Hector user manual
version 1.4

Machiel Bos, Rui Fernandes and Luıisa Bastos

April 18, 2015

Contents

1 Introduction

Hector is a software package that can be used to estimate the linear trend in time-series with temporal corelated noise. Trend estimation is a common task in geophysical research where one is interested in phenomena such as the increase in temperature, sea level or GNSS derived station position over time. It is well known that in most geophysical time-series the noise is correlated in time (Agnew, 1992; Beran, 1992) and this has a significant influence on the accuracy by which the linear trend can be estimated. Therefore, the use of a computer program such as Hector is advisable.

Hector assumes that the user knows what type of temporal correlated noise exists in the observations and estimates both the linear trend *and* the parameters of the chosen noise model using the Maximum Likelihood Estimation (MLE) method. Since for most observations the choice of noise model can be found from literature or by looking at the power spectral density, this is sufficient in most cases.

Instead of using Hector, one can also use the CATS software of Williams (2008) which is a good and popular choice. In fact, Hector was written from scratch in C++ with the objective to have a faster CATS. The reason is Hector is faster is that it accepts only stationary noise, with constant noise properties, and this allows the use of fast matrices operations. The obtained reduction in computation time is a factor of 10-100 compared to CATS, see Bos et al. (2012). On the other hand, CATS has the advantage that it offers a wider range of noise models. The choice is up to the user!

If the observations are non-stationary, then stationarity can be obtained by taking the first difference of the data or using an approximation of the noise model as explained by Bos et al. (2008, 2012) where also more information on the theoretical background and more references can be found.

This manual starts in section 2 by explaining how to install Hector on your computer. In section 3 a tutorial on using Hector is presented using two examples of analysing synthetic GPS data with offsets and outliers and a real GPS data set. This is followed by section 4 and 5 on acceptable data formats and implemented noise models respectively. Finally, a quick reference of the parameters in the control files are presented in 6.

1.1 How to cite Hector

If you find the Hector program useful, please cite it in your work as:

> Bos, M. S., Fernandes, R. M. S., Williams, S. D. P., and Bastos, L. (2012). Fast Error Analysis of Continuous GNSS Observations with Missing Data.*J. Geod.*, doi:10.1007/s00190-012-0605-0.

1.2 Main features

The main features of Hector are:

1. Correctly deals with missing data. No interpolation or zero padding of the data nor an approximation of the covariance matrix is required (as long the noise is, or has been made, stationary).
2. Allows yearly, half-yearly and other periodic signals to be included in the estimation process of the linear trend.
3. Allows the option to estimate offsets at given time epochs.
4. Includes power-law noise, ARFIMA, generalized Gauss-Markov and white noise models. Any combination of these models can be made.
5. Allows taking the first difference of the data if power-law noise model is chosen (including combination of white, flicker and random walk).
6. Comes with programs to remove outliers, to make power spectral density plots and to create files with synthetic coloured noise.

2 Installation

The hector software package is mainly intended to be run on computers with Unix-like operating systems. For the Linux operating system, a convenient debian binary package can be downloaded from `http://segal.ubi.pt/hector`. Hector requires the ATLAS (`http://math-atlas.sourceforge.net/`), FFTW3 (`http://www.fftw.org/`) and the GSL (`http://www.gnu.org/software/gsl/`) libraries and these should be installed first. On a computer with the Ubuntu linux distribution one can simply type:

```
sudo apt-get install libfftw3-3 libatlas3gf-base libgsl0ldbl
```

The installation procedure of hector on a 64-bit computer is now:

```
sudo dpkg -i hector_1.1_amd64.deb
```

This will put the binaries in /usr/bin, the documentation (including this manual) in /usr/share/doc/hector and the examples in /usr/share/hector/examples. The man pages of the binaries are stored in /usr/share/man/man1.

For 32-bit computers one should replace the debian package by hector_1.1_i386.deb. To remove the 64 bit hector package, type:

Table 1: List of programs provided by the Hector software package.

Name	Description
estimatetrend	Main program to estimate a linear trend.
estimatespectrum	Program to estimate the power spectral density from the data or residuals using the Welch periodogram method.
modelspectrum	Given a noise model and values of the noise parameters, this program computed the associated power spectral density for given frequency range.
removeoutliers	Program to remove outliers from the data.
simulatenoise	Program to files with synthetic coloured noise.
date2mjd	Small program to convert calender date into Modified Julian Date.
mjd2date	The inverse of date2MJD.

```
sudo dpkg -r hector_1.1_amd64.deb
```

If this installation went successful, then one can skip the rest of this section and try out the examples. For other systems the source code of Hector needs to be downloaded and compiled using a C++ compiler such as g++. This will also require the installation of some extra development libraries to obtain the header files:

```
sudo apt-get install libfftw3-dev libatlas-dev libatlas-base-dev libgsl0-dev
```

If these extra three libraries have been installed in their default directories, then one can compile the Hector source code by going to the `hector-1.1/` directory and type:

```
make
sudo make install
```

The programs, listed in Table 1, will by default be installed in the `/usr/bin` directory. If another directory for the binaries is required, then the variable PREFIX in Makefile.inc needs to be changed.

To run the examples one also need the `Tcl` scripting language and the `gnuplot` plotting program. Again, on an Ubuntu machine one can type:

```
sudo apt-get install pgplot tcl
```

3 Tutorial

By going step by step through the analysis of some example data sets the working of Hector will be explained. First, we look at some synthetic GPS data.

3.1 Example 1: Synthetic GPS time-series with spikes and offsets

In the directory `examples/ex1` a data file called TEST.enu is stored which represents some fictional GPS position data set. The file extension ‘enu’ stands for East-North-Up and these three components are stored in the 2^{nd}, 3^{rd} and 4^{th} column respectively. The first column contain the Modified Julian Date (MJD) which is convenient format to make plots. These data are stored in a simple ASCII text file and can be inspected by any normal text editor. When doing so, one will detect some additional header lines:

```
# sampling period 1.0
# offset 50284.0 0
# offset 50334.0 0
# offset 50334.0 1
# offset 50784.0 0
# offset 51034.0 0
```

The first line just tells the program that the sampling period of the data is daily (T=1 day). Furthermore, there are offsets at the MJD epochs 50284, 50334, 50784 and 51034. The last 0 indicates that these only occur in the East component (0=East, 1=North, 2=Up). If an offset occurs in more than one component, the header line for the offset is repeated such as is the case for MJD=50334. Hector cannot find the time of offset, these have to be provided by the user.

This information can also be provided in an another file instead of in the header of the file with the observations. In this case one must specify the keyword 'OffsetFile' and its name in the control file. One must also not forget to specify the component in the control file. Since most people prefer simple day-month-year format, the format for the offsets in the external file is a bit different. Using the offset data shown before, we have:

```
 20- 7-1996   NaN   0   0
  8- 9-1996   NaN NaN   0
  2-12-1997   NaN   0   0
  9- 8-1998   NaN   0   0
```

After these header lines, if not given in an external file, the data is given:

```
50084.0 -17.88951 -21.47271 -19.38038
50085.0 -16.88599 -20.40868 -18.27968
50086.0 -16.84916 -20.31029 -18.14525
...
```

As explained before, each row contains the time (MJD), East, North and Up component. The east component (column 2) can be plotted in gnuplot using the command:

```
plot 'TEST.enu' using (($1-51544)/365.25+2000):2 with lines
```

The time values are converted from MJD to years. The result is shown in Figure 1 where one can detect the offsets and the presence of outliers.

3.1.1 Removal of outliers

To remove the outliers we need to run `removeoutliers` which requires a control file called `removeoutliers.ctl`. Hector uses various control files which are simple text files and the rows with the keywords can occur in any order. If Hector cannot find a keyword, then it will complain unless the keyword is optional. If the keyword optional and omitted then its default value will be used. Another control file can be specified on the command line. For example:

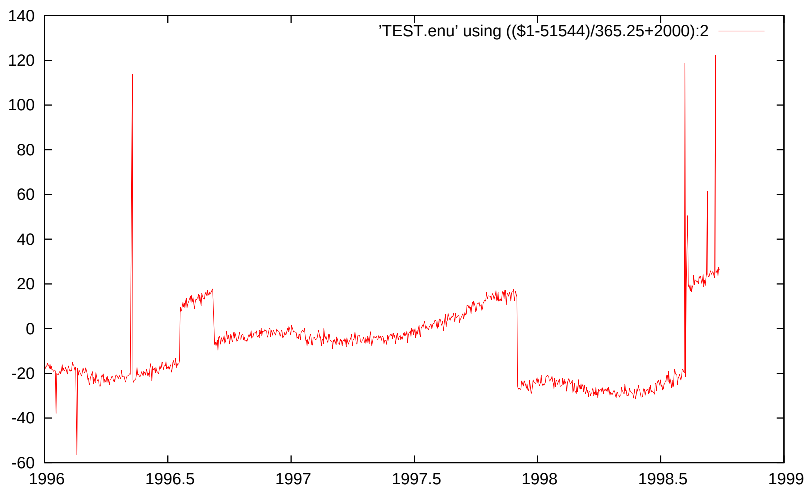

Figure 1: The raw data of the East component of TEST.enu.

```
removeoutliers othercontrolfile.ctl
```

The contents of `removeoutliers.ctl` in the `examples\ex1` directory is:

```
DataFile            TEST.enu
DataDirectory       ./
OutputFile          TEST_pre.mom
component           East
interpolate         no
firstdifference     no
seasonalsignal      yes
halfseasonalsignal  no
periodicsignals
estimateoffsets     yes
IQ_factor           3.0
PhysicalUnit        mm
ScaleFactor         1.0
```

The first line gives the name of the DataFile with the raw data which is TEST.enu in our case. The second line gives the directory where this file can be found and the third live contains the required name of the file with the preprocessed data (outliers removed): TEST_pre.mom. The output will always be in mom-format which stands for MJD, Observations, Model. The last column is optional and since `removeoutliers` only replaces the raw observations with the preprocessed observations, no third column will be added. See section 4 for more details on the acceptable data format.

`removeoutliers` fits a linear trend to the raw data using ordinary least-sqares and afterwards subtracts this linear trend from the observations to create residuals. These residuals are ordered by size and the interquartile range is computed (this is the value of the residual at 75% of the sorted array minus the value of the residual at 25% of the sorted array). Any residual with a value less than 3 times this interquartile range below

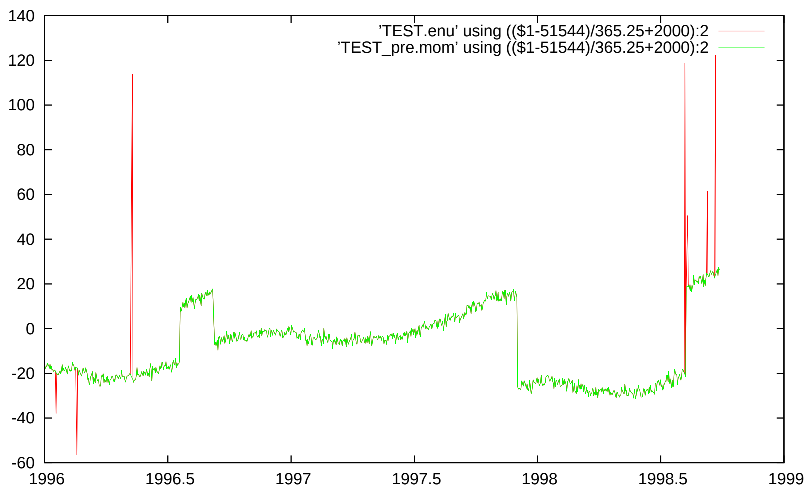

Figure 2: The raw and preprocessed data of the East component stored in TEST.enu and TEST_pre.mom.

or above the median is considered to be an outlier (Langbein and Bock, 2004). This factor of 3 is set by the keyword IQ_factor and can be changed by the user.

In this control file one must also give the physical unit of the data. This information is not essential but reminds the user to think about the unit of the data and if some scaling is required. Such scaling is set by the keyword 'ScaleFactor' which is 1.0 in this case. This keyword is optional and if omitted then a default value of 1.0 will be assumed.

The linear trend is estimated assuming a white noise model and, as can be seen from the `removeoutliers.ctl` file, a seasonal (i.e. yearly) signal is also included in the estimation process. Offsets are also estimated. On the other hand, no half-seasonal signal is estimated nor any other periodic signal and the missing data is not interpolated. The keyword 'periodicsignals' is optional and can be omitted. In principle one could take the first difference of the observations but this has not been done.

As was mentioned before, the information about the time of offsets can also be specified in another file instead of being listed in the header of the data file. In this case the keyword 'OffsetFile' must be set and the name of the file must be given after it.

The results of `removeoutliers`, stored in TEST_pre.mom, are shown in Figure 2 which have been generated with `gnuplot` using the command:

```
plot 'TEST.enu' using (($1-51544)/365.25+2000):2 with lines,\
     'TEST_pre.mom' using (($1-51544)/365.25+2000):2 with lines
```

3.1.2 Estimation of the linear trend

Now that the outliers have been removed, we can estimate the linear trend. The parameters that control this analysis are by default given in the file `estimatetrend.ctl`. As before, another name for the control file can be specified on the command line. The contents of `estimatetrend.ctl` is:

```
DataFile            TEST_pre.mom
DataDirectory       ./
OutputFile          TEST_out.mom
interpolate         no
firstdifference     no
seasonalsignal      yes
halfseasonalsignal  no
periodicsignals
estimateoffsets     yes
NoiseModels         Powerlaw White
LikelihoodMethod    AmmarGrag
PhysicalUnit        mm
ScaleFactor         1.0
```

Again, there is the keyword DataFile which should be given by the name of the input file which is TEST_pre.mom in this case because we are now going to use the preprocessed observations. These preprocessed observations together with the estimated trend in the third column, are written to the file name given after the keyword OutputFile. As before, the data is not interpolated and the first difference of the data is not applied. However, a seasonal signal and offsets are estimated. The chosen noise models are a combination of power-law noise plus white noise. Alternatively we could have chosen from: Flicker, RandomWalk, ARMA, ARFIMA or GGM (Generalized Gauss Markov) for the noise models.

The numerical method for finding the maximum likelihood value is the Nelder & Mead, also called Simplex, method. At the moment no other method has been implemented.

The chosen method for the Likelihood computation is 'AmmarGrag' which is explained in more detail in Bos et al. (2012). The keyword LikelihoodMethod is optional and can be omitted. If it is omitted, then the default method is 'AmmarGrag' is the percentage of missing data is less than 50% of the total observation period. Otherwise the Full Covariance matrix ('FullCov') method is used.

Note that if the ScaleFactor is not 1 in the file `removeoutliers.ctl`, then you probably want to set it to 1 in `estimatetrend.ctl` to avoid applying the scaling twice. Again, this keyword is optional and a default value of 1 is assumed when this keyword is not provided. As was mentioned before, the information of the offsets can be given in another file by using the keyword 'OffsetFile' and specifying the 'component' keyword.

The program `estimatetrend` shows the following on the screen:

```
************************************
    estimatetrend, version 1.2
************************************
Data format: MJD, Observations, Model
Filename             : ./TEST_pre.mom
Number of observations: 1000
Percentage of gaps    : 10.7

----------------
```

```
  AmmarGrag
----------------

Number of CPU's used (threads) = 4

    1    0.25000     0.17500   f()= 1736.224211  size=0.146
...
   41    0.51029     0.38316   f()= 1724.982836  size=0.000
converged to minimum at
   42    0.51033     0.38314   f()= 1724.982836  size=0.000

Likelihood value
--------------------
min log(L)=-1724.983
AIC       =3453.966
BIC       =3463.555

Powerlaw: fraction=0.490
d = 0.3831 +/- 0.0868

White: fraction=0.5103
No noise parameters to show
STD of the innovation noise: 1.6190
bias : 0.831535 +/- 0.994658 mm (at MJD=50583.500000)
trend: 16.393336 +/- 0.687413 mm/year
cos yearly : 4.090394 +/- 0.280224 mm
sin yearly : -3.936342 +/- 0.285006 mm
offset at 50284.0000 : 24.493855 +/- 0.687678 mm
offset at 50334.0000 : -24.686144 +/- 0.750109 mm
offset at 50784.0000 : -39.755662 +/- 0.702739 mm
offset at 51034.0000 : 38.529800 +/- 0.724268 mm
--> TEST_out.mom
Total computing time: 4.00000 sec
```

The first lines are self explanatory. The number of CPU's used is also shown to make sure that Multi-Threading on Multi-Core Processors is working. In this case 4 CPU's are used. The next few lines show the minimization steps of the negative value of the natural logaritm of the likelihood (which is thus maximised!).

Nowadays the quality of the chosen noise models in describing the noise in the data is evaluated using the Akaike Information Criteria (AIC) and the Baysian Information Criteria (BIC). These values are shown below the value of the natural logarithm of the Likelihood value.

Since we are using white and power-law noise, two noise parameters need to be estimated. The first parameter (ϕ), determines the distribution of white and power-law noise ($\phi = 1$ represents 100% white noise, $\phi = 0$ represents 100% power-law noise). The second noise parameter is the slope of the power-law noise d. This is equal to 2α and -2μ which is found in other papers on time-series analysis of GPS time-series. The values of these two parameters need to be determined using the numerical minisation scheme and their values during each step (42 steps are needed before convergence has been reached) are shown in the output.

For white noise there is no additional parameter to estimate so, that is why there is this line "No noise parameters to show" in the output in the white noise section. More details on the noise models are given in section 5.

The rest of the lines show the estimated values of the model such as nominal bias (also known at intercept at t_0 and which is equal to the estimated value at t_0), linear trend and a seasonal signal. To get the most accurate estimate of the linear trend, the linear trend in the design matrix has a zero mean. To explain this better, assume that we have 5 observations. The design matrix $\mathbf{H}$ looks like:

$$\mathbf{H} = \begin{pmatrix} 1 & -2 \\ 1 & -1 \\ 1 & 0 \\ 1 & 1 \\ 1 & 2 \end{pmatrix} \tag{1}$$

The first column will estimate the nominal bias, the second the linear trend. The two columns are orthogonal since $\mathbf{H}^T\mathbf{H}$ produces a diagonal matrix. Thus, the estimation of the nominal bias is not influenced by the estimation of the linear trend which is beneficial for the accuracy. It also means that the nominal bias corresponds to the value of the model at the time at row 3 (half of the time-series). hector notes this time.

If another reference epoch for the nominal bias is required, then this date can be provided after the keyword 'ReferenceEpoch', using year, month and day. For example, a reference epoch of 1 January 2008 is given by:

```
ReferenceEpoch          2008 1 1
```

See section 3.2.1 for a more detailed example. Also shown in the output are the values of the estimated offsets. The results of `estimatetrend`, stored in TEST_out.mom, are shown in Figure 3 which have been generated in `gnuplot` with the command:

```
plot 'TEST.enu' using (($1-51544)/365.25+2000):2 with lines,\
     'TEST_out.mom' using (($1-51544)/365.25+2000):2 with lines,\
     'TEST_out.mom' using (($1-51544)/365.25+2000):3 with lines
```

3.1.3 Plotting the power spectral density

We have used a power-law plus white noise model in our estimation process. To verify if this is correct, it is good to make a power spectral density (PSD) plot of the residuals (i.e. the difference between observations minus the estimated linear trend and additional offsets and periodic signals). This can be done using the program `estimatespectrum` which computes a Welch periodogram, stored in the file `estimatespectrum.out`. As usual, the behaviour of this program is controlled by the file `estimatespectrum.ctl`:

```
DataFile            TEST_out.mom
DataDirectory       ./
```

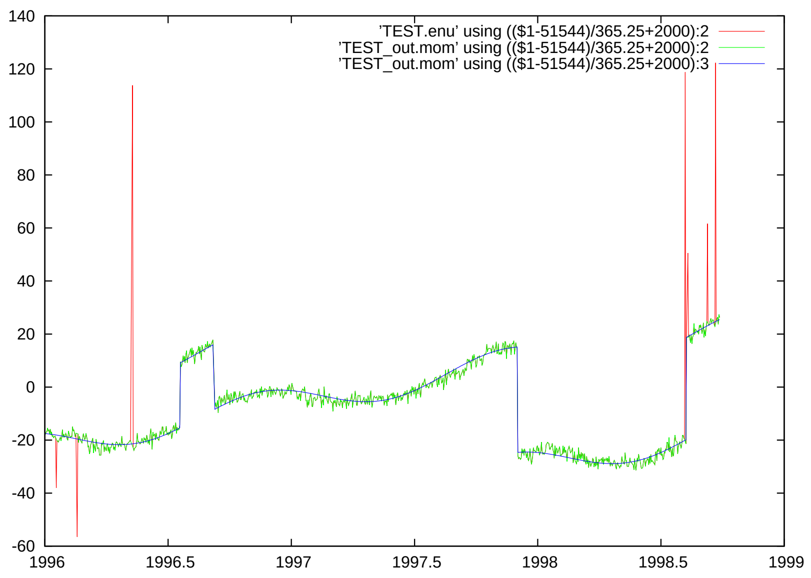

Figure 3: The raw, filtered data and the estimated model of the East component stored in TEST.enu and TEST_out.mom.

```
interpolate         no
firstdifference     no
ScaleFactor         1.0
```

The contents of `estimatespectrum.out` is shown in Figure 4. By default the time-series is devided into 4 parts by `estimatespectrum`. Since 50% overlap is used, there are 7 segments in total and in this case the length of each segment is 250 data points. The first and last 10% of each segment is smoothed to zero using a Parzen window function. If more segments are required, to get a better averaging of the periodograms but at the cost of a smaller frequency range, one can specify the number of divisions of the data on the command line. For example:

```
estimatespectrum 8
```

which will divided the time-series into 8 pieces, creating 15 segments due to the 50% overlap used. The area underneath the (one-sided) power spectral density plot should be equal to the variance of the time-series (Buttkus, 2000). This area underneath the PSD plot has been computed by simply assuming that each point represents a bar of width $1/\Delta t$ and adding them all up. Next, it is important to note the begin and end value of the frequency range.

```
----------------------
   EstimateSpectrum
----------------------

Data format: MJD, Observations, Model
Filename             : ./TEST_out.mom
```

```
Number of observations: 1000
Percentage of gaps    : 10.7
Number of data points n : 1000
Number of data used   N : 1000
Number of segments    K : 7
Length of segments    L : 250
Total variance in signal (time domain): 3.297
Total variance in signal (spectrum)   : 3.123
freq0: 4.6296e-08
freq1: 5.7870e-06
```

The PSD of our estimated noise model can be computed using the program `modelspectrum` which has no control file and which saves its output in `modelspectrum.out`. However, note that it gets information on the required set of noise models from the file `estimatetrend.ctl`. Normally one makes a PSD after estimating the trend so this should not be an inconvenience. The user must manually enter the requested values for the noise parameters and provide the begin and end value of the frequency range. For our example, the input looks like:

```
---------------
ModelSpectrum
---------------

Enter the standard deviation of the innovation noise: 1.6190
Enter the sampling period in hours: 24
Enter fraction for model Powerlaw: 0.490
Enter fraction for model White: 0.5103

Powerlaw:
Enter value of fractional difference d:0.3831

White:
1) Linear or 2) logarithmic scaling of frequency?: 2
Enter freq0 and freq1: 4.6296e-08 5.7870e-06
```

The contents of `estimatespectrum.out` and `modelspectrum.out` are plotted in Figure 4.

3.1.4 Some additional tests

So far we have explained how to remove the outliers in the data, estimate the linear trend and how to make a PSD plot of the residuals. Before ending this section, we would like to show how some parameters can be changed. First, let us show the advantage of using the method of Bos et al. (2012) in terms of computation speed with respect to the standard method which is also implemented in Hector. To use the standard method change the likelihood method to FullCov in `estimatetrend.ctl`:

```
LikelihoodMethod    FullCov
```

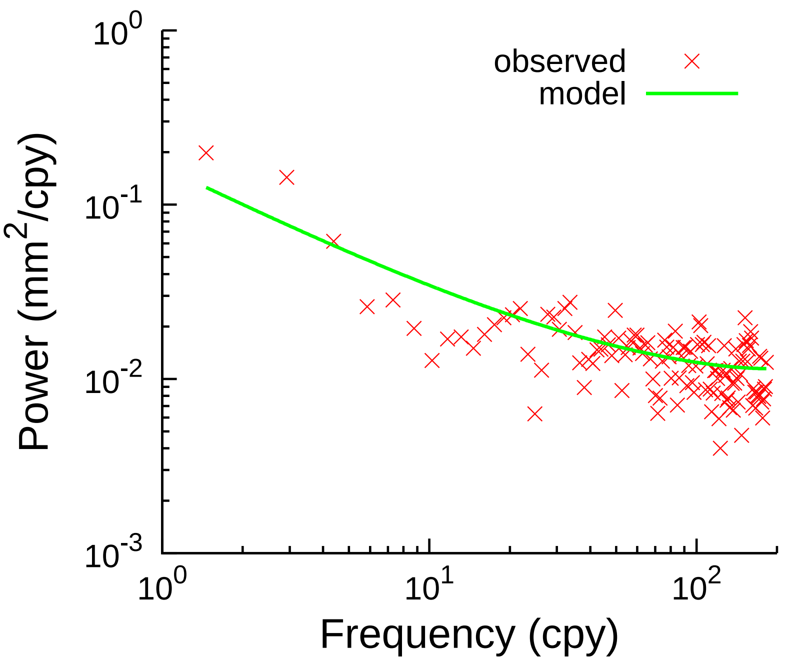

Figure 4: The PSD of the residuals and used noise model.

Running `estimatetrend` will now take about twice as long. Another method is 'Levinson' which is also slower than AmmarGrag but was used by Bos et al. (2008). Note that these three methods give the same answer as it should be. The Levinson method becomes a lot faster if the data is interpolated. This can be achieved by altering the keyword Interpolate in `estimatetrend.ctl`:

```
interpolate         yes
```

After running `estimatetrend`, which is now faster, we get:

```
Powerlaw: fraction=0.671
d = 0.3713 +/- 0.0810

White: fraction=0.3292
No noise parameters to show
STD of the innovation noise: 1.6077
bias : -0.318693 +/- 1.035911 mm (at MJD=50583.500000)
trend: 15.485505 +/- 0.724109 mm/year
```

One can see that linear interpolation of the missing data does not have a large effect on the estimated noise parameters but the nomial bias and the linear trend are altered, although still within the 95% confidence interval (2σ).

The spectral index is around 0.38, not too different from 0.5 which is the spectral index for flicker noise. To investigate what the linear trend error is for flicker plus white noise, we change the NoiseModels keyword to:

```
NoiseModels         Flicker White
```

However, `estimatetrend` does not accept this right away because flicker noise is non-stationary. Therefore, we also need to set the keyword firstdifference:

```
firstdifference     yes
```

Now `estimatetrend` runs fine but note that there are now 20.42% missing data because the missing data were artificially created at random times. Taking the first-difference thus doubles the percentage of missing data. In real GPS time-series the effect is normally less because of the presence of segments of missing data instead of only a set of single missing data points. As a result, we obtain a larger trend uncertainty:

```
trend: 16.260990 +/- 4.542752 mm/year
```

If interpolation is used, then this result changes to:

```
trend: 15.467635 +/- 0.927564 mm/year
```

which is similar to the results obtained before.

Finally, one could use the approximation of the power-law covariance matrix as explained by Bos et al. (2012). This requires no interpolation and no first-difference:

```
interpolation       no
firstdifference     no
NoiseModels         PowerlawApprox White
TimeNoiseStart      1000.0
```

Now the result is:

```
bias : 0.870049 +/- 0.962471 mm (at MJD=50583.500000)
trend: 16.437759 +/- 0.721958 mm/year
```

There is no single set up which works best for all situations. Nevertheless, the AmmarGrag likelihood method (i.e. how the likelihood value is computed) is the fastest method when the number of missing data is less than around 50%, otherwise the FullCov method should be used. For GPS time-series the spectral index is normally around 0.5 (equal to $\alpha = 1$ or $\nu = -1$) and we obtain in most case the best result with the noise model combination "PowerlawApprox White".

3.2 Example 2: A real GPS time-series with a lot of missing data

GPS position time-series are normally given in a Cartesian reference frame with the origin at the centre of the Earth, with the X and Y-axes lying in the equatorial plane and with the X-axis passing through the Greenwich meridian. For plate tectonic research it is more convenient to use the North, East and Up reference frame. This also separates the Up component, which is normally noisier, from the North and East component. In the directory `examples/ex2` the script `convert_sol_files.tcl` performs this transformation. This script reads all filenames with the .sol extention in its directory and converts each file from a Cartesian XYZ to a a geodetic East, North and Up reference frame (enu-format, see section 4). This .sol data format is a special format and contains the date in the first column, the year-fraction in the second and the X, Y and Z component in metres in columns 2 to 5. For example, the first four lines of the file PHLW.sol given in the ex2 directory are:

```
2000-08-22 2000.639344   4728141.4515   2879662.3793   3157146.8947
2000-08-23 2000.642077   4728141.4479   2879662.3730   3157146.8923
2000-08-24 2000.644809   4728141.4520   2879662.3746   3157146.8909
2000-08-25 2000.647541   4728141.4585   2879662.3813   3157146.8986
```

Columns 6 to 11 are not shown but contain the autocovariance and cross-correlation values. This information is not used by Hector and can therefore safely be omitted.

Here, we are interested in the North component which must be specified in removeoutliers.ctl. After preprocessing the data with `removeoutliers`, we can run `estimatetrend`. A problem with this time-series is that it consists for 65% out of missing data. The AmmarGrag likelihood method works, but is not particularly fast when the number of missing data is more than 50%. Therefore, in this case one should use the FullCov likelihood method.

The results of running `estimatetrend` (North component) are:

```
Likelihood value
--------------------
min log(L)=-3157.577
AIC       =6319.154
BIC       =6329.770

PowerlawApprox: fraction=0.140
d = 0.5388 +/- 0.0617

White: fraction=0.8603
No noise parameters to show
STD of the innovation noise: 1.9233
bias : 113.351514 +/- 0.938288 mm (at MJD=53939.500000)
trend: 18.907049 +/- 0.143908 mm/year
cos yearly : 0.625495 +/- 0.211591 mm
sin yearly : 0.751631 +/- 0.238983 mm
--> PHLW_out.mom
Total computing time: 29.00000 sec
```

3.2.1 Station position estimation

As was noted in the introduction of this manual, the objective of hector is to estimate the linear trend in geophysical time-series. A linear function is given by $f(t) = a + bt$ where t represents the time, a a constant and b a coefficient of the linear trend. Sometimes also the constant a is of interest as is the case when one wants to estimate the position of a GPS station with respect to a certain reference epoch.

We will now show how to compute the X-coordinate of station PHLW in the Earth fixed Cartesian reference frame with respect to reference epoch 1 January 2008. First,

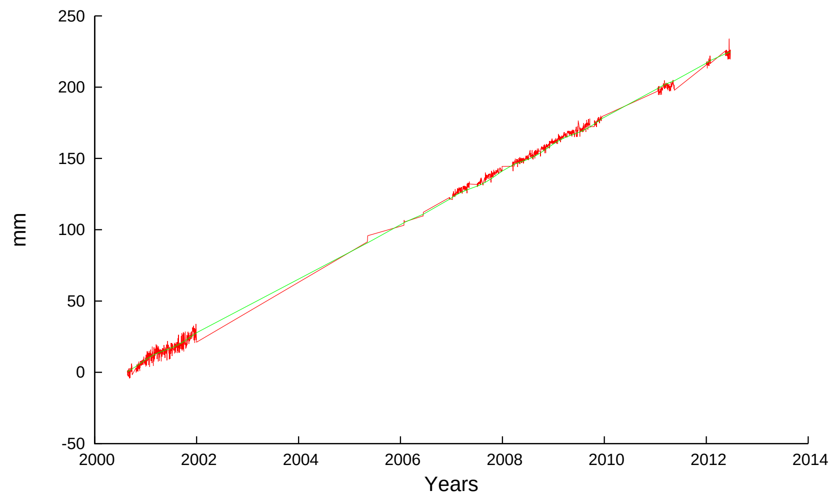

Figure 5: The preprocessed data and the estimated model of the North component of station PHLW, stored in PHLW_out.mom.

the Tcl-script convert_sol_files need to be run again but now with the argument "–position":

```
convert_sol_files.tcl --position
```

This creates three extra files, PHLW_X.mom, PHLW_Y.mom and PHLW_Z.mom, which contain the time-series for each component. The first few lines of PHLW_X.mom are:

```
51778 4728141.4515
51779 4728141.4479
51780 4728141.4520
51781 4728141.4585
51782 4728141.4526
```

The first column contains the Modified Julian Date, the second column contains the X-coordinate of the PHLW station in metres. The next step is to remove the outliers. The control file for this example is `removeoutliers_2.ctl`:

```
DataFile            PHLW_X.mom
DataDirectory       ./
OutputFile          PHLW_X_pre.mom
interpolate         no
firstdifference     no
seasonalsignal      yes
halfseasonalsignal  no
estimateoffsets     yes
PhysicalUnit        m
IQ_factor           3.0
```

Note that now the Physical unit is 'm', not millimetres. The optional keywords 'ScaleFactor' and 'periodicsignals' have been omitted. After running `removeoutliers removeoutliers_2.ctl`, one can run `estimatetrend estimatetrend_2.ctl`. Its controlfile is given by:

```
DataFile            PHLW_X_pre.mom
DataDirectory      ./
OutputFile          PHLW_X_out.mom
interpolate         no
firstdifference     no
seasonalsignal      yes
halfseasonalsignal  no
estimateoffsets     yes
NoiseModels         PowerlawApprox White
TimeNoiseStart      1000
PhysicalUnit        m
ReferenceEpoch      2008 1 1
```

Note here the keyword 'ReferenceEpoch'. The output of `estimatetrend` is:

```
   43    0.88540     0.59247   f()= -4858.989620  size=0.000
converged to minimum at
   44    0.88540     0.59247   f()= -4858.989620  size=0.000

Likelihood value
--------------------
min log(L)=4858.990
AIC       =-9713.979
BIC       =-9703.692

PowerlawApprox: fraction=0.115
d = 0.5925 +/- 0.0855

White: fraction=0.8854
No noise parameters to show
STD of the innovation noise: 0.0049
bias : 4728141.298695 +/- 0.003466 m (at 2008/1/1, 0:0:0.000000)
trend: -0.019957 +/- 0.000500 m/year
cos yearly : -0.002546 +/- 0.000630 m
sin yearly : 0.000349 +/- 0.000696 m
--> PHLW_X_out.mom
Total computing time: 22.00000 sec
```

In this case, the X-coordinate of PHLW is 4728141.299 metres at 1 January 2008.

3.3 Example 3: The monthly PSMSL tide gauge data at Cascais

In the directory `examples\ex3` we have stored the monthly tide gauge data of Cascaise, downloaded from PSMSL (`http://www.psmsl.org/data/obtaining/stations/52.php`).

This time-series has no outliers so we can directly estimate the linear trend. The control file `estimatetrend.ctl` is:

```
DataFile            52.rlrdata
DataDirectory       ./
OutputFile          52_out.mom
QuadraticTerm       no
interpolate         no
firstdifference     no
seasonalsignal      yes
halfseasonalsignal  yes
estimateoffsets     no
NoiseModels         ARMA
PhysicalUnit        mm
AR_p                1
MA_q                0
```

Here we are using the ARMA noise model. To be precise, there is 1 AR coefficient (set by the AR_p keyword) and 0 MA coefficients (set by the MA_q keyword). Thus, we can shorten our notation of ARMA(1,0) to AR(1). If we now run `estimatetrend` we obtain a linear trend of 1.270 $\pm$ 0.075 mm/yr. If we now change the noise model to AR(5), ARFIMA with AR_p=1 and MA_q=0 and GGM we obtain trends of 1.277 $\pm$ 0.103, 1.253 $\pm$ 0.175 and 1.259 $\pm$ 0.201 mm/yr which all have lower BIC and AIC values than the AR(1) noise model. Using `modelspectrum` and `estimatespectrum` one can produce the power spectral density plot shown in Figure 6. Note that the sampling time in hours is 730.5 hours. Furthermore, since only one noise model is used each time, the fraction is always 1. The controlfile `estimatespectrum.ctl` is:

```
DataFile            52_out.mom
DataDirectory       ./
interpolate         no
firstdifference     no
```

This provides us with the frequency range of 1.1317e-09 to 1.9013e-07 Hz which needs to be fed into `modelspectrum`.

In sea level research one is sometimes also interested in the acceleration. It is possible to estimate this by setting the optional keyword 'QuadraticTerm' to yes in `estimatetrend.ctl`. It's default value is no. If we do this, then we obtain, using the GGM noise model, an acceleration of 0.007 $\pm$ 0.012 mm/yr^2.

3.4 Example 4: Creating synthetic coloured noise

In order to perform Monte Carlo simulations, one must create time-series with synthetic coloured noise. This task can be performed with the program `simulatenoise`. It is based on the method described by Kasdin (1995) where an impulse response, different for each noise model, is convoluted with a white noise time-series. The result is our desired synthetic noise time-series. As usual, the convolution is performed using FFT. There might be some spin-up effects because implicitly it is assumed that the noise is

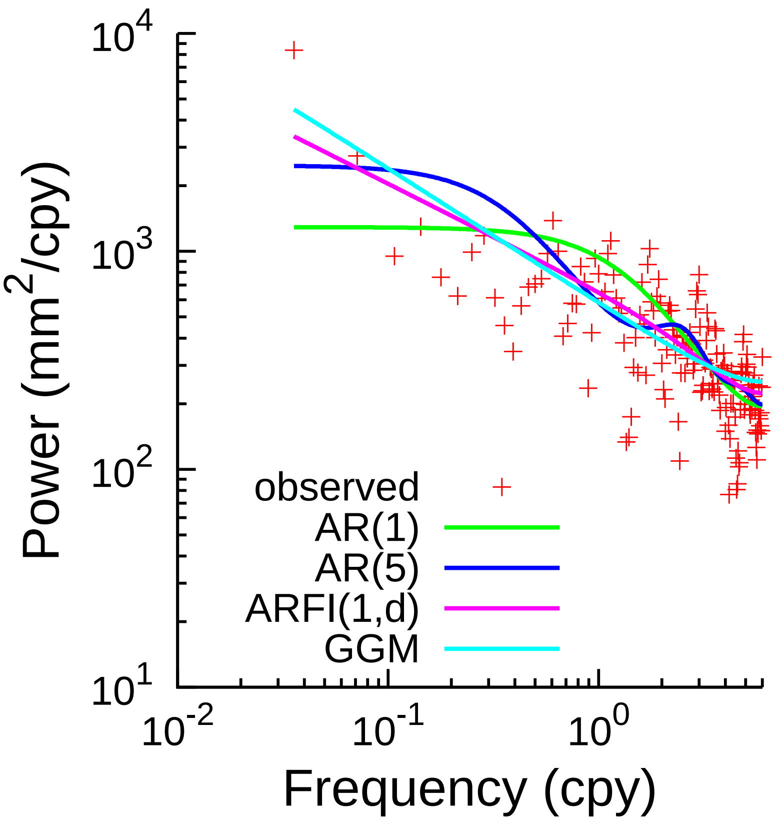

Figure 6: The power spectral density plot of tide gauge data at Cascais.

zero before the first observation. To migitate this problem, the keyword 'TimeNoiseStart' can have a large number, normally 1000 is enough, to specify the amount of extra points before the first observations need to be modelled, see also section 5.4.

In the directory `examples\ex4` the control file `simulatenoise.ctl` is given:

```
SimulationDir           ./
SimulationLabel         test_base
NumberOfSimulations     10
NumberOfPoints          5000
SamplingPeriod          1
TimeNoiseStart          1000
NoiseModels             ARFIMA
AR_p                    1
MA_q                    0
```

Some of these keywords are new. For example, 'SimulationDir' specifies in which directory the created files should be stored. The keyword 'SimulationLabel' specifies the base name of those files. The next keyword tells hector how many simulation runs are required. The filenames will in this case be: test_base0.mom, test_base1.mom, ..., test_base9.mom.

The keyword 'NumberOfPoints' specifies the number of points in the the time-series and the keyord 'TimeNoiseStart' was already discussed above.

When `simulatenoise` is run, it will ask the user to manually enter the parameter values of the chosen noise model. In this case it will ask the values of ϕ and d.

4 Acceptable data format

Hector can accept various data formats which are described in this section. All of the are plain ASCII files and the time should always be increasing and the time step should be constant. The data format is specified by the extension of the filename. For example, the file name TEST.enu has the extension 'enu'.

For the mom and enu-format, the sampling period in days can be specified in the header as follows:

```
# sampling period 1.0
```

If this information is missing, then hector tries to estimate the sampling period from the first few observations. The sampling periods it can detect automatically are: 0.5 hour, 1 hour, 1 day and 7 days. Note that this sampling period must always be given in days!

4.1 mom-format

This format expects 2 or 3 columns. The first column contains the time in MJD, the second the Observations. The third column is optional and should contain the estimated Model. Missing data are allowed.

4.2 enu-format

This format is similar to the mom-format but has four columns. The first column contains the time in MJD, the second to the fourth column contain the East, North and Up component. Missing data are allowed.

4.3 pos-format

This format is specified by the Plate Boundary Observatory (`http://pbo.unavco.org/data/gps`) The first 9 header files are ignored and from all rows with data, only the columns with the Modified Julian Date, the X-Y-Z coordinates and the Ndel-Edel-Udel values are read. Which component is being analysed is set with the keyword 'component' and one of the following values: X, Y, Z, North, East or Up. Note that the units are in metres in these files so you might consider set the keyword 'ScaleFactor' to 1000.0 and the PhysicalUnit to 'mm' to work in millimetres. Time of offsets for the East, North or Up component can be specified in a separate file. To use it, set the keywords 'OffsetFile' and 'component' and give the filename and value of the component (East,North,Up). An example of this file is:

```
 3- 3-2008  NaN 0  0
13-10-2010  NaN 0  0
```

where 'NaN' specifies that an offset needs to be estimated. In this case we thus specify offsets for the East component.

4.4 neu-format

This format is used by SOPAC (`http://sopac.ucsd.edu/`) and accepted by the CATS software. The first column contains the time as year-fractions and columns two to four contain the North, East and Up component in metres. Missing data are allowed. The unit of these files is normally metres and it is convenient to convert these to millimetres using the keyword ScaleFactor in `removeoutliers.ctl`. The year-fractions are converted inside Hector to MJD using the formula:

$$MJD = floor(365.25 * (T - 1970) + 40587 + 0.1) - 0.5 \tag{2}$$

This implies that only sampling periods which are an integer multiple of 1 day are acceptable. Hector can read the slightly different offset headers which are of the form, see the CATS manual (Williams, 2008):

```
# offset 2003.45479 7
```

Note however that if an external file with offset information is used, then the expected format is of the form day-month-year NaN NaN NaN. where the last three parameters stand for East, North and Up. NaN indicates that an offset needs to be estimated. A normal number such as 0, tells the program not offset needs to be estimated for that component.

4.5 rlrdata-format

Hector can read PSMSL's monthly data format, see `http://www.psmsl.org/`. To create an evenly spaced data set inside Hector, each month is assumed to take exactly 30.4375 days, equal to 730.5 hours.

5 Implemented Noise Models

Hector can use various types of noise models and in addition, accepts various combinations of them with power-law plus white noise being the most popular for GPS time-series. Williams (2008) wrote the covariance matrix $\mathbf{C}$ of this particular combination as:

$$\mathbf{C} = \sigma^2 \left(\cos^2\phi\,\mathbf{I} + \sin^2\phi\,\mathbf{E}(d)\right) \tag{3}$$

where $\mathbf{I}$ is the unit matrix (equal to the covariance matrix for unit white noise) and $\mathbf{E}$ the covariance matrix for power-law noise which depends on the spectral index d. The distribution of the magnitudes of both noise models is controlled by the parameter ϕ. The total variance of the noise is set by σ^2. This has been generalized to:

$$\begin{aligned}\mathbf{C} = \sigma^2\,(\phi_1(1-\phi_2)\ldots(1-\phi_N)\mathbf{E}_1 + (1-\phi_1)\phi_2\ldots(1-\phi_N)\mathbf{E}_2 + \\ (1-\phi_1)(1-\phi_2)\ldots\phi_N\mathbf{E}_{N+1})\end{aligned} \tag{4}$$

for $N+1$ noise models. All ϕ-parameters vary between 0 and 1. As was noted in section 1, only stationary noise is accepted. This creates a Toeplitz covariance matrix and only the first column of the covariance matrix needs to be stored. This column vector will be denoted by γ.

5.1 White noise

For white noise the covariance matrix is just the unit matrix The first column of the covariance matrix $\mathbf{C}$, with $\sigma = 1$, is:

$$\gamma_i = 1 \qquad \text{for } i = 0 \tag{5}$$

$$= 0 \qquad i \neq 0 \tag{6}$$

Its one-sided power spectrum density is:

$$S(f) = 2\frac{1}{f_s} \tag{7}$$

where f_s is the sampling frequency in Hz. If you integrate this from zero frequency to the Nyquist frequency, you get the variance that is observed in the time-series, as it should be. It can be used for first-differenced data.

5.2 Power-law noise

For power-law noise the first column of the covariance matrix is:

$$\gamma_i = \frac{\Gamma(d+i)\Gamma(1-2d)}{\Gamma(d)\Gamma(1+i-d)\Gamma(1-d)} \tag{8}$$

Its one-sided power spectrum density, with $\sigma = 1$, is:

$$S(f) = 2\frac{1}{f_s}\frac{1}{(2\sin(\pi f/f_s))^{2d}} \tag{9}$$

It can be used for first-differenced data.

5.3 Flicker noise and Random Walk noise

Flicker noise and Random walk noise are simply two types of power-law noise where the spectral index d has the fixed value of 0.5 and 1.0 respectively. It can be used for first-differenced data.

5.4 Power-law approximated

Bos et al. (2012) note that it is still not clear what causes the noise in GNSS time-series, nor is it known when the noise started. This could be when the receiver was switched on or since the launch of the GNSS satellites or some geophysical signal that has always been present. They used this uncertainty to let the noise start at some arbitrary time in the past. Since the noise is weakly stationary, after some time the covariance matrix almost obtains a Toeplitz structure which means that fast matrix operations can be used. This Toeplitz approximation can be chosen using the name 'PowerlawApprox' after the keyword Noisemodels. In addition, one must specify the number of days before the start of the observation when the noise is assumed to have started after the keyword TimeNoiseStart. A value of 1000 is a good first guess.

5.5 ARFIMA and ARMA

The definition of the ARFIMA noise model is Sowell (1992):

$$\Phi(L)(1-L)^d z_t = \Theta(L)\epsilon_t \tag{10}$$

L is the backshift operator ($Lx_i = x_{i-1}$), z_t is the residual at time t (observation minus modelled signal) and ϵ is a white noise signal. In other words, Eq. (10) says that the residuals in the observations can be produced by applying some transformations on a white noise process. The operators Φ and Θ are defined as, see Hosking (1981):

$$\Phi(L) = 1 - \phi_1 L - \phi_2 L^2 - \ldots - \phi_p L^p \tag{11}$$
$$\Theta(L) = 1 + \theta_1 L + \theta_2 L^2 + \ldots + \theta_q L^q \tag{12}$$

This definition is implemented in Hector but note that the definition of the signs before the ϕ coefficients in Φ are positive in Sowell (1992). To complicate matters further, the coefficients of Θ are negative in the formula's of Zinde-Wash (1988).

The value of the integers p and q are set by the keywords AR_p and MA_q respectively in `estimatetrend.ctl`. It is advised to use values for p smaller than 5 to ensure that the MLE procedure always start with coefficients values for $\phi_1, \dots, \phi_p$ of $\Phi(L)$ that produce stationary noise. If p and q are zero, then one obtains again a pure power-law noise process. We have implemented the method of Doornik and Ooms (2003) to compute the first column of the covariance matrix. For the special case when $d = 0$, we use the equations of Zinde-Wash (1988) and can be selected by using the name 'ARMA' after the keyword Noisemodels. For sea level research the first order auto-regressive noise model is a popular choice: ARMA(1,0). For pure ARMA noise models faster Maximum Likelihood Methods exist, see for example Brockwell and Davis (2002), but Hector will give the same result.

It can be used for first-differenced data.

5.6 Generalized Gauss Markov noise model

Langbein (2004) took the first order Gauss Markov noise model depending on the parameter ϕ and modified with an additional parameter, d, to create power-law noise with a slope of $2d$ in the power density spectrum which flattens to white noise at the very low and very high frequencies. The analytical expression for the autocovariance vector (with $\sigma = 1$) for this noise model is:

$$\gamma_i = \frac{\Gamma(d+i)(1-\phi)^i}{\Gamma(d)\Gamma(1+i)}\, {}_2F_1(d, d+i; 1+i; (1-\phi)^2) \tag{13}$$

This noise model can be used using the name 'GGM' after the keyword Noisemodels in `estimatetrend.ctl`.

6 Quick reference for the control files

6.1 removeoutliers.ctl

This file is read by `removeoutliers`.

Keyword	Value(s)
DataFile	name of file with observations
DataDirectory	directory where file with observations is stored
OffsetFile	name of file with header information (offsets)
OutputFile	name of file with observations *and* estimated model in .mom format
component	only required for the .enu and .neu format or when an OffsetFile is being used. Select East, North or Up
interpolate	yes|no
firstdifference	yes|no
QuadraticTerm	yes|no (optional, default=no)
seasonalsignal	yes|no
halfseasonalsignal	yes|no
periodicsignals	a sequence of numbers reprenting the period in days (optional)
estimateoffsets	yes|no
ScaleFactor	a number to scale the observations (optional, default=1)
PhysicalUnit	the physical unit of the observations
IQ_factor	the number used to scale the interquartile range

6.2 estimatetrend.ctl

This file is read by `estimatetrend` and by `modelspectrum` although the latter only reads the keyword NoiseModels and, if necessary, the keywords AR_p, MA_q and TimeNoiseStart. The program `modelspectrum` creates a file called `modelspectrum.out`.

Keyword	Value(s)
DataFile	name of file with observations
DataDirectory	directory where file with observations is stored
OffsetFile	name of file with header information (offsets)
OutputFile	name of file with observations *and* estimated model in .mom format
component	only required for the .enu and .neu format or when an OffsetFile is being used. Select East, North or Up
interpolate	yes|no
firstdifference	yes|no
QuadraticTerm	yes|no (optional, default=no)
seasonalsignal	yes|no
halfseasonalsignal	yes|no
periodicsignals	a sequence of numbers, separated by spaced, representing the period of the periodic signals in days (optional)
estimateoffsets	yes|no
ScaleFactor	a number to scale the observations (optional, default=1)
PhysicalUnit	the physical unit of the observations
NoiseModels	chose any set from: White, Flicker, RandomWalk, Powerlaw, PowerlawApprox, ARFIMA, ARMA and GGM. (**Note:** only White, Flicker, RandomWalk, Powerlaw can be used when firstdifference=yes)
LikelihoodMethod	chose one from: AmmarGrag, Levinson or FullCov (optional, default=Ammargrag if percentage of missing data is less than 50% of the whole time-series. Otherwise FullCov method is used.
AR_p	number of AR coefficients (only for ARFIMA or ARMA)
MA_q	number of MA coefficients (only for ARFIMA or ARMA)
TimeNoiseStart	number of days before the start of the observations when it is assumed that the noise started (only for PowerlawApprox)

6.3 estimatespectrum.ctl

This file is read by `estimatespectrum` which creates a file called `estimatespectrum.out`.

Keyword	Value(s)
DataFile	name of file with observations
DataDirectory	directory where file with observations is stored
interpolate	yes|no
firstdifference	yes|no
ScaleFactor	a number to scale the observations (optional, default=1)

6.4 simulatenoise.ctl

This file is read by `simulatenoise`.

Keyword	Value(s)
SimulationDir	directory where created files will be stored
SimulationLabel	base name of the created files
NumberOfSimulations	number of simulations
NumberOfPoints	length of each simulation
SamplingPeriod	specifies the time step in days
TimeNoiseStart	number of points that need to be included before 1st observation has been made.
NoiseModels	chose any set from: White, Flicker, RandomWalk, Powerlaw, PowerlawApprox, ARFIMA, ARMA and GGM.

7 License and Copyright

References

Agnew, D. C. (1992). The time-domain behaviour of power-law noises. *Geophys. Res. Letters*, 19(4):333–336.

Beran, J. (1992). Statistical methods for data with long-range dependence. *Statistical Science*, 7(4):404–416.

Bos, M. S., Fernandes, R. M. S., Williams, S. D. P., and Bastos, L. (2008). Fast error analysis of continuous GPS observations. *J. Geod.*, 82:157–166.

Bos, M. S., Fernandes, R. M. S., Williams, S. D. P., and Bastos, L. (2012). Fast Error Analysis of Continuous GNSS Observations with Missing Data. *J. Geod.*, ?:?–?

Brockwell, P. and Davis, R. A. (2002). *Introduction to Time Series and Forecasting*. Springer-Verlag, New York, second edition edition.

Buttkus, B. (2000). *Spectral Analysis and Filter Theory in Applied Geophysics*. Springer-Verlag Berlin Heidelberg.

Doornik, J. A. and Ooms, M. (2003). Computational Aspects of Maximum Likelihood Estimation of Autoregressive Fractionally Integrated Moving Average Models. *Computational Statistics and Data Analysis*, 42:333–348.

Hosking, J. R. M. (1981). Fractional differencing. *Biometrika*, 68:165–176.

Kasdin, N. J. (1995). Discrete simulation of colored noise and stochastic processes and $1/f^{\alpha}$ power-law noise generation. *Proc. IEEE*, 83(5):802–827.

Langbein, J. (2004). Noise in two-color electronic distance meter measurements revisited. *J. Geophys. Res.*, 109(B04406).

Langbein, J. and Bock, Y. (2004). High-rate real-time GPS network at Parkfield: Utility for detecting fault slip and seismic displacements. *Geophys. Res. Letters*, 31:15.

Sowell, F. (1992). Maximum likelihood estimation of stationary univariate fractionally integrated time series models. *J. Econom.*, 53:165–188.

Williams, S. D. P. (2008). CATS : GPS coordinate time series analysis software. *GPS Solutions*, 12(2):147–153.

Zinde-Wash, V. (1988). Some Exact Formulae for Autoregressive Moving Average Processes. *Econometric Theory*, 4(3):384–402.

A History of changes made in the various versions of Hector

A.1 Version 1.1

1. The programs `estimatetrend`, `removeoutliers`, `estimatespectrum` and `modelspectrum` now accept the name of the control file on the command line. For example:

```
estimatetrend mycontrol.ctl
```

2. The date of the nominal bias of the whole time-series can now be set in estimatetrend.ctl using the keyword 'ReferenceEpoch'. Furthermore, in the output this date is now shown in "day/month/year hour" format instead of Modified Julian Date. If this keyword is not provided, then the default behaviour is to define the ReferenceEpoch as the midpoint between the start and end time of the observations.

3. The ability to leave out a keyword also made it possible to define other default parameters. Now it is no longer necessary to provide the ScaleFactor keyword if this is not different from 1 and the LikelihoodMethod keyword is optinal. If it is missing, then the AmmarGrag method will be used when the amount of missing data is less than 50% of the whole time-series. Otherwise the FullCov method is used.

4. The ARFIMA model had two bugs, one due to a sign error and one due to accessing arrays outside their range, which have been resolved.

5. The keyword 'MinimizingMethod' has been removed because the Nelder-Mead Simplex method is the only method available.

6. The header information about the sampling period and the time of offsets can now be given in a separate file using the keyword 'OffsetFile'.
7. It is now also possible to estimate a quadratic polynomial by setting the keyword 'QuadraticTerm' to yes.
8. The program `simulatenoise` was added.
9. Implemented the dd-mm-year NaN NaN NaN offset format for external files.

A.2 Version 1.2

1. `simulatenoise` has now correct power.
2. The ARMA and ARFIMA noise model now also accept first-difference.
3. All programs now except ridiculously long names.

A.3 Version 1.3

1. Removed a bug which caused Hector to crash on some computers. The reason was that Nnumbers was not set to zero explicitly.
2. The parser of the control files had trouble when there was a space after the last label. For example "NoiseModels PowerlawApprox White " where there is a space behind the last word. This has now hopefully been corrected in Control.cpp
3. Using the Generalized Gauss Markov noise model did not converge in rare situations but they did occur. Now the maximum value of ϕ has been lowered from 0.9999 to 0.999.
4. The binaries are now stored in `/usr/local/bin` instead of `/usr/bin` which I hope is more following the Linux standard.

A.4 Version 1.4

1. Better command line parsing for estimatespectrum.
2. Added "# sampling period 1.0" in .enu file created by `convert_sol_files.tcl`.
3. Removed root-message in ARFIMA.cpp.
4. Removed trailing spaced bug in Control.cpp (again).
5. Improved C++ correctness (fp.getline(..)!=NULL) in Observations.cpp