	REAL Array (OUT)	The output spectrum
QC	REAL (OUT)	Quality control index (see below)
ErrorCode	INTEGER (OUT)	Subroutine error code (set to zero if the subroutine completed successfully)
PC_Scores(Num_EOFs)	REAL Array (OPTIONAL, OUT)	Amplitudes of Principal Components
ErrorMatrix (NumChans_Out, NumChans_Out)	REAL Array (OPTIONAL, OUT)	Estimated error covariance matrix (see below)

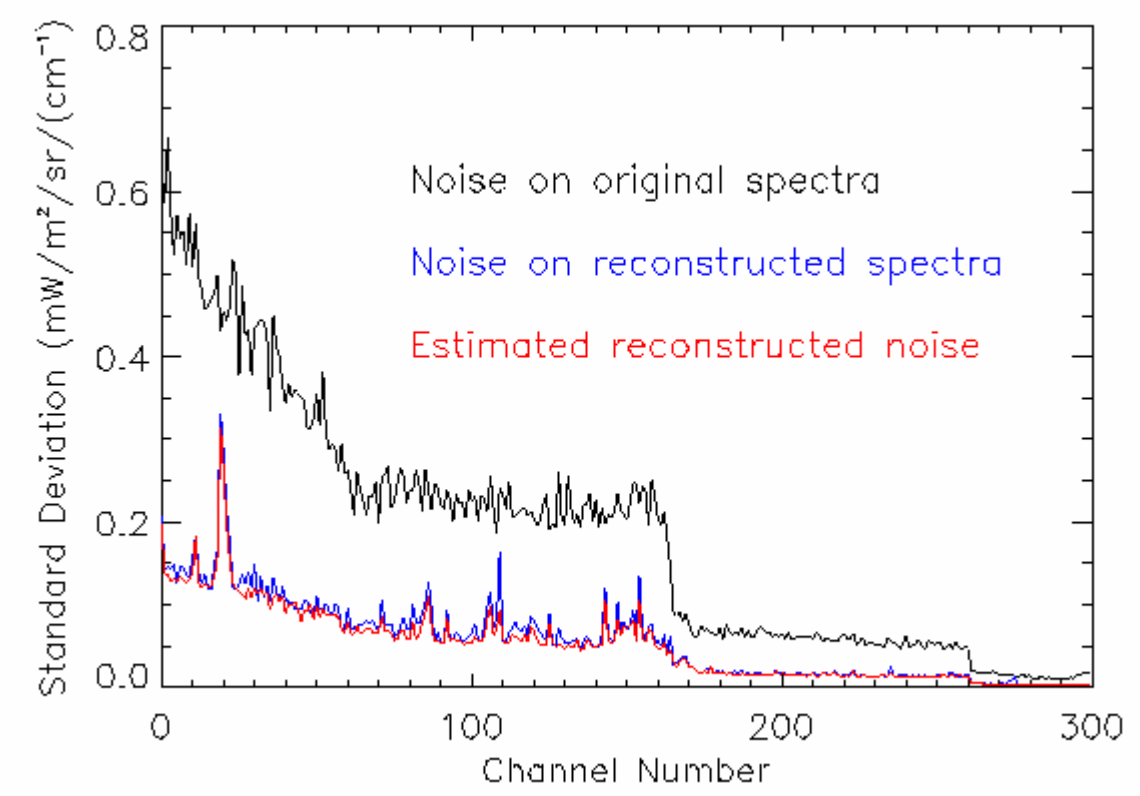
The QC index contains the RMS difference between the input and output spectrum. If the noise on the input spectrum and the assumed noise are similar, the QC values should average to somewhat less than unity. If the input spectra are noise-free (e.g., simulations), the QC values should be small (~0.1).

The estimated error covariance matrix, *ErrorMatrix*, calculates how the (assumed diagonal) instrument noise is transformed by the reconstruction process. It is calculated by: $\mathbf{R}_{RR} = \mathbf{L}_{NR}\mathbf{L}^{T}\mathbf{R}\mathbf{L}^{T}\mathbf{L}_{NR}$ where $\mathbf{R}$ and $\mathbf{R}_{RR}$ are the error covariance matrices before and after the reconstruction process and $\mathbf{L}_{NR}$ and $\mathbf{L}$ are the eigenvectors as defined in the introduction. As $\mathbf{L}$ is fixed, $\mathbf{R}_{RR}$ will not change as long as the channel choice is constant (so need only be evaluated once).

A namelist file, RR_Filter.NL, may be used to specify the Eigenvectors file being used. The namelist name is RRFilter with one variable: EigenvectorsFile. The default value for EigenvectorsFile is *Eigenvectors.out*.

A test program, RR_Filter_Test, that calls RR_Filter is supplied. This program may be compiled by modifying and executing the file *Make_RR_Filter_Test*. On running the resultant executable file, *RR_Filter_Test*, the test program will process 100 test spectra, reporting the QC index for each and outputting the ASCII file *RR_Filter_Test.out* which contains the means and standard deviations for the input and output spectra (for the 300 output channels specified in the program).

The following figure shows the standard deviations of the noisy-minus-true spectrum (black); the filtered-minus-true spectrum (blue); and the estimated noise on the filtered spectrum (red). Note that this example uses simulated data and also the test spectra are part of the training set.



3.2. Calculate_PCScores

Calculate_PCScores is similar to RR_Filter but calculates the amplitudes of the principal components of the input spectrum and does not produce filtered radiances. As with RR_Filter, the input spectrum must be in the same units as the spectra used to generate the EigenvectorsFile being used (in the examples provided these units are *mW/m^2/sr/(cm^{-1}) = erg/s/cm^2/sr/(cm^{-1})*).

The subroutine arguments are as follows:

Spectrum_In(NumChans_In)	REAL Array (IN)	The input spectrum
Chans_In(NumChans_In)	INTEGER Array (IN)	The channel numbers for the input spectrum. These channels must include *all* the channels in the Eigenvector file.
NumChans_In	INTEGER (IN)	The number of input channels
Num_EOFs	INTEGER (IN)	The number of EOFs to be used
Last_Call	LOGICAL (IN)	Set to .TRUE. to DEALLOCATE allocated arrays (the spectrum is

		not processed if LastCall=.TRUE.)
PC_Scores(Num_EOFs)	REAL Array (OUT)	Amplitudes of Principal Components
QC	REAL (OUT)	Quality control index (see below)
ErrorCode	INTEGER (OUT)	Subroutine error code (set to zero if the subroutine completed successfully)

The QC index contains the mean RMS difference per channel between the noise-normalised input spectrum and the same spectrum after filtering. If the noise on the input spectrum and the assumed noise are similar, the QC values should average to somewhat less than unity. If the input spectra are noise-free (e.g., simulations), the QC values should be small (~0.1). *Note that this QC value differs from that output by RR_Filter as the latter's QC is calculated from the output channels only.*

No estimated error matrix for the principal component scores is given as this will always be the identity matrix.

A namelist file, Calculate_PCScores.NL, may be used to specify the Eigenvectors file being used. The namelist name is RRFilter with one variable: EigenvectorsFile. The default value for EigenvectorsFile is *Eigenvectors.out*.

3.3. PCScores2Spectrum

PCScores2Spectrum is a subroutine that takes principal component amplitudes for a spectrum (as calculated by Calculate_PCScores or, optionally, RR_Filter) and outputs the spectrum reconstructed from the leading eigenvectors (as calculated by Create_EOFs). The output spectrum is in the same units as the spectra used to generate the EigenvectorsFile being used (in the examples provided these units are $mW/m^2/sr/(cm^{-1}) = erg/s/cm^2/sr/(cm^{-1})$).

The subroutine arguments are as follows:

PC_Scores(Num_EOFs)	REAL Array (IN)	Amplitudes of Principal Components
Num_EOFs	INTEGER (IN)	The number of EOFs to be used
Chans_Out(NumChans_Out)	INTEGER Array (IN)	The required channel numbers for the output spectrum
NumChans_Out	INTEGER (IN)	The required number of output channels
Fixed_Channels	LOGICAL (IN)	Set .TRUE. if and only if the input and output channels are fixed. It is recommended for reasons of efficiency that this is set to .TRUE. if at all possible.

Last_Call	LOGICAL (IN)	Set to .TRUE. to DEALLOCATE allocated arrays (the spectrum is not processed if LastCall=.TRUE.)
Spectrum_Out(NumChans_Out)	REAL Array (OUT)	The output spectrum
ErrorCode	INTEGER (OUT)	Subroutine error code (set to zero if the subroutine completed successfully)
ErrorMatrix (NumChans_Out, NumChans_Out)	REAL Array (OPTIONAL, OUT)	Estimated error covariance matrix (see below)

No QC index is produced by this subroutine as there is not un-filtered spectrum provided to do this calculation. One should use the QC provided by the program that produced the principal component scores.

The estimated error covariance matrix, *ErrorMatrix*, is calculated in an identical manner to that of RR_Filter_Test.

A namelist file, RR_Filter.NL, may be used to specify the Eigenvectors file being used. The namelist name is RRFilter with one variable: EigenvectorsFile. The default value for EigenvectorsFile is *Eigenvectors.out*.

A namelist file, PCScores2Spectrum.NL, may be used to specify the Eigenvectors file being used. The namelist name is RRFilter with one variable: EigenvectorsFile. The default value for EigenvectorsFile is *Eigenvectors.out*.

4. Test Scripts

Two test scripts are provided for testing. SetUp_Test_Quick.sh uses a reduced number of channels, spectra and eigenvectors and runs in about ten seconds. SetUp_Test.sh is a more comprehensive test and takes 2-3 hours.

4.1. SetUp_Test_Quick.sh

This section describes the operation of the SetUp_Test_Quick.sh script including a description of the input and output files.

The script compiles the programs, reads in spectra to produce a covariance matrix, and then computes the leading eigenvectors. The script should be executed from the directory in which it resides.

Two "Make" scripts are executed (they are simply shell scripts rather than proper Make files) to compile Create_EOFs (Make_CreateEOFS) and RR_Filter_Test_Quick (Make_RR_Filter_Test_Quick).

A covariance matrix this then constructed from input spectra. This is done through the Create_EOFs program which is controlled by the Create_EOFs_Test_Quick.NL namelist, reproduced below:

```
&CreateEOFs
RadiancesFile = 'data_quick/RADIANCES_100Subset.dat'
CovarianceFile = 'data_quick/Cov_Test.out'
ChannelFile = 'Chans2Use_Test.dat'
MaxSpec=100
AddNoise=F
CreateCovariance=T
WriteCovariance=T
CalculateEOFs=F /
```

Therefore this program reads in 100 spectra from the data_quick/RADIANCES_100Subset.dat file (which contains 10000 simulated IASI spectra in binary format). A covariance martrix is calculated and written to file (the binary file data_quick/Cov_Test.out). EOFs are *not* calculated.

The Create_EOFs program is now run a second time with the Create_EOFs_Test2_Quick.NL namelist:

```
&CreateEOFs
RadiancesFile = 'data_quick/RADIANCES_100Subsetx.dat'
CovarianceFile = 'data_quick/Cov_Test.out'
EigenvectorsFile = 'data_quick/Eigenvectors_Test.out'
ChannelFile = 'Chans2Use_Test.dat'
NumEOFs=100
MaxSpec=100
AddNoise=F
AddToExistingCovariance=T
CreateCovariance=T
WriteCovariance=T
CalculateEOFs=T /
```

Now 100 spectra are read in from a second input binary file, data_quick/RADIANCES_100Subsetx.dat and added to the covariance matrix calculated above. The matrix is once again written to disk but this time the leading 100 EOFs are also calculated for the first 1000 channels (defined in the Chans2Use_Test.dat file).and are written to the ASCII file data/Eigenvectors_Test.out. The contents of the eigenvector file are described in Section 2.2.3. This file may be compared to Test_Runs/Eigenvectors_Test_Quick.out.gz.

The final stage of SetUp_Test_Quick.sh is the generation of filtered radiances through the test program RR_Filter_Test_Quick. The control namelist for this program, RR_Filter_Test_Quick.NL only contains the name of the required eigenvector file, viz:

```
&RRFilter
EigenvectorsFile = 'data_quick/Eigenvectors_Test.out' /
```

This program reads in radiances from two binary files: data/RADIANCES_100_Noisy.dat and data/RADIANCES_100.dat. The former contains calculated IASI radiances to which random noise has been added. The latter contains the same radiances without the noise and is used as "truth" when calculating noise statistics.

The program calculates the noise-filtered radiances from the first 100 spectra in the noisy radiance file. The noise filtered radiances for a subset of 136 channels are calculated. These parameters being hard-coded in the test program. On generating the filtered radiances, a quality control value is returned to standard out for each spectrum, plus its mean value for all spectra. The statistics of the filtering process are output into the file RR_Filter_Test_Quick.out as follows:

- Number of channels
- Mean difference between the input noisy spectra and the noise-free spectra
- The standard deviation of the difference between the input noisy spectra and the noise-free spectra (this should approximate the instrument noise)
- Mean difference between the filtered spectra and the noise-free spectra.
- The standard deviation of the difference between the filtered spectra and the noise-free spectra. (This should be a fraction of the instrument noise).

This file may be compared to Test_Runs/RR_Filter_Test_Quick.out. Note that results will not be bit identical to the output in Test_Runs/ as floating point calculations depend in detail on the machine, compiler and compiler options used. Agreement in the reconstructed radiances should be at least 5 sig.figs. and there should be similar accuracy in the leading eigenvectors.

4.2. SetUp_Test.sh

The SetUp_Test.sh is very similar to SetUp_Test_Quick.sh except that all channels and 10000 spectra are used in each of the two Create_EOFs runs; 500 EOFs are calculated in the second of these runs and 300 channels are output from RR_Filter_Test. The output from the latter is RR_Filter_Test.out and may be compared to Test_Runs/RR_Filter_Test.out. The eigenvector file for comparison is at Test_Runs/Eigenvectors_Test.sav which is a link to a file in the data for disk usage reasons.

5. Troubleshooting

Memory fault:
A memory fault can arise if the called and calling subroutine do not have matching argument types. To prevent this occuring interface blocks have been used throughout. This is particularly important for RR_Filter where the called subroutine has an optional argument.

Problems are encountered reading binary files:
If problems are encoutered reading binary files, note that the files supplied with this package are big-endian, so a suitable compiler flag should be used if the machine being used is little-endian.

Also, it is assumed that when opening a direct access file (such as the radiances file) the record length "RECL" is given in units of bytes. Some compilers use 4-byte units by default, so you will either have to modify the code or use an appropriate compiler directive (e.g. for ifort you can use the directive "-assume byterecl").

Test runs are not bit identical to supplied results:
Results will not be bit identical to the output in Test_Runs/ as floating point calculations depend in detail on the machine, compiler and compiler options used. Agreement in the reconstructed radiances should be at least 5 sig.figs. and there should be similar accuracy in the leading eigenvectors.

Package is being run on a little-endian machine and compiler lacks a byte-swapping option: If one is running on a little-endian machine and using a compiler that does not support byte-swapping, one may request a big-endian dataset from the NWPSAF at http://www.metoffice.gov.uk/research/interproj/nwpsaf/feedback.html

You can contact the NWPSAF team using the form at:
http://www.metoffice.gov.uk/research/interproj/nwpsaf/feedback.html

References

Anderson, E. and Bai, Z. and Bischof, C. and Blackford, S. and Demmel, J. and Dongarra, J. and Du Croz, J. and Greenbaum, A. and Hammarling, S. and McKenney, A. and Sorensen, D., LAPACK Users' Guide, Third Edition, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1999, ISBN: 0-89871-447-8 (paperback)

P. Antonelli, H.E. Revercomb, L.A. Sromovsky, W.L. Smith, R.O. Knuteson, D.C. Tobin, R.K. Garcia, H.B. Howell, H.-L. Huang, and F.A. Best (2004). A principal component noise filter for high spectral resolution infrared measurements, *J. Geophys. Res.*, **109**, D23102-23124.

L. S. Blackford, J. Demmel, J. Dongarra, I. Duff, S. Hammarling, G. Henry, M. Heroux, L. Kaufman, A. Lumsdaine, A. Petitet, R. Pozo, K. Remington, R. C. Whaley, *An Updated Set of Basic Linear Algebra Subprograms (BLAS)*, ACM Trans. Math. Soft., 28-2 (2002), pp. 135-151.

J. J. Dongarra, J. Du Croz, S. Hammarling, and R. J. Hanson, *An extended set of FORTRAN Basic Linear Algebra Subprograms*, ACM Trans. Math. Soft., 14 (1988a), pp. 1-17.

J. J. Dongarra, J. Du Croz, S. Hammarling, and R. J. Hanson, *Algorithm 656: An extended set of FORTRAN Basic Linear Algebra Subprograms*, ACM Trans. Math. Soft., 14 (1988b), pp. 18-32.

J. J. Dongarra, J. Du Croz, I. S. Duff, and S. Hammarling, *A set of Level 3 Basic Linear Algebra Subprograms*, ACM Trans. Math. Soft., 16 (1990), pp. 1-17.

J. J. Dongarra, J. Du Croz, I. S. Duff, and S. Hammarling, *Algorithm 679: A set of Level 3 Basic Linear Algebra Subprograms*, ACM Trans. Math. Soft., 16 (1990), pp. 18-28.

J. Dongarra, *Basic Linear Algebra Subprograms Technical Forum Standard*, International Journal of High Performance Applications and Supercomputing, 16(1) (2002), pp. 1-111, and International Journal of High Performance Applications and Supercomputing, 16(2) (2002), pp. 115-199.

C. L. Lawson, R. J. Hanson, D. Kincaid, and F. T. Krogh, *Basic Linear Algebra Subprograms for FORTRAN usage*, ACM Trans. Math. Soft., 5 (1979), pp. 308-323.