Create and Analyse Time Series : CATS software V3.1.1

Simon Williams

July 11, 2005

1 Introduction

cats is a program to use Maximum Likelihood Estimation to fit a multi-parameter model to a time series (such as continuous GPS). The routine solves for all parameters simultaneously, but in two parts to increase speed; the linear part includes an offset and slope, the possibility of abrupt steps (as from earthquakes) and sinusoidal terms (for example annual and semiannual terms), while the non-linear part solves for several specific noise models and combinations.

- White noise.
- Power-law noise (otherwise known as fractionally integrated noise)
- First-Order Gauss Markov noise (equivalent to autoregressive AR(1) noise)
- Band-pass noise (See *Langbein* [2004])
- Generalized Gauss Markov noise (See *Langbein* [2004])
- Variable white noise (i.e. using the formal errors)
- Step-variable white noise (i.e. a change in the scale of white noise between two epochs)
- Time-variable white noise (i.e. an exponential decay in the scale of the white noise)

The background to this program can be found in the following papers, *Langbein and Johnson* [1997], *Zhang et al.* [1997], *Mao et al.* [1999], *Williams* [2003], *Williams et al.* [2004] and *Langbein* [2004]. The program is command-line orientated, so it does not require any other files except for the time series file in order to run. For the power-law noise models, **cats** uses the fractional difference method [*Hosking*, 1981] for creating the covariance matrix. *Zhang et al.* [1997] and *Mao et al.* [1999] used an approximate covariance matrix for flicker noise derived from an algorithm described by *Gardner* [1978] for simulating such noise (H. Johnson, pers. comm., 1996). The constants in this matrix were chosen so that the power spectrum of flicker noise and random walk noise cross at a period of one year when both have a sampling interval of one day and equal amplitude ($b_{-1} = b_{-2}$). This earlier covariance matrix is not exactly the same as that derived from the above transformation matrix. The major difference is in the scaling of the amplitude. The scaling between the "new" and "old" amplitudes can be estimated from their respective power spectrum equations. That is

$$\begin{aligned} P_{old} &= \frac{b_{old}^2 f^{-1}}{2\pi^2} \\ P_{new} &= \frac{b_{new}^2 f^{-1}}{\pi\sqrt{f_s \times 24 \times 60 \times 60 \times 365.249}} \end{aligned} \tag{1}$$

therefore

$$b_{new} = \frac{(f_s \times 24 \times 60 \times 60 \times 365.249)^{1/4}}{\sqrt{2\pi}}\, b_{old}. \tag{2}$$

When the sampling frequency is once per day then

$$b_{new} = 1.7440\, b_{old}. \tag{3}$$

Therefore, values quoted for the amplitude of flicker noise in papers that use the *Zhang et al.* [1997] covariance matrix must be scaled before being used in any studies using the new matrix.

In the formulisation used in this program, the system is scaled so that for any spectral index, the power spectra cross over at the same frequency given the same sampling frequency and the same size σ. For daily sampling, the cross over frequency is about 120 days. This also means that for spectral indices other than random walk and white noise the scaling parameter changes with different sampling frequencies. This must be taken into account when comparing results from daily solutions to those from weekly solutions or if the data has been decimalised. For instance, given flicker noise and a sampling interval of 2 days, the flicker noise

parameter will be $2^{1/4}$ times smaller than the daily value. John Langbein uses a different scheme again so that flicker noise amplitudes from this program will be 4.3717 times larger than his results.

2 The Time Series File

There are currently two accepted formats for entry; a "cats" file and a "psmsl" file. In the future, other formats may become available. Users are free to suggest new formats (writing the code would be even better - see read_series.c in the lib directory for examples)

2.1 "cats" file description

The data file is fairly free format consisting of two parts, header information and the time series. The header information is basically a list of parameters relevant to the GPS site the time series comes from. The header information (plus any line you want to eliminate from the processing) starts with a # symbol. The only header information relevant to **cats** is the list of offsets. The data set consists of seven columns (no particular format) corresponding to, in order, time (in decimal years) north, east, up, north error, east error, up error. The positions are, by default, assumed to be in metres, however this can be altered with the -Sscale_factor option. At present the position uncertainties are not used in the program. In the future the individual formal errors may be used to define a variable white noise component. However this slows the algorithm down considerably when compared to assuming the white noise is the same at each epoch. If, for instance, there was only one component of information required for processing (for example tide gauge data) then the other columns must still be present but the program can be told not to process those columns by using the --columns (-C) option described below. An example of a file following this format is shown below.

```
# Site : vyas
# X :  -2483507.1216
# Y :  -4672361.9853
# Z :   3549320.5426
# Latitude : 34.030915
# Longitude : -117.992047
# Height : 7.856557
# offset 1999.79178082 7
1998.5736   -0.02370    0.04070    0.00290  0.0009  0.0009  0.0037
1998.5763   -0.02360    0.04190    0.00410  0.0009  0.0009  0.0030
1998.5791   -0.02180    0.04140    0.00650  0.0010  0.0009  0.0038
```

```
1998.5818   -0.02270    0.04100   -0.00130  0.0009  0.0009  0.0038
1998.5845   -0.02330    0.04070    0.00050  0.0009  0.0009  0.0037
1998.5873   -0.02300    0.04050   -0.00020  0.0009  0.0009  0.0038
1998.5900   -0.02320    0.04090    0.00560  0.0009  0.0009  0.0037
1998.5928   -0.02390    0.04030   -0.00180  0.0010  0.0009  0.0039
1998.5955   -0.02370    0.04020    0.00230  0.0009  0.0009  0.0037
1998.5983   -0.02320    0.04150    0.00480  0.0010  0.0009  0.0038
1998.6010   -0.02390    0.04020   -0.00620  0.0010  0.0009  0.0039
1998.6036   -0.02240    0.03970   -0.00140  0.0010  0.0009  0.0038
1998.6091   -0.02190    0.03990   -0.00190  0.0010  0.0009  0.0030
```

If there any known discontinuities or offsets in the time series then these should be specified in the header of the file in the following format

```
# offset decimal_date component_code
```

The component code is calculated in a manner similar to file permissions on a unix-style system. The component code is the sum of the components in which the offsets appears with the values north (4), east (2) and vertical (1). This gives an exclusive number between 0 and 7 for all combinations of affected components. For example for an offset occurring in the north and up components the component code is 5. For all components the value is 7. The offset line is repeated for as many offsets as there are thought to be in the time series. The offsets need not be in chronological order nor do you even have to ensure that they are within the time span of the data set. The program will sort the offsets correctly when processing the file.

2.2 “psmsl” file description

This type of data file is again very simplistic and is described on the Permanent Service for Mean Sea Level web site (http://www.pol.ac.uk/psmsl/datainfo). It basically consists of one two main columns. Column one is the time stamp and column two is the data. **Cats**, since it is written in C, is not so specific about the exact format, it just tries to read 3 columns on each line. If there are two columns (time and data) then it uses that epoch. An example of a data file is given below.

```
1964.625  7036
1964.708  7103
1964.792  7057
1964.875  7139
1964.958  7091
1965.042  7072
1965.125  6834
1965.208  6935
1965.292  6941
1965.375  6920
1965.458  7011
1965.542  6974 12
```

```
1965.625  7026  8
1965.708  7078  3
1965.792  7078  3
1965.875  7075
1965.958  7115
1966.042  6990
1966.125  7087
1966.208  7005
```

As for the "cats" format, this file format has been adapted so that it can read in offsets and ignore lines that begin with the # symbol.

3 Command Line Options

The command line options now come in two "flavours" depending on whether your machine architecture can handle long options or not. In the following section the long options will be described with the short option listed in brackets. There is a slight difference between the version that can accept long options to the version that cannot. The long option version would look something like this

```
cats vyas.neu --sinusoid 1y1 --verbose --columns 4 --output vyas_all.mle
```

or

```
cats vyas.neu --sinusoid=1y1 --verbose --columns=4 --output=vyas_all.mle
```

or

```
cats vyas.neu -A 1y1 -V -C4 -Ovyas_all.mle
```

It can have a mixture of long and short. If using long options, then options with additional parameters should either have a space or an equal sign between the option and the parameter/s. If you choose to use short options then the additional parameter should follow on from the option (e.g. -C4) or there should be a space between (e.g. -C 4). For the version that only accepts short options then the options should always take the form

```
cats vyas.neu -A1y1 -V -C4 -Ovyas_all.mle
```

with no gaps between option and the additional parameters.

--model (-M) A description of the stochastic model. The first two characters in the string indicate the type of stochastic model to use. These include **pl** for power-law noise, **gm** for first-order gauss markov, **wh** for white noise, **bp** for band-pass noise, **vw** for variable white noise, **gg** for generalised gauss-markov, and **sw** for step-variable white noise.

For the power-law noise model you can fix the model to a specific spectral index by appending an additional parameter as follows

```
--model pl:k-1
```

The above example would fix the spectral index to -1 (flicker noise). If a spectral index is not specified then the program will also try to solve for the spectral index.

For the first-order gauss markov model you can fix the model to a specific β (roughly equivalent to the cross-over frequency) by appending an additional parameter as follows

```
--model gm:b23.0
```

The above example would fix β to 23.0. If β is not specified then the program will also try to solve for β.

For the band-pass model there are three parameters $f_{central}$ (c), width (w) and the number of poles (p). You can fix any of these parameters as follows

```
--model bp:c1w1p1
```

The band-pass model is defined slightly differently to that in *Langbein* [2004]. Instead of the parameters f_l and f_h to define the limits of the pass-band, we use f_c, the central frequency, and $width$, the width of the pass-band. The two sets of parameters are related using the following

$$\begin{aligned} f_l &= \frac{f_c}{width} \\ f_h &= f_c width \end{aligned} \tag{4}$$

or alternatively

$$\begin{aligned} f_c &= \sqrt{f_h f_l} \\ width &= \sqrt{\frac{f_l}{f_h}} \end{aligned} \tag{5}$$

The variable white noise model uses the formal errors recorded in the data file (only for the **cats** files) and solves for a scale parameter (as opposed

to a noise amplitude for white noise). The are no other parameters for this model.

The generalised-gauss-markov model is described in *Langbein* [2004]. There are two parameters for this model, equivalent to the spectral index of the power-law noise model and β for the FOGM noise model. As for above you can fix these parameters as follows

```
--model gg:b23.0k-1
```

The equation for the power spectrum in *Langbein* [2004] (equation 16) is incorrect and takes the form

$$P = P_0 \left((\frac{\beta}{S})^2 + 4\pi^2 f^2 \right)^{\frac{\kappa}{2}} \tag{6}$$

To scale these equations so that they match the power-law only power spectrum and the First-Order Gauss Markov power spectrum we have

$$P = \frac{2\sigma_2 S^{\frac{\kappa}{2}}}{f_s^{\frac{\kappa}{2}+1}} \left((\frac{\beta}{S})^2 + 4\pi^2 f^2 \right)^{\frac{\kappa}{2}} \tag{7}$$

The step-variable white noise model is an ad-hoc model devised so that we can introduce a step function in the size of the white nosie component between two specified times. You can let the minimization solve for the times but this is very unstable (as its not a continuous function) and is not recommended. This noise model should ALWAYS be used in conjunction with a white noise or variable white noise model. The model should be specified as follows

```
--model sw:b1997.0e1998.0
```

where b is the start time of when the white noise changes and e is the end time. You can specify start or end times outside the range of the time series to give a single step function in the noise (but not both).

For each model you might want to specify the sigmas by appending the additional parameter as follows.

```
--model wh:s0.0005/s0.0005/s0.0015
```

The / seperation allows you to specifiy different sigmas for each column of data. The sigma "option" is almost meaningless at the moment, since I have virtually eliminated the need for starting values. However if you want to fully specify a model and just compute the weighted least squares (–method W) then the sigmas are required and should be imput as shown.

--method <M/S/E/W> (-B) Estimation method. Choose from Maximum Likelihood Estimation (MLE), Empirical estimation, Spectral estimation or Weighted Least Squares Estimation. Additional parameter should be M,E,S or W.

--columns <number> (-C) Only use a certain column from the data file. The number after the --columns defines which columns to use. The number is basically the decimal representation of a binary sequence; 1 for use and 0 for ignore. So if there are three columns and you wish to use the middle column this would be in binary 010, which is 2 in decimal so the number would be 2. If you want to use all 3 columns then the number would be 7. This is essentially scalable, if in the future there is a file with more columns then it can cope with this. Of course if you want to use all columns then you dont need to use this option.

--scale scale_factor (-S) . Divide the data by scale_factor before estimating the model.

--sinusoid <period><time_flag><harmonics> (-A) Include a sinusoid (and n harmonics) of period <period> to the linear parameters to solve. Use time_flag to describe what units the period is in :

y years

d days

h hours

m minutes

s seconds

To solve for an annual sinusoid use - **-sinusoid 1y**. To solve for an annual sinusoid and a semi-annual use - **-sinusoid 1y1**.

--verbose (-V) verbose. Output more information.

--speed <**0/1/2/3**> **(-Z)** Speed up the computation. At every point in the minimization algorithm the noise parameters are chosen, the covariance matrix is calculated, a weighted least squares is performed on the data and the residuals calculated. The residuals, along with the noise parameters, are used to calculate the MLE value (which is the objective function in the minimization). However if you have good starting parameters then the residuals will not alter much between different noise parameter choices and so the minimization can be made faster by using the same residuals for many parts of the algorithm. In general this gives results that are a tenth of a mm or less different than the full blown algorithm for a considerable increase in speed. Note this was more of an issue in the previous version of the software, when using the simplex algorithm. This new version performs at the same speed as the previous even without this "cheat". The higher the number the less times the residuals are estimated in the algorithm.

--help (-H) Bring up a help page.

--delta <**Delta**> **(-D)** This sets the accuracy of the starting parameters. Delta is 1.1 for 10% accuracy, 1.01 for 1% ... This is used in the simplex algorithm for controlling the size of the initial walks through parameter space. **Currently defunct.**

--tolerance <**tol1**>/<**tol2**>/<**tol3**> **(-T)** Tolerance values for the simplex algorithm. **tol1** is the PRECISION required for all values of the noise parameters at solution. **tol2** is the PRECISION required for the objective function (the MLE) at solution. **tol3** : is the MAXIMUM VALUE of all the noise parameters (OR-ed with **tol2**). **Currently defunct.**

--output <**output_file**> **(-O)** All results are written to this file. If this option is not set then the output goes to **stdout**.

--filetype <**cats/psmsl**> **(-F)** Filetype, either **cats** (Default) or **psmsl**. See above.

--psdfile <**filename**> **(-P)** Create the power spectrum for this data set and output it to <filename>. If the dataset is complete (no gaps or missing data) then the psd is generated from the FFT otherwise scargle's periodogram is used.

--notrend (-E) Do no compute a trend to the data.

--cov_form <**1/2**> **(-X)** Specify the form of the covariance matrix. In the previous versions the covariance matrix for power-law noise was formed as

described in *Williams* [2003] i.e. the transformation matrix (which is used to produce the covariance matrix) is scaled by the individual ΔTs ($\Delta Tj = T_j - T_{j-1}$). For extremely unevenly spaced data, then this is the only way to do this, but it can lead to uncertain assumptions when trying to interpret the results, in particular with reference to an evenly spaced dataset. Alternatively, one can create a covariance matrix assuming a certain time-sampling and then remove the columns and rows associated with missing data. The second method is, I believe, generally more stable and corresponds to the method used by *Langbein* [2004]. It is also the only real way of computing the covariance matrix for certain noise models such as band-pass, autoregressive and moving average. The default is 1, removing columns and rows of missing data.

3.1 Example Command Line

To estimate the amount of flicker noise and white noise in the file vyas.neu assuming there is an annual signal in the series we can use the command

```
cats vyas.neu --sinusoid 1y --model pl:k-1 --verbose --output vyas.fn_mle
```

To estimate the spectral index plus the amplitudes of white and power-law noise in the east and up component of file vyas.neu assuming an annual and semi-annual signal and using the fast option we can use the command

```
cats vyas.neu --sinusoid 1y1 --model pl: --columns 3 --verbose --speed 2 --output vyas.pl_mle
```

4 Typical Output

Below is the output for the North component of the file vyas.neu using the command

```
cats --model pl:k-1 --model wh: --columns 4 --sinusoid 1y --verbose --output vyas.fn_mle vyas.neu
```

The long list of numbers are the vertex in the stages of the simplex algorithm together with the MLE value and the noise amplitudes.

```
 Sampling frequency 1.15664e-05 (Hz), 1.00 days
 Number of samples 1 period apart = 565 of 587
 Number of points in full series  = 612
 Cats Version : 3.1.1
 Cats command : cats --model pl:k-1 --columns 4 --sinusoid 1y --verbose --output vyas.fn_mle vyas.neu
 Series[0] = 1
 Series[1] = 0
```

```
 Series[2] = 0
 Number of series to process : 1
 Data from file : vyas.neu
 cats : running on bilai
 Linux release 2.4.21-27.0.1.EL (version #1 Mon Dec 20 18:47:51 EST 2004) on i686
 userid : sdwil

 Start Time : Tue Jan 25 14:21:23 2005

 work(1)  = 19992.000000
 info     = 0
 Time taken to create covariance matrix and compute eigen value and vectors : 8 seconds
 wh_only = 0.00113253 (3154.2463), cn_only = 0.00503155 (3141.9072)
 Starting a one-dimensional minimisation : initial angle 45.00
 angle = 45.000000 mle = 3176.04935824 radius =     1.476166 wh =     1.043807 cn =     1.043807
 Next choice of angle = 27.811530
 angle = 27.811530 mle = 3184.98028986 radius =     2.038823 wh =     0.951243 cn =     1.803313
 Next choice of angle = 17.188471
 angle = 17.188471 mle = 3181.49799071 radius =     2.810471 wh =     0.830539 cn =     2.684949
 Next choice of angle = 28.089645
 angle = 28.089645 mle = 3184.92628502 radius =     2.024794 wh =     0.953379 cn =     1.786298
 Next choice of angle = 25.922946
 angle = 25.922946 mle = 3185.20570106 radius =     2.140598 wh =     0.935788 cn =     1.925217
 Next choice of angle = 22.586673
 angle = 22.586673 mle = 3184.87538613 radius =     2.352437 wh =     0.903525 cn =     2.172004
 Next choice of angle = 25.139212
 angle = 25.139212 mle = 3185.21929412 radius =     2.186426 wh =     0.928835 cn =     1.979324
 Next choice of angle = 25.390604
 angle = 25.390604 mle = 3185.22046980 radius =     2.171482 wh =     0.931103 cn =     1.961729
 Next choice of angle = 25.644510
 angle = 25.644510 mle = 3185.21630040 radius =     2.156624 wh =     0.933357 cn =     1.944189
 Finished :

 Angle = 45.000000 mle = 3176.04935824 wh =     1.043807 cn =     1.043807
 Angle = 27.811530 mle = 3184.98028986 wh =     0.951243 cn =     1.803313
 Angle = 17.188471 mle = 3181.49799071 wh =     0.830539 cn =     2.684949
 Angle = 28.089645 mle = 3184.92628502 wh =     0.953379 cn =     1.786298
 Angle = 25.922946 mle = 3185.20570106 wh =     0.935788 cn =     1.925217
 Angle = 22.586673 mle = 3184.87538613 wh =     0.903525 cn =     2.172004
 Angle = 25.139212 mle = 3185.21929412 wh =     0.928835 cn =     1.979324
 Angle = 25.390604 mle = 3185.22046980 wh =     0.931103 cn =     1.961729
 Angle = 25.644510 mle = 3185.21630040 wh =     0.933357 cn =     1.944189
 Number of Parameters = 7
+NORT MLE = 3185.22046980
+NORT               INTER :    -33.2048 +-     0.5869
+NORT               SLOPE :     16.3914 +-     0.5012
+NORT             0 SIN   :     -0.2780 +-     0.2122
+NORT             0 COS   :     -0.0141 +-     0.2237
+NORT 1999.79190000 OFFST :      2.9781 +-     0.4382
+NORT               WH    :      0.9311 +-     0.0412
+NORT               PL    :      1.9617 +-     0.2468
+NORT FIXED         INDEX :    -1.000000

+COVAR
XX     0.3445    -0.2495     0.0467     0.0225     0.0694    -0.0011     0.0084
XX    -0.2495     0.2512    -0.0311    -0.0056    -0.1049     0.0006    -0.0049
XX     0.0467    -0.0311     0.0450     0.0002     0.0008     0.0000    -0.0001
```

```
XX     0.0225    -0.0056     0.0002     0.0501    -0.0339    -0.0003     0.0022
XX     0.0694    -0.1049     0.0008    -0.0339     0.1920     0.0006    -0.0044
XX    -0.0011     0.0006     0.0000    -0.0003     0.0006     0.0017    -0.0059
XX     0.0084    -0.0049    -0.0001     0.0022    -0.0044    -0.0059     0.0609
-COVAR

 End Time : Tue Jan 25 14:21:38 2005

 Total Time : 15
```

5 Processing Speeds

The time taken to process a time series depends on the length of time series and the model you are running. For example, the white noise only model is almost instantaneous whereas a stochastic model such as power-law noise plus white where the spectral index is also being estimated takes the longest. I will come up with some graphs based on a fixed data set for different machines. From my experience however, SUNS (Ultra II) are slower than SGI's. and PC's running linux or CYGWIN (under NT/XP) are around 10 times faster than the SGI 02. PC's running solaris as also quite slow compared to an equivalent machine running linux. A gain of 10 times can mean a lot when a time series can take over a day to process! I would recommend a machine with at least 512Mb (and preferably 1024Mb or more) of ram for this sort of work. I have not attempted so far to run this on a cluster and try some parallelization of the code.

References

[1] Gardner, M., Mathematical Games : White and brown music, fractal curves and one-over-f fluctuations. *Scientific American, 238,* 4, 16-32, 1978.

[2] Hosking, J.R.M., Fractional differencing. *Biometrika 68,* 1, 165-176, 1981

[3] Langbein, J., Noise in two-color electronic distance meter measurements revisited, *J. Geophys. Res., 109,* B04406, doi:10.1029/2003JB002819, 2004.

[4] Langbein, J., and H. Johnson, Correlated errors in geodetic time series: Implications for time-dependent deformation, *J. Geophys. Res., 102,* B1, 591-603, 1997.

[5] Mao A., C. G. A. Harrison, T. H. Dixon, Noise in GPS coordinate time series. *J. Geophys. Res.*, 104, 2797-2816, 1999.

[6] Williams, S. D. P., The effect of coloured noise on the uncertainties of rates estimated from geodetic time series, *J. Geodesy*, 76 (9-10), 483-494, 2003.

[7] Williams, S. D. P., Y. Bock, P. Fang, P. Jamason, R. M. Nikolaidis, L. Prawirodirdjo, M. Miller, and D. J. Johnson, Error analysis of continuous GPS position time series, *J. Geophys. Res., 109*, B03412, doi:10.1029/2003JB002741, 2004.

[8] Zhang, J., Y. Bock, H. Johnson, P. Fang, S. Williams, J. Genrich, S. Wdowinski, and J. Behr, Southern California Permanent GPS Geodetic Array : Error analysis of daily position estimates and site velocities, *J. Geophys. Res., 102*, B8, 18035-18055, 1997.