Hector user manual
version 2.0

Machiel Bos

November 13, 2021

Contents

1 Introduction **3**
1.1 How to cite Hector . . . 3
1.2 Main features . . . 4
1.3 Disclaimer . . . 4

2 Installation **4**

3 Hector conventions **5**

4 Tutorial **7**
4.1 Example 1: Synthetic GNSS time series with spikes and offsets . . . 7
4.1.1 Removal of outliers . . . 8
4.1.2 Estimation of the linear trend . . . 10
4.1.3 Plotting the power spectral density . . . 15
4.2 Example 2: The monthly PSMSL tide gauge data at Cascais . . . 17
4.3 Example 3: Creating synthetic coloured noise . . . 19
4.4 Example 4: Post-seismic relaxation . . . 20
4.5 Example 5: Automatic analysis + varying seasonal signal . . . 21

5 Model specification **23**
5.1 Post-seismic deformation . . . 24
5.2 Multi-trend . . . 25
5.3 Geophysical Signal . . . 25
5.4 Signal to Correct Observations . . . 26

6 Acceptable data format **26**
6.1 mom-format . . . 26
6.2 enu-format . . . 27
6.3 neu-format . . . 27
6.4 rlrdata-format . . . 27

7 Implemented Noise Models 27
7.1 White noise . . . 28
7.2 Power-law noise . . . 29
7.3 Power-law approximated . . . 29
7.4 ARFIMA and ARMA . . . 29
7.5 Generalized Gauss Markov noise model . . . 30
7.6 Flicker noise and Random Walk noise . . . 30
7.7 Matern noise model . . . 31
7.8 VaryingPeriodic (Band-pass) noise model . . . 32

8 The Akaike and Bayesian Information Criteria 32

9 Auxiliary Python scripts 33
9.1 analyse_timeseries.py . . . 34
9.2 analyse_and_plot.py . . . 36
9.3 find_offset.py . . . 36
9.4 find_all_offsets.py . . . 37
9.5 apply_WF.py . . . 37

10 Quick reference for the control-files 38
10.1 removeoutliers.ctl . . . 38
10.2 estimatetrend.ctl and findoffset.ctl . . . 39
10.3 estimatespectrum.ctl . . . 41
10.4 modelspectrum.ctl . . . 41
10.5 simulatenoise.ctl . . . 41

11 Advanced Features 42
11.1 Offsets . . . 42

12 License 42

A History of changes made in the various versions of Hector 45
A.1 Version 1.1 . . . 45
A.2 Version 1.2 . . . 45
A.3 Version 1.3 . . . 46
A.4 Version 1.4 . . . 46
A.5 Version 1.5 . . . 46
A.6 Version 1.5.1 . . . 47
A.7 Version 1.5.2 . . . 47
A.8 Version 1.6 . . . 47
A.9 Version 1.7.2 . . . 47
A.10 Version 1.9 . . . 48
A.11 Version 2.0 . . . 48

1 Introduction

Hector is a software package that can be used to estimate a trend in time series with temporal correlated 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. The trend can be linear or a higher degree polynomial and in addition one can estimate periodic signals, offsets and post-seismic deformation. Together they represent the model that is fitted to the observations.

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 model parameters 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 model parameters *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). 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. Non-stationary noise is approximated by stationary noise (Bos et al., 2008, 2013).

Another alternative is the program est_noise of Langbein (2010). Recent versions include some tricks from Bos et al. (2013) to deal with missing data but with a different way to construct the covariance matrix (Langbein, 2017). In the book by Montillet and Bos (2019) more examples on the analysis of geodetic time series with temporal correlated noise can be found.

This manual starts in section 2 by explaining how to install Hector on your computer and in section 3 a general description of Hector is given. In section 4 a tutorial on using Hector is presented using two examples of analysing synthetic GNSS data with offsets and outliers and a real GNSS data set. This is followed by section 5 where the models that can be fitted to the observations are described. Sections 6 and 7 explain the acceptable data formats and implemented noise models respectively.

In section 8 we discuss various information criteria that can be used to select the best noise model to describe the stochastic properties of the residuals. In section 9 we present some auxiliary Python script that call Hector executables in order to automise various tasks. Finally, a quick reference of the parameters in the control-files are presented in 10.

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. (2013). Fast Error Analysis of Continuous GNSS Observations with Missing Data. *J. Geod.*, Vol. 87(4), 351–360, 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, generalised Gauss-Markov (GGM), Matern, Varying Periodic (i.e. Bandpass) and white noise models. Any combination of these models can be made.
5. Comes with programs to remove outliers, to make power spectral density plots and to create files with synthetic coloured noise.
6. Has a program to automatically detect offsets in the time series.
7. Provides a script to estimate a varying seasonal signal.

1.3 Disclaimer

The user should take care to select a proper noise model to describe the stochastic properties in the time series. Hector cannot not (yet) do this task for the user. If a wrong noise model is included to the list of models then the maximum likelihood method will in most cases correctly estimate a zero variance for this model. However, if the convergence takes a long time then inspection of the selection of noise models is the first thing to look at.

2 Installation

The Hector software package is intended to be run on computers with Unix-like operating systems. For the previous versions of Hector special Debian binary packages were created but these were, as far as we know, never used. Also the compilation of the source code was troublesome for most users because the library names keep on changing and they are slightly different on the various Linux distributions. Therefore, we provide the static binaries for Ubuntu 18.04, 20.04, 21.04, Scientific Linux (SL7) CentOS 7 and Oracle Linux 8 together with the C++ source code. On a Mac computer no static executables are given. In this case one must install Homebrew and follow the compilation instructions found in the distribution of the source code. To use Hector on Windows 10, one can install WSL (`https://wiki.ubuntu.com/WSL`). People brave enough to look at the source code will note is was mostly written in Santa Clara. This is a small parish of the city Coimbra in Portugal, not Santa Clara - California.

The static binaries (64 bit) and source code can be obtained from the website: http://segal.ubi.pt/hector. It is advisable to put the binaries in the directory /usr/local/bin.

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.
findoffset	Program to find the epoch of a possible offset in the time series.
simulatenoise	Program to files with synthetic coloured noise.
date2mjd	Small program to convert calendar date into Modified Julian Date.
mjd2date	The inverse of date2mjd.

If the binaries are put in another directory, then make sure it's added to the `PATH` variable in your shell environment.

The list of programs provides is given in Table 1. To compile the source code, extra libraries are needed such as: Boost, OpenBLAS and FFTW.

To run the examples that are described in this manual, which can be downloaded from the website, one also need the `Python` language and the `gnuplot` plotting program.

3 Hector conventions

Hector consists out of a set of simple command line programs that help you through the steps of time series analysis: removal of outliers, estimation of a trend, estimating the power spectral density and modelling of the estimated power spectral density. In addition, a program exists to find offsets in the data. Finally, synthetic time series with coloured noise can be created.

Each program comes with its own control-file in which parameter values needed by the program are given. For example, the program `removeoutliers` has a control-file called `removeoutliers.ctl`. All programs follow the same naming scheme for their control-files.

This control-file is a simple ASCII text file containing on each row a keyword followed by its value or a list of values. Some keywords are optional and if they are not specified, then the default value will be used. A list of all possible keywords for each control-file is given in section 10.

Following the nomenclature of Bevis and Brown (2014), Hector estimates station trajectory models. These models can include a trend, seasonal signals, offsets and post-seismic deformation. See section 5 for more details. In Hector you can specify the type of trend, which seasonal and other periodic signals in the control-file. In addition you can specify in the control-file if you want to estimate offsets, breaks and/or post-seismic deformation. These are a kind of *global* model parameters that you normally apply to

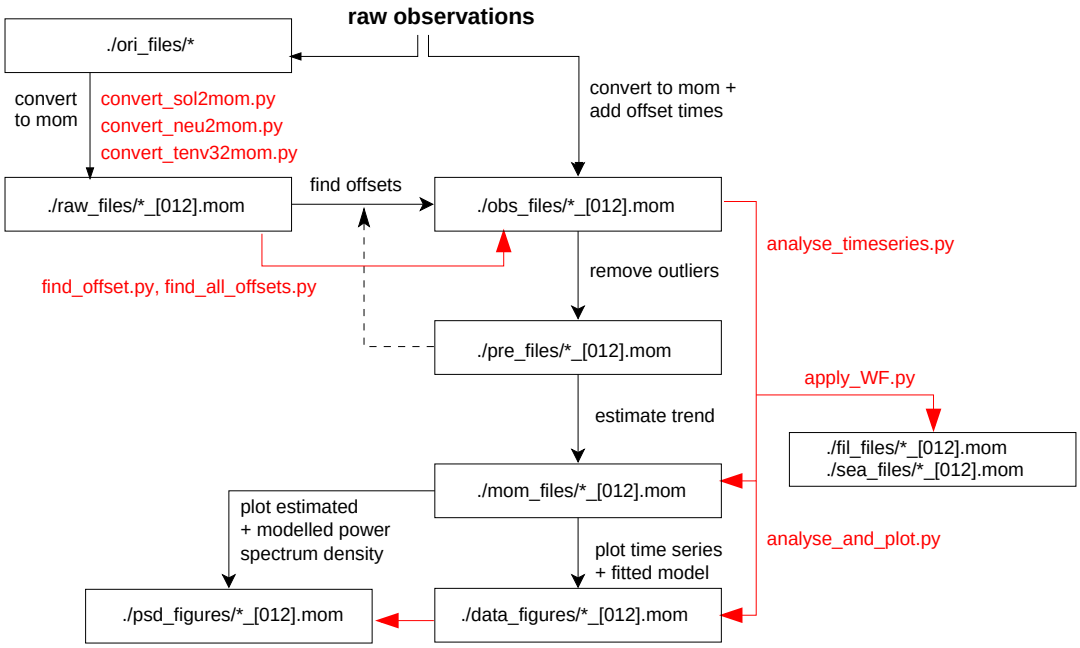

Figure 1: The steps taken in the analysis of time series. Boxes represent the standard directory names to hold the data at each step. The red names are the Python scripts that can be used to call the Hector executables and automate the analysis steps. The `_[012]` extension only applies for time series that have East, North and Up components.

all your time series. On the other hand, the time of an offset, break or post-seismic deformation differs from station to station and are a kind of *local* model parameters. In Hector this information is normally written in the header of the time series.

Next, a schematic work-flow of the various steps taken during the analysis of geodetic time series is shown in Figure 1. The observations are stored in a file and the first step is to convert this into a format Hector can read. Several formats are admissable, see section 6, but to make optimal use of the Python scripts, the mom-format is preferrable. These files can all be stored in a directory such as `./raw_files`. If the observations are made for the East, North and Up component, then one can append `_0`, `_1` and `_2` to the station name to represent the three components respectively. This will facilitate running the Python script for detecting offsets.

The files in the `./raw_files` will be analysed to detect possible offsets and these will be copied to the `./obs_files` with additional information in the header about the date of the possible offsets.

The following step is to search for outliers. The resulting pre-processed files should be stored in `./pre_files`. Actually, it makes sense to remove outliers before attempting to detect offsets. The removal of outliers followed by the detection of offsets can be an iterative process. Once this cycle has finished, one can estimate the trend. The resulting files are stored in the directory `./mom_files`. The user is free to chose his or her own directory names but the Python scripts assume this structure. It has again been summarised in Table 3.

Table 2: Convention of directory names used in this manual.

Step	Directory	Purpose
0	`./ori_files`	The time series in their original format (not mom-format).
1	`./raw_files`	Contains raw observations (mom-format) with outliers and no header information about offsets, breaks etc.
2	`./obs_files`	Contains observations. Header information about offsets, breaks etc. is added but time series still has outliers.
3	`./pre_files`	Contains pre-processed observations which means outliers have been removed.
4	`./mom_files`	Contains the output of `estimatetrend` which are again the pre-processed files but another coloumn is added with the estimated model (m=MJD, o=observations, m=model)
5	`./sea_files`	The estimated varying seasonal signal
6	`./fil_files`	The observations minus the estimated varying seasonal signal

Finally, to visualise the results the time series with the fitted model should be plotted. To inspect how well the noise model represents reality, one should make a power spectral density plot of the residuals and the estimated noise model.

4 Tutorial

The examples are provided together with the source code which you can download from the Hector website and put in any directory you like.

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 GNSS data.

4.1 Example 1: Synthetic GNSS time series with spikes and offsets

In the directory `ex1/obs_files` the file `TEST.mom` is stored. It represents some fictional observed GNSS coordinate time series. The file extension 'mom' stands for **M**odified Julian Date - **O**bservations - **M**odel. Here the last component (The fitted model) is missing. The Modified Julian Date (MJD) is a convenient format to make plots. You can use the programs `date2mjd` and `mjd2date` to convert between year/month/day/hour/minute/second and MJD values. 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 a few header lines:

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

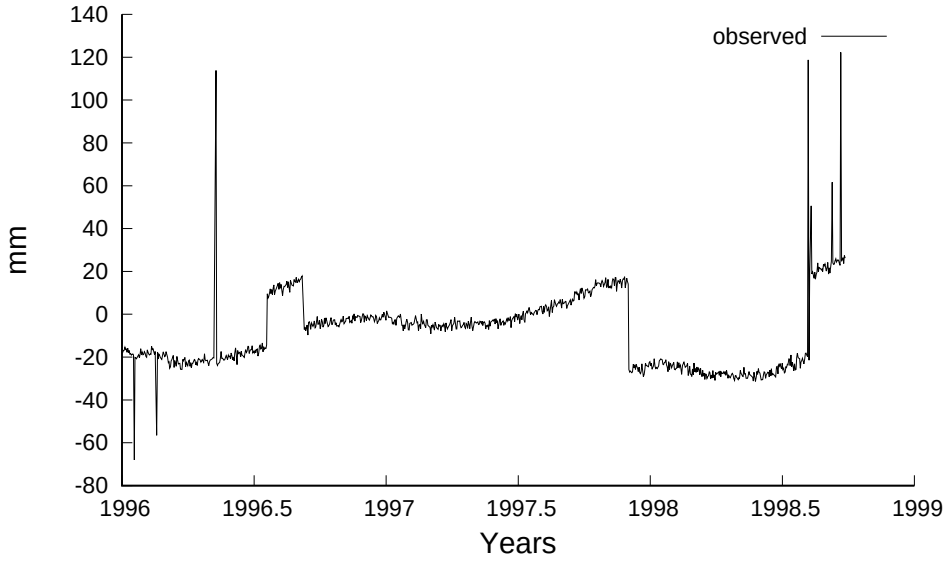

Figure 2: The observed data of `TEST.mom`.

The first line just tells the program that the sampling period of the data is daily (T=1 day). The other lines tell Hector and which epochs an offset needs to be estimated. Since this information is known, the files were stored in the `obs_files` directory and not in `raw_files`. For detecting offsets, see section 9.3.

For the moment we assume that the information about these epochs of the offsets is given. For example, these are times when the GNSS receiver was changed as specified in the logfile. Later on we will deal with the situation when this information is missing. After the header line, the data are listed (`TEST.mom`):

```
50084.0 -17.88951
50085.0 -16.88599
50086.0 -16.84916
...
```

The east component (column 2) can be plotted on the screen using gnuplot (type gnuplot on the command line first to start this plotting program) with the command:

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

The time values are converted from MJD to years. The result is shown in Figure 2 where one can detect the offsets and the presence of outliers.

4.1.1 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 a keyword is optional

and has been omitted, then its default value will be used. Another control-file can be specified on the command line. For example:

```
removeoutliers othercontrolfile.ctl
```

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

```
DataFile            TEST.mom
DataDirectory       ./obs_files
OutputFile          ./pre_files/TEST.mom
interpolate         no
seasonalsignal      yes
halfseasonalsignal  no
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.mom` in our case. The second line gives the directory where this file can be found and the third line 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 **M**JD, **O**bservations, **M**odel. 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 6 for more details on the acceptable data format. To run `removeoutliers`, simply type `removeoutliers` on the prompt. A file called `removeoutliers.out` will be created that contains the MJD values of the points that have been removed.

`removeoutliers` fits a linear trend to the raw data using ordinary least-squares 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 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 are not interpolated. The keyword 'periodicsignals' is optional and can be omitted.

The results of `removeoutliers`, stored in `./pre_files/TEST.mom`, are shown in Figure 3 which have been generated with `gnuplot` using the command:

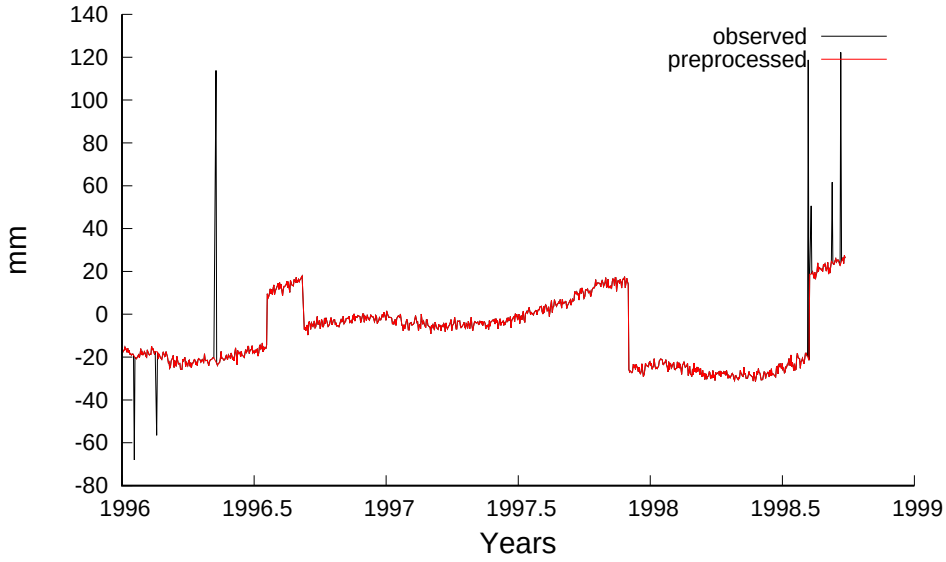

Figure 3: The observed and preprocessed data stored in `./obs_files/TEST.mom` and `./pre_files/TEST.mom`.

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

4.1.2 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.mom
DataDirectory       ./pre_files
OutputFile          ./mom_files/TEST.mom
interpolate         no
seasonalsignal      yes
halfseasonalsignal  no
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.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 are not interpolated. 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: PowerlawApprox, FlickerGGM, RandomWalkGGM, ARMA, ARFIMA or GGM (Generalised Gauss Markov). These are described in more detail in section 7. Note that Power-law only works for spectral indices larger than -1 which implies, stationary noise (Bos et al., 2008, 2013). To work with non-stationary power-law noise, nowadays we use:

```
NoiseModels           GGM White
GGM_1mphi             6.9e-06
```

An interested reader can try this out and see it gives very similar results. See also section 7. For now, we just assume that the 'Powerlaw' noise model was selected.

The chosen method for the Likelihood computation is 'AmmarGrag' which is explained in more detail in Bos et al. (2013). 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 are 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. The information of the offsets can be given in another file by using the keyword 'OffsetFile' and specifying the 'component' keyword, see section 11.

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

```
***********************************
    estimatetrend, version 1.9
***********************************
0) Powerlaw
1) White
Data format: MJD, Observations, Model
Filename              : ./pre_files/TEST.mom
Number of observations: 1000
Percentage of gaps    : 10.6

----------------
  AmmarGrag
----------------
No Polynomial degree set, using offset + linear trend
No extra periodic signal is included.

Number of CPU's used (threads) = 8

   66    0.12668     0.38344
  Return code IFAULT = 0

  Estimate of minimizing value X*:
```

```
  F(X*) = 1726

  Number of iterations = 66
  Number of restarts =   0

Likelihood value
--------------------
min log(L)=-1726.399
k         =8 + 2 + 1 = 11
AIC       =3474.799
BIC       =3527.552
BIC_tp    =3507.335
BIC_c     =3558.760
ln_det_I  =23.208

Noise models
------------
Powerlaw:
fraction  = 0.48942
sigma     = 3.50791 mm/yr^0.19172
d         = 0.3834 +/- 0.0334
kappa     = -0.7669 +/- 0.0667

White:
fraction  = 0.51058
sigma     = 1.15596 mm
No noise parameters to show

STD of the driving noise: 1.61775
bias : 0.838 +/- 1.001 mm (at 1997/5/15, 12:0:0.000)
trend: 16.400 +/- 0.689 mm/year
cos yearly : 4.092 +/- 0.281 mm
sin yearly : -3.937 +/- 0.286 mm
Amp yearly : 5.685 +/- 0.283 mm
Pha yearly : -43.896 degrees
offset at 50284.0000 :   24.49 +/-  0.69 mm
offset at 50334.0000 :  -24.69 +/-  0.75 mm
offset at 50784.0000 :  -39.76 +/-  0.70 mm
offset at 51034.0000 :   38.52 +/-  0.72 mm
--> ./mom_files/TEST.mom
Total computing time: 0.00000 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. The next few lines show information about the minimisation process.

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 Bayesian Information

Criteria (BIC). These values are shown below the value of the natural logarithm of the Likelihood value. A little more information about these two criteria is given in section 8.

Since we are using white and power-law noise, two noise amplitudes need to be estimated. The way how this is done in Hector has confused some users and therefore some extra explanation is in order. First, we model our power-law noise as a Fractionally Brownian motion (Mandelbrot and Van Ness, 1968). This type of noise can be created by filtering white noise. In CATS, Hector and est_noise this covariance matrix is created for the case of filtering white noise with variance 1. The noise amplitude is simply a scale factor of this covariance matrix.

Secondly, we add the covariances to get the total covariance:

$$\mathbf{C} = \sigma_{wn}^2 \mathbf{I} + \sigma_{pl}^2 \mathbf{E}(\kappa) \tag{1}$$

where σ_{wn}^2 and σ_{pl}^2 are the two noise amplitudes we need to estimate. However, as Williams (2008) explains, it is convenient to write this as:

$$\mathbf{C} = \sigma^2 \left[\phi \mathbf{I} + (1-\phi)\mathbf{E}(\kappa)\right] \tag{2}$$

σ^2 can directly be computed from the residuals which means we only need to search for the value of ϕ that maximises our likelihood value. A reduction of the dimension of the parameter space that needs to be searched is beneficial for the speed of the maximum likelihood estimation. In section 7 more details are given.

To return to the output of `estimatetrend`, we can see both the value of ϕ and $1-\phi$, called fractions, and the main driving noise σ^2 which has for this case has a standard deviation of 1.61775 mm. The standard deviation of the white noise is:

$$\sigma_{wn} = \sqrt{0.51058} \times 1.61775 = 1.15596 \tag{3}$$

In Hector the covariance matrix $\mathbf{E}$ is not scaled by the factor $\Delta T^{-\kappa/2}$, where ΔT is the sampling period in years (Williams, 2003). However, to facilitate comparison with amplitude values for power-law noise quoted in the literature, we divide the estimated amplitude by $\Delta T^{-\kappa/4}$:

$$\sigma_{pl} = \frac{\sqrt{0.48942} \times 1.61775}{(1/365.25)^{(0.25 \times 0.7669)}} = 3.50804 \tag{4}$$

The second noise parameter of the power-law noise is the spectral index d which is $-1/2$ times the more often used parameter κ in other papers on GNSS time series. The values of d and ϕ need to be determined using the numerical minimisation scheme.

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 7.

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. By default, the epoch t_0 is chosen at the midpoint of the time series. To explain this better, assume that we have 5 observations. The design

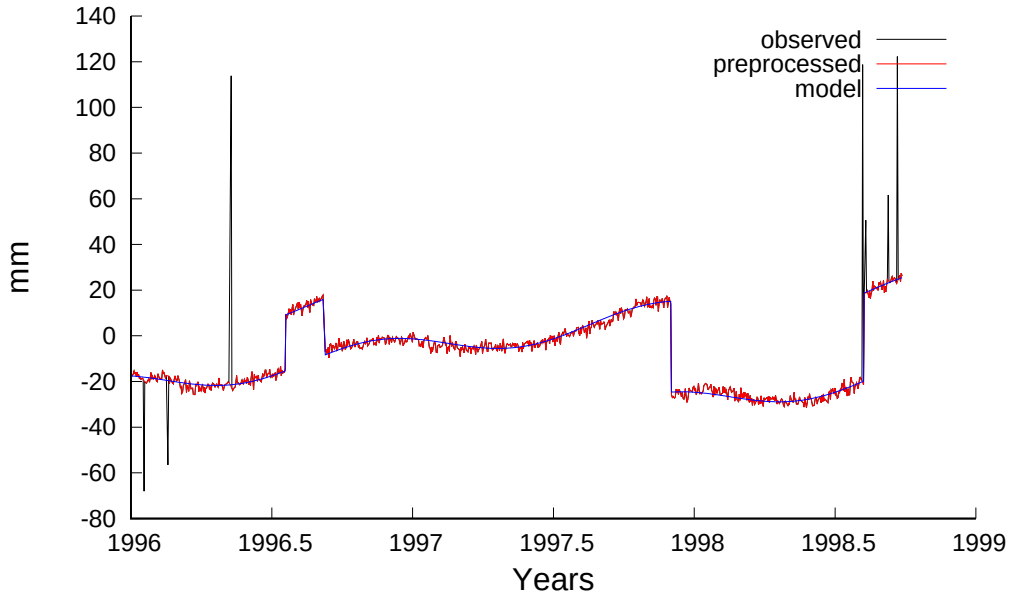

Figure 4: The raw, filtered data and the estimated model of the `TEST.mom` time series.

matrix $\mathbf{H}$ looks like:

$$\mathbf{H} = \begin{pmatrix} 1 & -2 \\ 1 & -1 \\ 1 & 0 \\ 1 & 1 \\ 1 & 2 \end{pmatrix} \tag{5}$$

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 of the nominal bias. 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 in the output.

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
```

Also shown in the output are the values of the estimated offsets. The results of `estimatetrend`, stored in `./mom_files/TEST.mom`', are shown in Figure 4 which have been generated in `gnuplot` with the command:

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

4.1.3 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.mom
DataDirectory       ./mom_files
interpolate         no
ScaleFactor         1.0
WindowFunction      Parzen
Fraction            0.1
```

The contents of `estimatespectrum.out` is shown in Figure 5. 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 (set by the keyword "Fraction") is smoothed to zero using a Parzen window function. The window function is set by the keyword "WindowFunction". Another choice is "Hann". 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. For now, just run:

```
estimatespectrum
```

The output printed on the screen is:

```
**********************************
    estimatespectrum, version 1.9
**********************************
Data format: MJD, Observations, Model
Filename              : ./mom_files/TEST.mom
Number of observations: 1000
Percentage of gaps    : 10.6
window function : Parzen
window function fraction: 0.1
Number of data points n : 1000
Number of data used   N : 1000
```

```
Number of segments    K : 7
Length of segments    L : 250
U : 0.9299
dt: 8.64e+04
scale for G to get Amplitude (mm): 0.3043
Total variance in signal (time domain): 3.295
Total variance in signal (spectrum)   : 2.963
freq0: 4.6296e-08
freq1: 5.7870e-06
--> estimatespectrum.out
```

The PSD of our estimated noise model can be computed using the program `modelspectrum` which has a small control-file with a list of noise models, together with the physical unit:

```
PhysicalUnit        mm
NoiseModels         Powerlaw White
```

The output is saved in `modelspectrum.out`. The user must manually enter the requested values for the noise parameters (see output of `estimatetrend` and provide the begin and end value of the frequency range. For our example, the input looks like:

```
***********************************
    modelspectrum, version 2.0
***********************************
0) Powerlaw
1) White
Enter the standard deviation of the driving noise: 1.61775
Enter the sampling period in hours: 24
Enter fraction for model Powerlaw: 0.48942
Enter fraction for model White: 0.51058

Powerlaw:
Enter value of fractional difference d:0.3834

White:
1) Linear or 2) logarithmic scaling of frequency?: 2
Enter freq0 and freq1: 4.6296e-08 5.7870e-06
freq0 : 4.6296e-08
freq1 : 5.787e-06
--> modelspectrum.out
```

The contents of `estimatespectrum.out` and `modelspectrum.out` are plotted in Figure 5. It can be generated using the command:

```
gnuplot plot_spectra.gpl
```

The plot is located in the directory psd_figures.

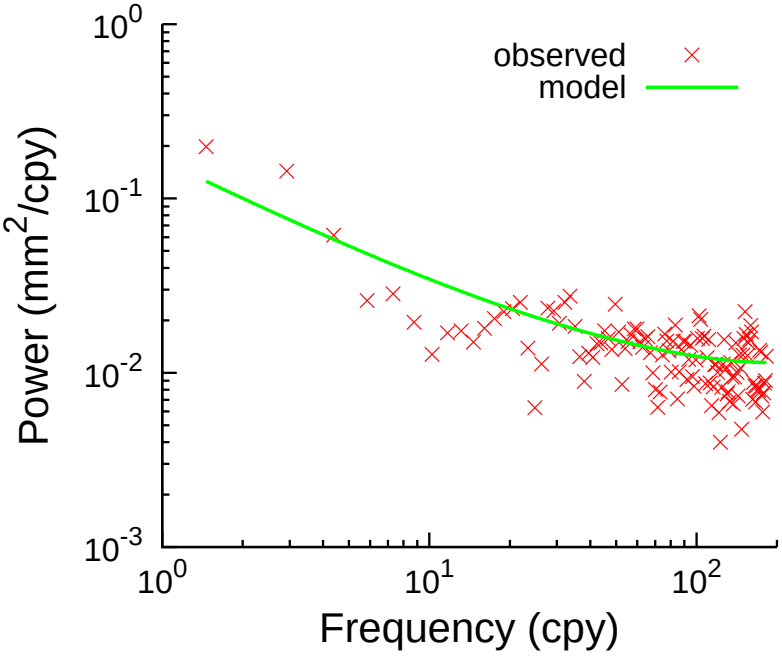

Figure 5: The PSD of the residuals and used noise model.

4.2 Example 2: The monthly PSMSL tide gauge data at Cascais

In the directory ex2 we have stored the monthly tide gauge data of Cascais, 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
DegreePolynomial    1
interpolate         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 $\pm$

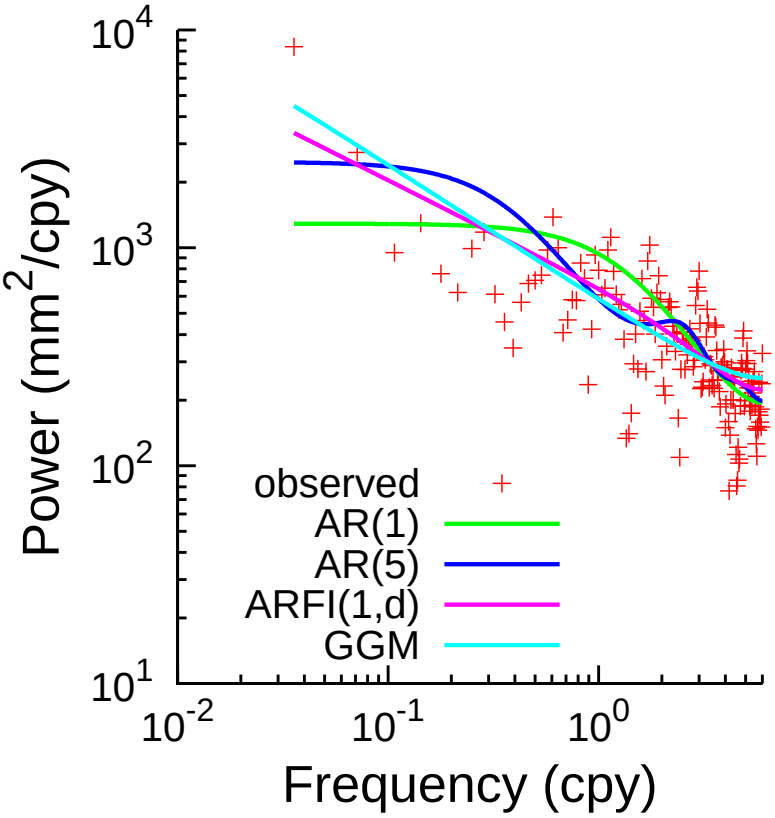

Figure 6: The power spectral density plot of tide gauge data at Cascais.

0.103, 1.253 $\pm$ 0.175 and 1.265 $\pm$ 0.192 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
```

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 'DegreePolynomial' to 2 in `estimatetrend.ctl`. It's default value is 1. If we do this, then we obtain, using the GGM noise model, an quadratic term of 0.004 $\pm$ 0.006 mm/yr^2. Note that this is half of the acceleration.

Next, one can include additional geophysical signals in the analysis. A simple regression coefficient will be estimated. To do so, set the keyword 'estimatemultivatiate yes' in the control file and specify the file with the geophysical signal, see also section 5.3.

For example:

```
estimatemultivariate       yes
MultiVariateFile           hadslp2_cascais.mom
```

Here we include the monthly surface pressure provided by the Hadley Centre Sea Level Pressure dataset (HadSLP2). If we run `estimatetrend`, using again the GGM noise model, we get:

```
STD of the driving noise: 38.17881
bias : 19329.610 +/- 454.450 mm (at 1937/12/31, 12:45:0.000)
trend: 1.292 +/- 0.186 mm/year
cos yearly : 20.003 +/- 1.991 mm
sin yearly : -27.878 +/- 1.941 mm
Amp yearly : 34.369 +/- 1.964 mm
Pha yearly : -54.340 degrees
cos hyearly : 0.650 +/- 1.625 mm
sin hyearly : -10.391 +/- 1.553 mm
Amp hyearly : 10.533 +/- 1.579 mm
Pha hyearly : -86.418 degrees
scale factor of channel 1 :  -12.17 +/-  0.45
```

The last line shows the value for the regression coefficient (i.e. scale factor) for the geophysical signal in column 1 of the multi-variate file. This multi-variate file should be in the mom-format but to be more flexible, one can add as many columns as one like, each containing another geophycial signal. In this case the regression coefficient is -12.17 mm/mbar, which is close to the standard inverted barometer value.

4.3 Example 3: 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 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 7.3.

In the directory `ex3` the control-file `simulatenoise.ctl` is given:

```
SimulationDir           ./
SimulationLabel         test_base
NumberOfSimulations     10
NumberOfPoints          5000
SamplingPeriod          1
TimeNoiseStart          1000
NoiseModels             Flicker White
PhysicalUnit            mm
```

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 keyword '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.

In some cases it is desirable to create the same set of synthetic time series each time `simulatenoise` is run. To achieve this, one can set the keyword 'RepeatableNoise' to yes.

4.4 Example 4: Post-seismic relaxation

After a large earth-quake, the Earth's surface slowly returns to a new position. This post-seismic relaxation can be modelled with an exponential or logarithmic function, see section 5. In the directory `ex4` we have created a synthetic time series with various relaxation signals. We *know* when the synthetic earthquakes occurred and therefore add offset epochs in the header of the observation file `TEST.mom` in the directory `./obs_files`. Furthermore, add information about the relaxtion times (unit is days) for each event:

```
# sampling period 1.0
# offset  51994.0
# offset  53544.0
# offset  55044.0
# log  51994.0   10.0
# log  55044.0   10.0
# exp  53544.0  100.0
```

To force `estimatetrend` model this signal, include the line "estimatepostseismic yes" in the control file:

```
DataFile              TEST.mom
DataDirectory         ./obs_files
OutputFile            ./mom_files/TEST.mom
seasonalsignal        no
halfseasonalsignal    no
estimateoffsets       yes
estimatepostseismic   yes

#--- I model power-law noise by GGM and fixed 1-phi value
NoiseModels           GGM White
GGM_1mphi             6.9e-06

PhysicalUnit          mm
ScaleFactor           1.0
```

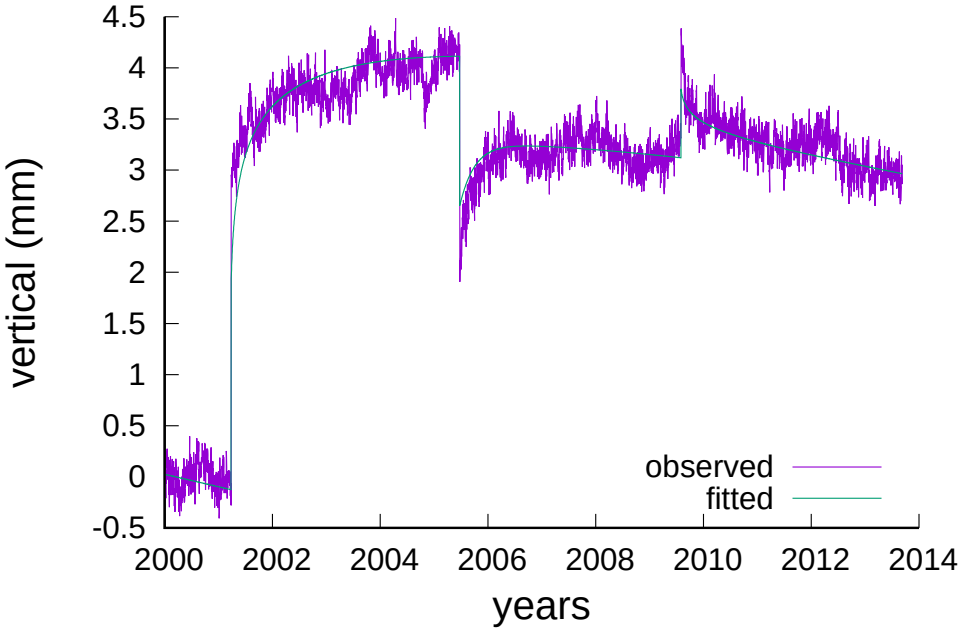

Figure 7: The synthetic time series plus the fitted signal which include various exponential and logarithmic post-seismic relaxations.

The observations and fitted model is plotted in Figure 7. To include slow-slip events the procedure is similar. You need to include the line 'estimateslowslipevent yes' in the control file and add to the mom-file in the header `# tanh MMMM DD` where MMM is the Modified Julian Date and DD the relaxation time in days, see section 5.

4.5 Example 5: Automatic analysis + varying seasonal signal

In this example we have real GNSS time series for the stations MAS1, LHAZ, AUCK and ULAB. The first thing to do is to detect the location of offsets. This can be done by running the Python script `find_all_offsets.py` in the `./ex5` directory. This takes several hours, depending on your computer. The results are stored in the directory `./obs_files`.

Running the Python script `analyse_and_plot.py` searches for outliers, estimate a trend + seasonal signal + offsets and makes plots of the time series and power-spectral density. This is the easiest way of using Hector when one has lots of GNSS time series to analyse.

The stations were selected for this example because they show a significant varying seasonal signal. A first attempt to model these variations is to model the amplitude of the annual + semi-annual signal by a Chebyshev polynomial, following the work of Bennett (2008). The control file for `estimatetrend` is:

```
estimatevaryingseasonal    yes
varyingseasonal_N          3
```

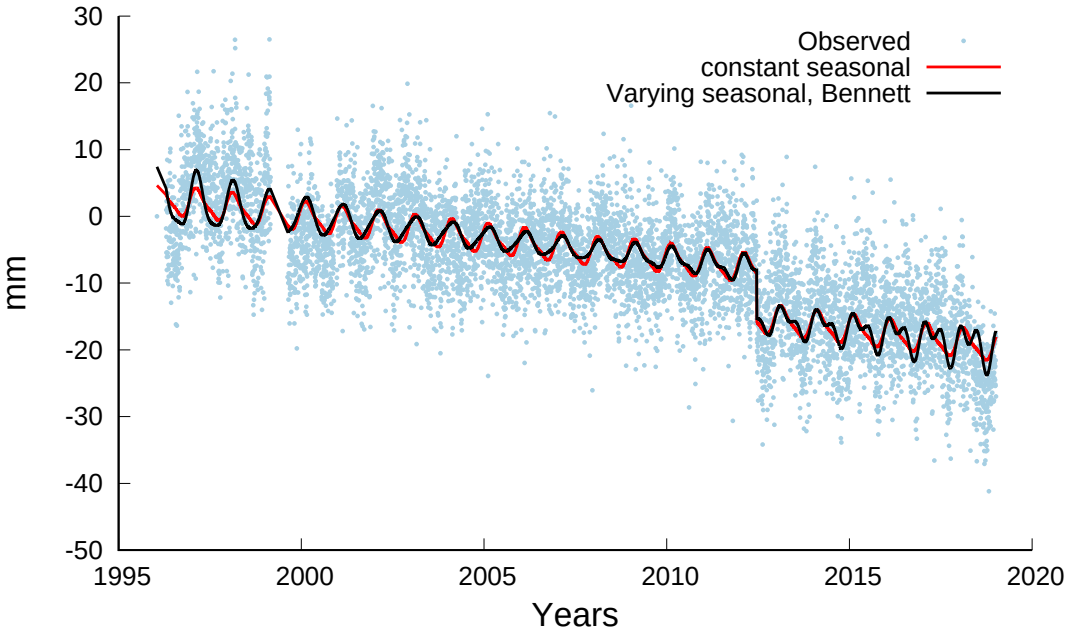

Figure 8: The fitted model to the MAS1 time series. Red is the constant seasonal signal model, black the one estimated with 'varyseasonal_N 3'.

The complete control file is stored in `estimatetrend_Bennett.ctl`. The result is plotted in Figure 8. More details can be found in Klos et al. (2017).

Another approach is to use a Wiener filter as described by Klos et al. (2019b). For this to work, the noise properties of the time series need to be known in addition to the values of the random part of the varying annual and semi-annual signal. The first can be obtained by running the Python script `analyse_timeseries.py`. However, one can simply use the Python script `apply_WF.py` which calls this script and does all the filtering. One can call is as:

```
apply_WF.py MAS1_2 0.10 0.06 0.9999
```

Here 0.10 and 0.06 are the standard deviations for the AR(1) random variations in the amplitude of the annual and semi-annual signal respectively. The value 0.9999 is the first order autoregressive coefficient (very close to 1 which is random walk). This script creates files in the `sea_files` and `fil_files` directory, containing the estimated varying signal and observations minus varying seasonal signal respectively. The result is plotted in Figure 9. Note that data gaps are set to zero which should be okay as long as the number of data gaps is smaller than around 10%.

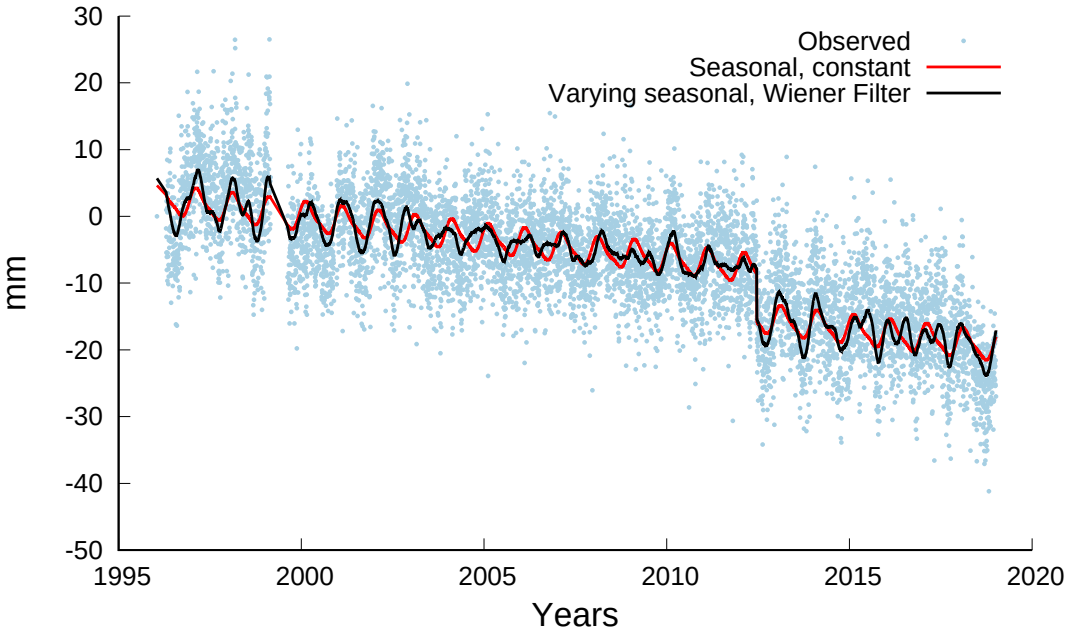

Figure 9: The fitted model to the MAS1 time series, vertical component. Red is the constant seasonal signal model, black the one estimated with the Wiener filter.

5 Model specification

We assume that the observations are the sum of a deterministic model and stochastic noise. This deterministic model is (Bevis and Brown, 2014):

$$\begin{aligned}\mathbf{x} = &\sum_{i=0}^{n_P} \mathbf{p}_i (t - t_R)^i + \sum_{i=1}^{n_J} \mathbf{b}_j H(t - t_j) \\ &+ \sum_{i=1}^{n_F} \mathbf{s}_i \sin(\omega_i t) + \mathbf{c}_i \cos(\omega_i t) \\ &+ \sum_{i=1}^{n_T} \mathbf{e}_i (1 - \exp(-(t - t_i)/T_i)) + \\ &+ \sum_{j=1}^{n_L} \mathbf{a}_j \log(1 + (t - t_j)/T_j)) + \\ &+ \sum_{k=1}^{n_T} \frac{\mathbf{u}_k}{2} \left[\tanh\left((t - t_k)/T_k\right)) - 1\right] + \\ &+ \sum_{l=1}^{n_l} a_l \times f_l(t)\end{aligned} \tag{6}$$

where $\mathbf{p}_i$ are the coefficients of a n_P degree polynomial. By default $n_P = 1$: a linear trend. t_R is normally the time that is exactly in the middle of start and end point of the time series. Following convention, we give the rate in unit per year, where a year

is defined as 365.25 days. Note that acceleration is normally defined to be twice the quadratic term. Thus, we fit $a(t - t_R)^2$ where a is estimated.

$H(t)$ is the Heaviside step function and used to model offsets with amplitudes $\mathbf{b}_j$.

An annual and semiannual signal are commonly included in the model and therefore have their own keywords 'seasonal' and 'halfseasonal'. In the equation their angular velocities are represented by ω_i Other periodic signals need to be entered using the keyword 'periodicsignals', followed by a list of their periods in days.

It is often desirable to speak about the amplitude of the annual and semi-annual period. To facilitate this, Hector shows also this parameter. It has been computed by first assuming a mean uncertainty for s_i and c_i which we call σ. In this case the amplitude follows a Rice distribution with a mean of $\sigma\sqrt{\pi/2}\, L_{1/2}(-\nu^2/(2\sigma^2))$ where $\nu = \sqrt{c_i^2 + s_i^2}$ and $L_{1/2}(x) = {}_1F_1(-1/2, 1, x)$.

The probability distribution function of the phase is more complicated. Therefore, simply a Monte Carlo simulation is performed using 10000 samples to get the mean and standard deviation of the phase.

5.1 Post-seismic deformation

Since version 1.6 one can also include post-seismic deformation in the model by extending the header of the data file by adding lines that contain the time when the relaxation starts (in MJD) and the relaxation period in days. An example is:

```
# log  51994.0   10.0
# log  55044.0   10.0
# exp  53544.0  100.0
```

One can use an exponential or logarithmic function, see Eq. 6. To estimate these post-seismic relaxations, one must set the keyword 'estimatepostseismic' to yes in estimatetrend.ctl. Note that one cannot chose between East, North or Up component. It is applied to all if a file with more than one component is used.

In most cases, one should estimate an offset (due to the earthquake) and the post-seismic deformation. Therefore, one normally has both in the header. An example is:

```
# offset 53544.0
# exp    53544.0  100.0
```

Since version 1.7 one can in addition model slow slip event using a hyperbolic tangent function (Larson et al., 2004). The two parameters t_k and T_k, representing the date of the mid-point of the slow slip event and its width respectively, must also be set in the header as follows:

```
# tanh 51994.0   10.0
```

One must add the keyword 'estimateslowslipevent yes' in estimatetrend.ctl. Note that one can consider this function as some kind of offset. Thus, no additional offset should be estimated. Of course this is just a general indication.

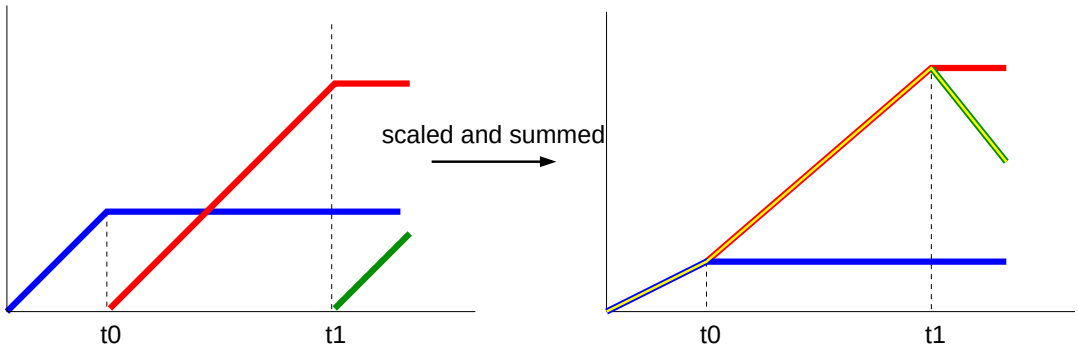

Figure 10: The left panel shows the three basis functions in blue, red and green that scaled and summed provide the piecewise linear function in yellow (3 segments).

5.2 Multi-trend

So far we have assumed that there is only one linear trend in the time series. However, it has become clear that some GNSS time series show a different linear trend after a major earthquake. In those cases a piecewise linear trend is more appropriate. To estimate the linear trend in each segment, the piecewise linear function is written as a sum of special functions that grow linearly in a particular segment and afterwards remain constant. For a piecewise linear trend with 3 segments, these functions are shown in blue, red and green in Figure 10. These functions will be stored in three columns of the design matrix. The final piecewise linear function is the sum of these three after scaling.

The time of a break in the linear trend is given in the header of the time series. Assuming that the two earthquake mentioned in the previous example were accompanied by a change in linear trend in the north and east component, then this is indicated by:

```
# break  51994.0 0
# break  51994.0 1
```

The indication of the component is not required if a file with only one component is used.

To estimate multi-trends, one must set the keyword 'estimatemultitrend' to yes in estimatetrend.ctl.

5.3 Geophysical Signal

The last term in Eq. 6 represents a scale factor a_l (also known as regression coefficient) times a function f_l. This can represent a geophysical signal. In example 2, we showed how surface pressure can be included in the analysis and how we obtained a scale value close to the inverted barometer value. Another example is the local pressure admittance in tidal gravity time series (van Dam and Francis, 1998). By specifying in the control file:

```
estimatemultivariate        yes
MultiVariateFile            XXXX.mom
```

This can be achieved. File XXXX.mom is in the mom format, MJD in the first column, but each following column can contain a separate geophysical signal for which a regression coefficient is estimated.

Note that this file should have the same time sampling as the observations and the XXXX.mom file should contain all epochs of the observations. The file can contain gaps, start before or end after the time span of the observation but it should contain all epochs for which observations are available.

5.4 Signal to Correct Observations

In some cases it is convenient to subtract another signal from the observations before performing the time series analysis. This can be done using the keywords `subtractsignal` and `SignalFile`. The latter is a normal mom-file where the second column contains the signal that will be subtracted from the observations. The first column contains the MJD values. An example of the extra lines in the control file is:

```
subtractsignal      yes
SignalFile          ./sig_files/signal.mom
```

Note that the time epochs need to be exactly the same as those in the original observation file.

6 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'. In addition, Hector accepts free format which means the exact number of digits or the spacing between the columns is flexible.

As was mentioned before, to make use of the Python scripts, the use of the 'mom' format is necessary.

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!

6.1 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.

6.2 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.

6.3 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:

$$MJD = floor(365.25 * (T - 1970) + 40587 + 0.1) - 0.5 \tag{7}$$

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 *but only for this neu-format*. Furthermore, only offsets can be put in the header, nothing about post-seismic deformation for which the mom or enu-format should be used. An example is for an offset for all components:

```
# 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.

6.4 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.

Since some years have 365 days while others 366, we need to introduce some definitions of our own about how to define the epochs in order to create a regularly spaced monthly time series. Inside Hector, these epochs are defined as:

$$MJD = 30.4375\,(12(year - 1859) + (month - 1)) + 59 \tag{8}$$

7 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 GNSS time series. Williams (2008) wrote the covariance matrix $\mathbf{C}$ of this particular combination as:

$$\mathbf{C} = \sigma^2 \left(\cos^2 \phi\, \mathbf{I} + \sin^2 \phi\, \mathbf{E}(d) \right) \tag{9}$$

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 which can have a value between 0 and $\pi/2$. The total variance of the noise is set by σ^2 (This is called the 'driving' noise in the output). This has been generalised using n-spheres for which the coordinates are defined as follows:

$$\begin{aligned} x_1 &= \cos(\phi_1) \\ x_2 &= \sin(\phi_1)\cos(\phi_2) \\ x_3 &= \sin(\phi_1)\sin(\phi_2)\cos(\phi_3) \\ &\vdots \\ x_{n-1} &= \sin(\phi_1)\ldots\sin(\phi_{n-2})\cos(\phi_{n-1}) \\ x_n &= \sin(\phi_1)\ldots\sin(\phi_{n-2})\sin(\phi_{n-1}) \end{aligned} \tag{10}$$

The last term of x_n is always a $\sin$. Thus, for $n = 2$ we have:

$$\begin{aligned} x_1 &= \cos(\phi_1) \\ x_2 &= \sin(\phi_1) \end{aligned} \tag{11}$$

For $n = 3$ we have:

$$\begin{aligned} x_1 &= \cos(\phi_1) \\ x_2 &= \sin(\phi_1)\cos(\phi_2) \\ x_3 &= \sin(\phi_1)\sin(\phi_2) \end{aligned} \tag{12}$$

In Hector the square of these coordinates are called 'fractions', $f_i = x_i^2$. Their sum is of course 1 because they are defined on a unit sphere. As a result, the diagonal of matrix $\mathbf{C}$ is filled with ones. We now generalise the previous equations as follows:

$$\mathbf{C} = \sigma^2 \left(f_1\mathbf{E}_1 + f_2\mathbf{E}_2 + f_3\mathbf{E}_3 + \ldots + f_n\mathbf{E}_n\right) \tag{13}$$

for n noise models which require $n-1$ angles $\phi_1, \ldots, \phi_{n-1}$. All 'fractions' 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.

7.1 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:

$$\gamma_i = 1 \qquad \text{for } i = 0 \tag{14}$$

$$= 0 \qquad i \neq 0 \tag{15}$$

Its one-sided power spectrum density is:

$$S(f) = 2\frac{1}{f_s} \tag{16}$$

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.

7.2 Power-law noise

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

$$\gamma_i = \frac{\Gamma(d+i)\Gamma(1-2d)}{\Gamma(d)\Gamma(1+i-d)\Gamma(1-d)} \tag{17}$$

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

$$S(f) = 2\frac{1}{f_s}\frac{1}{(2\sin(\pi f/f_s))^{2d}} \tag{18}$$

7.3 Power-law approximated

Bos et al. (2013) introduced a Toeplitz approximation for power-law noise which can be chosen using the label "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. This approximation is only valid for weakly non-stationary noise and therefore the minimal value of for the spectral index κ that is allowed is set to -1.6. Note that nowadays we prefer to use the Generalised Gauss Markov noise model with ϕ close to 1. This approximates power-law noise and better and is valid for spectral indices down to -2.

7.4 ARFIMA and ARMA

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

$$\Phi(L)(1-L)^d z_t = \Theta(L)\epsilon_t \tag{19}$$

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. (19) 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):

$$\Phi(L) = 1 - \phi_1 L - \phi_2 L^2 - \ldots - \phi_p L^p \tag{20}$$

$$\Theta(L) = 1 + \theta_1 L + \theta_2 L^2 + \ldots + \theta_q L^q \tag{21}$$

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 $\phi_1, \ldots, \phi_p$ 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. To specify AR(1) in 'estimatetrend.ctl' one must write:

```
NoiseModels         ARMA
AR_p                1
MA_q                0
```

7.5 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:

$$\gamma_i = \frac{\Gamma(d+i)(1-\phi)^i}{\Gamma(d)\Gamma(1+i)} \, {}_2F_1(d, d+i; 1+i; (1-\phi)^2) \tag{22}$$

This noise model can be used using the name 'GGM' after the keyword Noisemodels in `estimatetrend.ctl`. Bos et al. (2014) provide some additional formula's.

The $1-\phi$ parameter can be held fixed a priori by adding to the control-file:

```
GGM_1mphi           XXXX
```

where XXXX is the value you want to give this parameter. If it is small enough, 6.9e-06 is a good value, then GGM approximates a pure power-law model, see also next sub-section.

GGM makes use of the hypergeometric ${}_2F_1$ function and great care is required to keep the parameters that go into it within range to avoid numerical problems. This is shown in Figure 11. The spectal index κ should be larger than -3 (or $d < 1.5$) and for small values of 1-ϕ (ϕ close to 1), d should even be smaller which is implemented using a simple linear relation between ϕ and d. In this way, Hector should always be able to avoid over-flow of the function ${}_2F_1$ and keep the covariance matrix definite positive. If not, then try to increase the value of GGM_1mphi.

7.6 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. However, as was noted in the introduction, Hector can only deal with stationary noise because that results in Toeplitz covariance matrices that allow fast inversion techniques. This can be achieved by using the Generalised Gauss Markov noise model with a small value for the $1-\phi$ parameter. Example:

```
NoiseModels         FlickerGGM
GGM_1mphi           6.9e-06
```

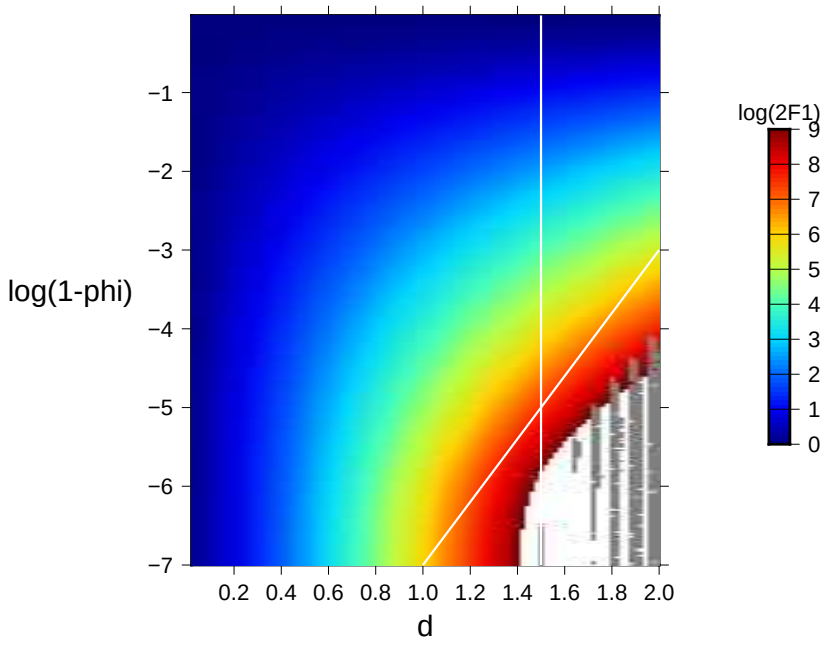

Figure 11: The region (d, $\log(1-\phi)$ space) for which the GGM noise parameters are valid. Below the lower white line one enters in the danger zone which is thus excluded in Hector. To be extra safe, also no values of $d > 1.5$ are allowed. The colour indicates the value of $\log({}_2F_1)$, again base 10, not $\ln$.

The value of 6.9×10^{-6} looks strange to many people. It is simply the smallest value that did not give numerical problems in previous versions of Hector. In version 1.9 a more stable subroutine for the Gaussian hypergeometric function is used (using the Boost library) but still, 6.9×10^{-6} is a good value for most cases.

In the past, people have analysed time series that caused Hector to use values of d (i.e. $-\kappa/2$) and ϕ outside the permitatable limits. In version 1.9 better limits have been implemented to avoid this problem. In Figure 11 the permitatable range of d and $\log_{10}(1-\phi)$ is shown, together with the associated $\log_{10}\ {}_2F_1$ value ($i = 10000$).

7.7 Matern noise model

The Matern noise model is well explained by Lilly et al. (2016). It is very similar to the Generalised Gauss Markov model and also has two parameters: the spectral index α and a constant λ. Of course α is our κ but with opposite sign. λ is related to ϕ of GGM and controls at which frequency of the spectrum occurs. One or both parameters can be kept fixed to specific values in the control file `estimatetrend.ctl`. For example:

```
NoiseModels         Matern
kappa_fixed         -0.8
lambda_fixed        0.01
```

Lilly et al. (2016) describe how one can simulate synthetic noise time series by performing a Cholesky decomposition and multiplying the decomposed matrix $\mathbf{U}$ with

the vector $\mathbf{w}$ which contains white noise (Gaussian random variables). In Hector synthetic noise is created by convolving white noise with an impulse function (Kasdin, 1995).

These two approaches differ in their treatment of the lack of values before the first observation. The Cholesky approach, to ensure the desired variance and co-variance values are produced, adjusts the impulse function coefficients (the rows in matrix $\mathbf{U}$). The impulse function approach keeps its coefficients constant and thus will not produce the desired variance and co-variance values at the start of the time series. After some time, the two methods converge to the same result. A consequence of this is that the decomposed upper triangular $\mathbf{U}$ becomes more and more Toeplitz (values on each diagonal are the same) and these become identical with the impulse response coefficients, see also Bos et al. (2013). We performed a Cholesky decomposition of the covariance matrix and take the last row of the matrix $\mathbf{U}$ as our impulse response coefficients.

7.8 VaryingPeriodic (Band-pass) noise model

The amplitude of the annual signal is mostly not exactly constant over time but varies a little in a random manner. Amplitude modulation of a periodic signal causes that the pure peak in the power-spectrum plot widens a bit. This can be viewed as a separate type of noise which can be added to the other noise models in the analysis.

Langbein (2004) introduced a band-pass spectrum that captures this widening effect in the power-spectrum. From this spectrum one can compute the impulse response coefficients h_i from which one can again compute the autocovariance function. However, to maintain a Toeplitz covariance matrix that will ensure that we can continue to use all our numerical tricks to speed up the computations, we use a variant. We use the results of Klos et al. (2019a) where the amplitude modulation of the annual signal is modelled by a first order autoregressive model, AR(1). The corresponding autocovariance function is:

$$\gamma_i = \sigma^2 \frac{\phi^k}{2(1-\phi^2)} \cos(\omega_0 k) \tag{23}$$

Besides an annoying overloading of the symbol ϕ for yet another noise model, this is just the autocovariance of the AR(1) process multiplied by a cosine and divided by a factor 2. ω_0 is the period of the periodic signal (here one year). In Hector one can specify varying annual and semiannual signals as follows:

```
Noisemodels         VaryingAnnual VaryingSemiAnnual
phi_varying_fixed   0.999
```

As usual, one can estimate the value of ϕ or set it fixed by including `phi_varying_fixed` in the control file. Using the fact that $\cos(a)\cos(b) = [\cos(a+b) + \cos(a-b)]/2$, we have for the spectrum:

$$S(f) = \frac{\sigma^2}{f_s}\left[\frac{1}{1-2\phi\cos(\omega+\omega_0)+\phi^2} + \frac{1}{1-2\phi\cos(\omega-\omega_0)+\phi^2}\right] \tag{24}$$

8 The Akaike and Bayesian Information Criteria

Hector allows the estimation of linear trend or a higher order polynomial, seasonal and other periodic signals and a variety of noise models which can be combined. To chose

the best model one can make use of the Akaike or Bayesian Information Criteria (Akaike, 1974; Schwarz, 1978). Both use the log-likelihood as their starting point but add penalties for adding parameters in order to avoid overfitting.

The definition of the log-likelihood is:

$$\ln(L) = -\frac{1}{2}\left[N\ln(2\pi) + \ln\det(\mathbf{C}) + \mathbf{r}^T\mathbf{C}^{-1}\mathbf{r}\right] \tag{25}$$

where N is the actual number of observations (gaps do not count). The covariance matrix $\mathbf{C}$ is decomposed as:

$$\mathbf{C} = \sigma^2\bar{\mathbf{C}} \tag{26}$$

where $\bar{\mathbf{C}}$ is the sum of various noise models and σ the standard deviation of the 'driving' white noise process as was explained in section 7. σ is estimated from the residuals:

$$\sigma = \sqrt{\frac{\mathbf{r}^T\bar{\mathbf{C}}^{-1}\mathbf{r}}{N}} \tag{27}$$

Using this relation and the fact that $\det c\mathbf{A} = c^N \det \mathbf{A}$, the following formulation for the likelihood is implemented:

$$\ln(L) = -\frac{1}{2}\left[N\ln(2\pi) + \ln\det(\bar{\mathbf{C}}) + 2N\ln(\sigma) + N\right] \tag{28}$$

The number of parameters k is the sum of parameters in the Design matrix $\mathbf{H}$ and the noise models and the variance of the driving white noise process. For example, for estimating a linear trend using power-law + white noise models 5 parameters are involved: nominal bias, linear trend, distribution of variances between power-law and white noise, spectral index of the power-law and the variance of the driving white noise process ($k = 2 + 2 + 1 = 5$).

The Akaike Information Criterion (AIC) and Bayesian Information Criterion (BIC) are defined as:

$$\text{AIC} = 2k + 2\ln(L) \tag{29}$$

$$\text{BIC} = k\ln(N) + 2\ln(L) \tag{30}$$

$$\text{BIC}_{tp} = k\ln(N/(2\pi)) + 2\ln(L) \tag{31}$$

The preferred model is the one with the minimum AIC/BIC value. Note that these are relative measures between various choices, not absolute criteria. We have added BIC_{tp} as a third option which lies somewhere between AIC and BIC. For more details, see He et al. (2019).

In the output of `estimatetrend` one can also see a `BIC_c`. This is used by `findoffset` It is `BIC` plus some extra penalties.

9 Auxiliary Python scripts

In previous versions of Hector, a few Tcl-scripts were provided to perfom conversion of time series formats. However, this wonderful and underappreciated scripting language is not adopted by the scientific community which favours Python instead. Since the

Abbreviation	Description
WN	White noise
FN	Flicker noise
PL	Power-law noise
RW	Random Walk noise
GGM	Generalised Gauss-Markov noise model
AR1	ARMA(1,0) first-order autogressive noise model
MT	Matern noise model
VA	Varying Annual (Bandpass type noise)
VSA	Varying Semi-Annual (Bandpass type noise)

purpose of making the Hector package available to the scientific community is that they actually use it, we have now converted to Python and added some additional scripts which will simplify the various steps of the analysis process.

In all these scripts, the same abbreviations for the noise models are used which are listed in Table 9. Combinations of noise models abbreviations is also allowed in most cases. For example, PLWN is a common combination for the analysis of GNSS time series.

9.1 analyse_timeseries.py

The steps of running 'removeoutliers' followed by 'estimatetrend' is very common and has been implemented in the auxiliary Python script `analyse_timeseries.py`. It always estimates offsets and includes an annual and semi-annual signal. The correct usage is:

```
analyse_timeseries.py station_name {GGM|MT|PL|FN|RW|WN|AR1|VA|VSA}+
```

where station name is actually the filename without the `.mom` extension stored in the `./obs_files/` directory. The second argument is the combination of noise models.
If we go to the directory `./ex1` and run the command:

```
analyse_timeseries.py TEST PLWN
```

Two JSON files are created, called `removeoutliers.json` and `estimatetrend.json`. In Hector 1.7.2 the output was returned in a long list of variables which was not very flexible. JSON output is easier to understand and becoming the standard for exchanging data on the internet. The first JSON file has the following contents:

```
{
  "N" : 1000,
  "gap_percentage" : 10,
  "outliers" : ["1996-01-18T00:00:00.000Z","1996-02-18T00:00:00.000Z",
  "1996-05-10T00:00:00.000Z","1998-08-07T00:00:00.000Z",
  "1998-09-09T00:00:00.000Z","1998-09-21T00:00:00.000Z"]
}
```

which list the date of observations considered outlier in the ISO8601 format. This should facilitate parsing it into databases. The file `estimatetrend.json` contains:

```
{
  "N" : 1000,
  "gap_percentage" : 10.6,
  "ln_L" : -1726.23,
  "AIC" : 3478.47,
  "BIC" : 3540.81,
  "BIC_tp" : 3516.92,
  "BIC_c" : 3578.04,
  "ln_det_I" : 29.2236,
  "NoiseModel" : {
    "GGM" : {
      "sigma" : 3.52812,
      "d" : 0.399311,
      "kappa" : -0.798621,
      "1-phi" : 6.9e-06,
      "fraction" : 0.453206
    },
    "White" : {
      "sigma" : 1.19309,
      "fraction" : 0.546794
    }
  },
  "driving_noise" : 1.61347,
  "trend" : 16.4165,
  "trend_sigma" : 0.704617,
  "Sa_cos" : 4.10265,
  "Sa_cos_sigma" : 0.288988,
  "Sa_sin" : -3.9375,
  "Sa_sin_sigma" : 0.298971,
  "Ssa_cos" : -0.00201704,
  "Ssa_cos_sigma" : 0.207547,
  "Ssa_sin" : -0.052321,
  "Ssa_sin_sigma" : 0.210893,
  "jumps_epochs" : ["1996-07-20T00:00:00.000Z","1996-09-08T00:00:00.000Z",
     "1997-12-02T00:00:00.000Z","1998-08-09T00:00:00.000Z"]
  "jumps_sizes" : [24.5091,-24.7139,-39.7924,38.5336],
  "jumps_sigmas" : [0.714333,0.779712,0.72997,0.76003]
}
```


Most fields are self explanatory except Sa and Ssa which denote the yearly and half-yearly period using Doodson numbers (tides). Note that this script uses the Generalized Gauss Markov model with a small value of 1-phi to simulate power-law noise. This explains the small differences with the values discussed in example 1, see section 4.1. As a result, the files `estimatetrend.ctl` and `removeoutliers.ctl` are now different.

This file also lists the 'fraction' values. These need to be used when one wants to create synthetic time series using `simulatenoise` with noise properties that are similar to those observed in the real time series.

9.2 analyse_and_plot.py

The script `analyse_timeseries.py` is called by the script`analyse_and_plot.py` which performs the following steps for all stations listed in the directory `./obs_files`:

1. remove outliers in time series.
2. run `estimatetrend`.
3. plot the time series together with the fitted model.
4. estimate power spectral density of residuals and of the fitted noise model.
5. make a power spectral density plot.

The correct way to call this script is:

```
analyse_and_plot.py {GGM|MT|PL|FN|RW|WN|AR1|VA|VSA}+ [station]
```

The last argument is optional. If given, then it will only analyse and plot the result for the given station. Otherwise, all station found in the `./obs_files` will be analysed. The output of `estimatetrend` is stored in `./mom_files`. The data plots in `./data_figures` and power spectral density plots in `./psd_figures`.

This plot also produces 2 JSON files calle `hector_removeoutliers.json` and `hector_estimatetrend.json`. These contain, for each station, the JSON output generated by `analyse_timeseries.py`.

9.3 find_offset.py

In the introduction we mentioned that Hector now comes with a program to detect the epoch when an offset occured: `findoffset`. So far we have not discussed this program. The reason is that this program only searches for the epoch which is the most likely to have an offset. Normally one is interested to find *all* offsets in a time series. Therefore, the program `findoffset` must be called several times until some threshold has been reached which indicates that it is likely that there are no more offsets in the time series. This task is performed by the script `find_offset.py`. The way to call it is:

```
find_offset.py station_name PLWN|FNWN|RWFNWN|WN [3D]
```

As before, the time series are assumed to be stored in the directory `./raw_files`, in the mom-format. The third argument, 3D, is optional and meant for finding offsets using the East, North and Up component time series in a single analysis process. For this to work, the filenames should be following the convention XXX_[012] where XXX is the station name and 0, 1 or 2 stands for East, North or Up. For example, `KOTG_0.mom`, `KOTG_1.mom` and `KOTG_2.mom` would be three valid filenames (all stored in directory `./raw_files`).

Returning to our example 1 presented in 4.1, the last lines printed after running the command

```
find_offset.py TEST PLWN
```

are:

```
...
0 MJD=       0.0 BIC_c=   5147.78
1 MJD=   50784.0 BIC_c=   5031.45
2 MJD=   51034.0 BIC_c=   4910.31
3 MJD=   50284.0 BIC_c=   4840.55
4 MJD=   50335.0 BIC_c=   4722.66
5 MJD=   51065.0 BIC_c=   4731.53
Computation time in seconds: 21.486539
```

The list the epoch (given in Modified Julian Dates) and BIC_c value for having zero to five offsets. Looking at the value of BIC_c, the lowest value is obtained for 4 offsets which is therefore chosen as the maximum number of offsets found. The time of these offsets is added to the mom-file and stored in the directory `./raw_files`. Note that only the offset at MJD=50335 is one day of with the real offset specified and listed in the header of TEST.mom in `./obs_files`.

9.4 find_all_offsets.py

For the finding all offsets in a network of GNSS stations, one can use the small script `find_all_offsets.py`. It simply reads all station names in the directory `./raw_files` and calls for each of them `find_offset.py`, using PLWN noise model combination and the '3D' option.

For now, offset detection is only implemented for the 'AmmarGrag' method and not 'FullCov'. If 'NumericalMethod' is not specified in the control file, then Hector will automatically select 'FullCov' when the amount of missing data is largen than 50%. Thus, if the user really wants to detect offsets for time series with lots of data gaps, the line 'NumericalMethod AmmarGrag' needs to be added to the control file of `estimatetrend`.

9.5 apply_WF.py

In section 5 we explained that Hector can fit various periodic signals to the observations with fixed amplitude and phase-lag. However, in some time series, the amplitude of the annual and semi-annual signal varies from year to year. To model this, Klos et al. (2019b) describe a simple Wiener Filter that provides a good estimate of the varying part of the seasonal signal. This has been implemented in the script `apply_WF.py`. It makes again use of our core script `analyse_timeseries.py` to estimate the constant part of the seasonal signal, to remove the trend and to estimate the noise parameters using a PLWN noise model combination. The estimated varying seasonal signal is stored in the directory `./sea_files`. The observations minus the varying seasonal signal (i.e. the filtered signal) is stored in the directory `./fil_files`.

The correct way of calling the program is:

```
apply_WF.py station_name sigma_a sigma_sa phi
```

where `station_name` is the name of the file, mom-format, stored in the `./obs_files` directory, without the `.mom` extension. `sigma_a` and `sigma_sa` are the standard deviations of the AR(1) processes that describe the variations in the amplitude of the annual and semi-annual signal respectively. `phi` is the first order autoregressive, thus AR(1),

coefficient. It is normally close to 1, almost equal to random walk. For more details, see Klos et al. (2019b). Note that data gaps are set to zero in this script which corresponds to zero random variations in the seasonal signal. This should be okay as long as the percentage of data gaps is below 10%.

10 Quick reference for the control-files

10.1 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 offset information (optional)
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
DegreePolynomial	degree of polynomial: 0-6, (optional, default=1)
estimatemultitrend	yes|no (optional, default=no. If yes, then DegreePolynomial keyword is ignored. Only linear trends are estimated)
estimatepostseismic	yes|no (optional, default=no)
estimateslowslipevent	yes|no (optional, default=no)
seasonalsignal	yes|no
halfseasonalsignal	yes|no
periodicsignals	a sequence of numbers representing 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
estimatemultivariate	yes|no (optional, default=no)
MultiVariateFile	Name of mom-file with geophysical to be included in the analysis.
subtractsignal	yes|no (optional, default=no)
MultiVariateFile	Name of mom-file with signal to be subtracted from the observations.
JSON	yes|no (optional, default=no. If yes, then file removeoutliers.json is created)

Note that since version 1.7.2 this program also produces a file called `removeoutliers.out` that contains the epoch of the outliers, one on each line. It purpose is to facilitate importing this information into other programs. However, one should consider using the

JSON file removeoutliers.json for this purpose.

10.2 estimatetrend.ctl and findoffset.ctl

This file is read by `estimatetrend` and `findoffset`.

Keyword	Value(s)
DataFile	name of file with observations
DataDirectory	directory where file with observations is stored
OffsetFile	name of file with offset information (optional)
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
DegreePolynomial	degree of polynomial: 0-6, (optional, default=1)
estimatemultitrend	yes|no (optional, default=no. If yes, then DegreePolynomial keyword is ignored. Only linear trends are estimated)
estimatepostseismic	yes|no (optional, default=no)
estimateslowslipevent	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, FlickerGGM, RandomWalkGGM, Powerlaw, PowerlawApprox, Matern, ARFIMA, ARMA and GGM.
LikelihoodMethod	chose one from: AmmarGrag 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)
GGM_1mphi	value of $1-\phi$ (optional, only for GGM)
kappa_fixed	keep value of spectral index fixed to given value (optional, only for Powerlaw, Matern and GGM)
lambda_fixed	keep value of lambda fixed (optional, only for MAtern)
TimeNoiseStart	number of days before the start of the observations when it is assumed that the noise started (only for PowerlawApprox)
RandomiseFirstGuess	yes|no (optional, default=no)
estimatevaryingseasonal	yes|no (optional, default=no)
varyingseasonal_N	degree of polynomial (normally between 1 and 8)
estimatemultivariate	yes|no (optional, default=no)
MultiVariateFile	Name of mom-file with geophysical to be included in the analysis.
subtractsignal	yes|no (optional, default=no)
MultiVariateFile	Name of mom-file with signal to be subtracted from the observations.
JSON	yes|no (optional, default=no. If yes, then file estimatetrend.json or findoffset.json is created)

10.3 estimatespectrum.ctl

This file is read by `estimatespectrum` which creates a file called `estimatespectrum.out` if keyword "OutputFile" is not set.

Keyword	Value(s)
DataFile	name of file with observations
DataDirectory	directory where file with observations is stored
OutputFile	name of file where computed spectra will be saved (optional, default="estimatespectrum.out")
interpolate	yes|no
ScaleFactor	a number to scale the observations (optional, default=1)
WindowFunction	Parzen|Hann (Parzen is default)
Fraction	number between 0 and 1 (0.1 is default).

10.4 modelspectrum.ctl

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

Keyword	Value(s)
OutputFile	name of file where computed spectra will be saved (optional, default="modelspectrum.out")
interpolate	yes|no
NoiseModels	chose any set from: White, Flicker, RandomWalk, FlickerGGM, RandomWalkGGM, Powerlaw, Matern, ARFIMA, ARMA and GGM.
AR_p	number of AR coefficients (only for ARFIMA or ARMA)
MA_q	number of MA coefficients (only for ARFIMA or ARMA)
GGM_1mphi	value of $1-\phi$ (optional, only for GGM)
PhysicalUnit	the physical unit of the observations

10.5 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
NoiseModels	chose any set from: White, Flicker, RandomWalk, FlickerGGM, RandomWalkGGM, Powerlaw, PowerlawApprox, ARFIMA, ARMA and GGM.
AR_p	number of AR coefficients (only for ARFIMA or ARMA)
MA_q	number of MA coefficients (only for ARFIMA or ARMA)
GGM_1mphi	value of $1-\phi$ (optional, only for GGM)
TimeNoiseStart	number of days before the start of the observations when it is assumed that the noise started (only for PowerlawApprox)
RepeatableNoise	(yes/no). Each time 'simulatenoise' is run, the same synthetic time series are created. The default is no

11 Advanced Features

11.1 Offsets

The information about the time of an offset 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
```

This external file with time of offsets can be used with `removeoutliers` and `estimatetrend`.

12 License

References

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

Akaike, H. (1974). A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716–723.

Bennett, R. A. (2008). Instantaneous deformation from continuous gps: Contributions from quasi-periodic loads. *Geophysical Journal International*, 174(3):1052–1064.

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

Bevis, M. and Brown, A. (2014). Trajectory models and reference frames for crustal motion geodesy. *Journal of Geodesy*, 88:283–311.

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. (2013). Fast Error Analysis of Continuous GNSS Observations with Missing Data. *J. Geod.*, 87(4):351–360.

Bos, M. S., Williams, S. D. P., Araújo, I. B., and Bastos, L. (2014). The effect of temporal correlated noise on the sea level rate and acceleration uncertainty. *Geophysical Journal International*, 196:1423–1430.

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.

He, X., Bos, M. S., Montillet, J. P., and Fernandes, R. M. S. (2019). Investigation of the noise properties at low frequencies in long gnss time series. *Journal of Geodesy*.

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.

Klos, A., Bos, M. S., and Bogusz, J. (2017). Detecting time-varying seasonal signal in gps position time series with different noise levels. *GPS Solutions*, 22(1):21.

Klos, A., Bos, M. S., Fernandes, R. M., and Bogusz, J. (2019a). Noise-dependent adaption of the wiener filter for the gps position time series. *Mathematical Geosciences*, 51(1):53–73.

Klos, A., Bos, M. S., Fernandes, R. M. S., and Bogusz, J. (2019b). Noise-dependent adaption of the wiener filter for the gps position time series. *Mathematical Geosciences*, 51(1):53–73.

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

Langbein, J. (2010). Computer algorithm for analyzing and processing borehole strainmeter data. *Computers & Geosciences*, 36(5):611–619.

Langbein, J. (2017). Improved efficiency of maximum likelihood analysis of time series with temporally correlated errors. *Journal of Geodesy*, 91(8):985–994.

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.

Larson, K. M., Lowry, A. R., Kostoglodov, V., Hutton, W., Sánchez, O., Hudnut, K., and Suárez, G. (2004). Crustal deformation measurements in guerrero, mexico. *Journal of Geophysical Research: Solid Earth*, 109(B4).

Lilly, J. M., Sykulski, A. M., Early, J. J., and Olhede, S. C. (2016). Fractional brownian motion, the matérn process, and stochastic modeling of turbulent dispersion. *arXiv preprint arXiv:1605.01684*.

Mandelbrot, B. B. and Van Ness, J. W. (1968). Fractional Brownian motions, fractional noises and applications. *SIAM-REVIEW*, 10(4):422–437.

Montillet, J.-P. and Bos, M. S. (2019). *Geodetic Time Series Analysis in Earth Sciences*. Springer.

Schwarz, G. (1978). Estimating the Dimension of a Model. *The Annals of Statistics*, 6(2):461–464.

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

van Dam, T. M. and Francis, O. (1998). Two years of continuous measurements of tidal and nontidal variations of gravity in boulder, colorado. *Geophysical research letters*, 25(3):393–396.

Williams, S. D. P. (2003). The effect of coloured noise on the uncertainties of rates from geodetic time series. *J. Geod.*, 76(9-10):483–494.

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 optional. 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 (removed from version 1.5 onwards).

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 Generalised 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. (has been solved in version 1.6 and now the limit is closer to 0.999999)

4. The binaries are now stored in `/usr/local/bin` instead of `/usr/bin` which we 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

A.5 Version 1.5

1. Removed the option to apply first difference to data. This option was never used and it makes the source code unnecessary complicated.

2. Added Hann window to `estimatespectrum`. You can now chose between the Hann and Parzen window function and select the percentage of data to which this window will be applied at both ends of the segment.

3. Implemented a Taylor expansion of the Hypergeometric function in GenGauss-Markov.cpp and now ϕ can be closer to 1: 0.9999. This helps to simulate pure power-law noise.

4. Changed the output of estimatetrend by eliminating parameters d, driving noise and fractions. The noise amplitudes now follow CATS.

5. The Plate Boundary Observatory changed their time series file format. We find it easier to write a little script to convert the new format to my mom-format than to update the Observations.cpp class.

6. Removed subsection "Some additional tests" since it was not read or created confusion for those who did.

A.6 Version 1.5.1

1. Removed a small bug in the class Minimizer.cpp that left the first parameter 0.001 larger than the optimal value (residual of computing Fisher information matrix). It had a large effect if "Random Walk" was the first chosen noise model.

2. Added to the manual that you can set the $1 - \phi$ value a priori in the control-file.

A.7 Version 1.5.2

1. Still problem of having spaces in list of items in Control.cpp which now hopefully has been solved.

A.8 Version 1.6

1. Introduced logarithmic and exponential post-seismic relaxation in Designmatrix.

2. Generalised linear and quadratic trend to N degree polynomial. To keep maintenance simple, I removed the 'QuadraticTerm' keyword.

3. Number of allowed periodical signals in the estimation process has been increased from 20 to 40.

4. One can select to estimate multi linear trends instead of a singly polynomial.

5. `estimatespectrum` and `modelspectrum` now allow the keyword "OutputFile" to redirect their output to another file. Note that now the default control-file for `modelspectrum` is `modelspectrum.ctl` to be consistent.

6. Removed memory leaks.

7. Added program FindOffset.

A.9 Version 1.7.2

1. Added the program `findoffset`.

2. Added Python scripts: analyse_timeseries.py and analyse_and_plot.py.

3. Use generalised spheres to compute fractions which solves the problem in older versions of Hector when all fraction values went haywire when one of the fraction values approached zero.

4. Added hypergeometric tangent to model slow slip events.

5. The MLE performs a numerical minimisation search. The initial guess of the noise parameter values is fixed. In this version the starting values are allowed to vary a bit randomly each run when the keyword 'RandomiseFirstGuess yes' is set in the control file. So far, we have not encountered any benefit of this but rumours have it that CATS, and perhaps Hector, sometimes do not converge (it would help us if people send us their problematic time series). This extra randomisation helps to

prove, by running Hector several times and always finding the same answer, that a unique solution has been found.

6. `removeoutliers` now produces another file, `removeoutliers.out`, that contains the epoch of the found outliers.

A.10 Version 1.9

1. There is no 1.8 because that was a disaster. Reason for this disaster was that ATLAS is no longer maintained by MacPorts. It simply does not install. On CentOS it still installs but now GSL is conflicting with it because of slightly different CBLAS headers. My attempts to continue with ATLAS and GSL failed.

2. The source code has been adapted to make use of the OpenBlas library instead of ATLAS. Furthermore, I now use the Boost library to generate random numbers and compute hypergeometric functions 1F1 and 2F2. For complex numbers, I still use my own subroutine. Some open source code was used to find roots of polynomials and for the Nelder & Mead method. Many thanks to Bill Hallahan and R ONeill / John Borkardt for making the code for these subroutines freely available on the web!

3. Added the Matern noise model.

4. `modelspectrum` now can perform a Monte Carlo simulation (i.e. producing many synthetic time series with given noise properties which are then submitted to `estimatespectrum`) to get an idea of how well one can estimate the spectrum.

5. More Python scripts are included. Now these script can be used to convert the output of Hector in a Json file which should facilitate its parsing into other programs.

6. Created Dockerfiles in order to quickly cross-compile Hector to other Linux Distrubutions. It is working fine but I had hoped people would ask me for Docker images which so far has not happened. Thus, the full potential of Dockers has not yet been achieved.

A.11 Version 2.0

1. Bugs in multivariate subroutine were corrected. The problem occured when the multivarite file ended on the same date as the observations which causes a crash. If no crash occured, then values were correct.

2. Multi-trend now has proper piecewise linear segments without the need to add offsets on the date when the trend changes.

3. The phase of estimated periodic signals has now a standard deviation.

4. `simulatenoise` now always creates different synthetic time series when it is run. To create the same synthetic time series, one must set the keyword 'RepeatableNoise' to yes.