TSMP

(Time Series Manipulation)

by

Bernhard Friedrichs

May 2017
eliminated for Windows:
Gauss Krueger coordinates
sql database

Table of Contents

List of Figures

1 Introduction

First note: **tsmp** is made for being used inside UNIX scripts and uses a tremendous amount of parameters. If tsmp understands your command line options you will not be prompted again – *even if you are overwriting your own data files*.

This program is for reading and converting ats files from the ADU06 into ASCII format and back.

It also useful for calculating spectra and coherencies.

The program enables you to make permanent corrections of the header of the ats file such as correcting the start time, position, serial numbers or calibration.f

Time series can be manipulated with adding or multiplying a number or removing a trend; that way data offsets can be removed or the polarity can be changed if electrodes were wrong connected. The program also allows to add and multiply the data of two atsfiles.

NOTE: All values are stored in memory; template classes and valarrays are used for computation. Hence that you might need a incredible amount of memory for processing.

2 Installation

2.1 How to obtain tsmp

Please download from ftp.metronix.de/Geo.

2.2 Requirements

Linux, Qt. If you compile this program using Windows you should prove the length of your integer sizes, use 32 bit compilation.

In general you use qmake, make.

The *cygwin* compiler should understand the configure and makefiles.

Your compiler must support the Standard Template Library.Actually I am using gcc 4.4.1.

If you use other machines than *INTEL 32bit* please refer to section 17.The program should run on all 32bit systems and most 64bit systems.

2.3 Compilation and Installation

In order to compile and install tsmp on your system, type the following in the base directory of the tsmp distribution:

```
%export LDFLAGS=”-lfftw3”
% ./configure
% make
% make install
```

Since tsmp uses autoconf you should have not trouble compiling it. Should you run into problems please report them to the the author at B. Friedrichs

In case of missing libraries (a kdevelop bug, or forgotten LDFLAGS, edit src/Makefile and change the line LDFLAGS = -lfftw3).

Actual kdevelop project options are:

configure: -I/usr/include/mysql -I/usr/include/mysql++

LDFLAGS: -lfftw3 -L/usr/lib/mysql/ -L/usr/lib -lmysqlclient -lmysqlpp -L/usr/local/lib -lm -lgslcblas -lgsl

Using cygwin you might have to replace the keyword floorl with floor. On a freshly installed cygwin environment you might have to copy from /usr/bin to /usr/i686-pc-gygwin/bin the files ar.exe as.exe, ld.exe, nm,exe, ranlib.exe and strip.exe

Since tsmp is developed under kdevelop you should also be able to use kdevelop to compile the program.

To optimize DO NOT try export CXXFLAGS='-O2 -g0 -Wno-deprecated -msse2 -mfpmath=sse -march=i686'; it does not work! ./configure --enable-shared --enable-sse2 works fine on pentium CPUs.

2.4 Other Required Programs

For the full functioning program the FFTW 3.0.1 or 3.1.1 lib is needed. The code can be downloaded from www.fftw.org

Try to compile with --enable-shared and if you have a new pentium --enable-sse2 !!!!! You might use --disable-fortran.

One some systems you have to check the LD_LIBRARY_PATH; by default the system look in /lib or /usr/lib; FFTW installs it libraries per default in /usr/local/lib. If you don't want to change the LD_LIBRARY_PATH you might create a symbolic link as root in /lib with ln -s /usr/local/lib/fftw.so fftw.so and so on; disadvantage: if you compile a new version of the fftw you have to change this link; better to use export

LD_LIBRARY_PATH=/usr/local/lib:$LD_LIBRARY_PATH in your bashrc for example.

The cygwin complier sometimes requires a --disable-fortran.

tsmp allows you also to access a MySQL database as well as to use functions from the GSL (Gnu Scientific Library). In case that in the file

`allinclude.h` the lines

```
//#define myMySQL
//#define myGSL
```

are enabled you can make use of this functions. The tsmp software now hase to be compiled with

included header files

```
-I/usr/include/mysql -I/usr/include/mysql++
```

and additional libraries:

```
-lfftw3 -L/usr/lib/mysql/  -L/usr/lib  -lmysqlclient
-lmysqlpp  -L/usr/local/lib -lm -lgslcblas -lgsl
```

3 BUGS

First of all: tsmp make intensively usage of so called valarrays for fastest computing. These might cause problems with several compilers because they are “brand new” . Hence that you do NOT get a classical division by zero error during runtime if errors occur. So if the program does not crash – it does not tell you that everything was ok.

Duplicate frequency entries: error in interpolation routine. Please check your test_calibv.dat dump file.

If more than one ats file is read in only the information of the last computation is written. An error message is given.

4 Usage

tsmp [options] files [output file]

Hence that some options are leading to similar results: -mul -1 changes the polarity. Changing the direction of the E-field layout will (must) do the same. Please check before what the possible reason for a twisted polarity could have caused.

option	**explanation**
	always reads the complete file / use start/stop/nodata to change this if no params except filenames atre give only a logfile is written
-add 5; -add mean	*adds 5 mV to atsfiles ; -add mean automatically removes the mean of the time series*
-mul 2; -mul mean	*multiplies atsfiles with and two; -mul mead divides the time series by its mean;if you use -mul -1 you will change the polarity of the electrical / magnetical field*
-detrend	*removes a trend of the time series*
-ascii	*atsfile will be converted into ASCII; most ascii output is suppressed; if -back is switche on only the inverse FFT data is written to ascii - not the raw data*
-nspw	*ats data secion is read in but not written to disk; useful if you calculate spectra and do not need conversion into ASCII; stddev/step/spike vectors are written*
-start 4096	*skips the first 4096 samples of the atsfile (together with -stop or -use)*
-use 16384	*reads 16384 samples after given start point (together with -start)*
-stop 20000	*reads from start to stop defined samples from the atsfile (together -start)*

option	**explanation**
-wl 2048	*uses 2048 point for the FFT; activates spectra calculations automatically; -wl 1 uses all available samples*
-rect	*uses a rectangular window instead of a Hanning window (default); also must be use for inverse FFT*
-dcfft	*includes the dc part in FFT result (bad for log scales, use for inverse FFT)*
-ll nn	*lower limit of spectral output -ll 5 suppresses the first five values of the spectrum*
-ul nn	*upper limit of spectral output -ul 5 suppresses the last five values of the spectrum*
-trf auto	*enable automatically reading of the transfer function of the used sensor as specified in the headerin the log file you find Sensor Type; if type is MFS05 the program tries to read mfs05_117.txt and so on if the serial number was 117.The sensor cal file name "SENSOR.CAL" in the header is a template and is not a valid entry. otherwise the sensor cal file name will be read*
-trf theo	*uses the built in transfer functions of the spectra module*
-trf hw	*uses a built in transfer function (good for HF band where the theoretical function is poor) .. and you don't have the according calibration available*
-trf test.txt	*reads the specified file with 3 columns; 1. frequency, 2. amplitude, 3. phase in degree (not radians)*
-trf1 a.txt -trf2 b.txt	*reads both file in the two files only mode (like with atsadd for example)*
-nosystrf	*together with trf auto ONLY the system (data logger) transfer function is switched off*
-systemname adu06 -corr	*-changes system name in the header*

option	**explanation**
-back [scale_f || scale_f2] -rect [-trf xxx]	*enables backward FFT calculation; use rectangular window for correct results; scale_f indicates extra norm by f in spectra before inverse FFT, scale_f2 norm by f square*
[-dcfft] -nspw	*together -trf option the true signal in nT can be generated*
-datetime nnn -corr -wrh	*sets the UNIX time stamp into the header*
-tc	*activates the time column in the ascii output; if not activated you get a 1 column file with values*
-abst shift -tc	*absolute time in time column; correct output only if sample frequency >= 1s;* ***shift = 0 = true time****;*
	shift = -1000000 shifts the time col 1000000s backwards (to suppress high numbers)
-dateformat ISO (together with -tc and -abst 0)	*puts a date string like 2002-05-25T19:12:22.343 according to ISO 8601*
	- can be read by xmgrace tick x-axis format to date; ***to be used with -tc -abst 0***
-systemcal -corr -wrh	*adds a calibration filename to the system*
-atsadd	*adds the given atsfiles into the last given new atsfile name*
-atssub	*s.o.; ex.: -atssub file1.ats file2.ats newatsfile.ats*
-atsmul	*s.o.*
-atsdiv	*s.o.*
-spectra_add	*add all spectra to the first spectrum*
-spectra_sub	*s.o.*
-spectra_mul	*s.o.*
-spectra_div	*s.o.*

option	**explanation**
-rda sample_freq column year month day hour min sec -wrh -corr -atsoufile “name” ascii_inputfile	*prepare to read an ASCII file with given samplefrequency, column and given date. column starts with **1*** *rda 512 1 2000 12 24 9 15 0 for a file recorded at Christmas 2000 at breakfast time 9:15*
-atsoutfile	*sets the atsoutfile for -rda option is required*
-lin_conv filter_len freq's	*do a linear convolution with; see examples;* *-lin_conv 1024 16.666 50 150*
-noscale_e	*prevent E field from automatic scaling;* *spos/diplen changed to 1000 metres;* *(with autom. scaling result in time series is mV/km)*
-merge	*merge output columns; last argument is an ASCII file name; does not work together with*
	atsadd/sub/mul/div and -noascii; columns are placed in the order as filenames given.
-fil 32 or -fil 4 [-raster -1 -raster 512]	*applies 32x or 4x data reduction; -raster -1 avoids putting the start time of a (e.g) 512 raster; giving a positive value: sync start time to this raster since 1.1.1970*
-filw 32 [2,4,8] [-raster -1] [-run NN]	*same as above but new ats file is written; raster -1 avoids putting tzhe start time of a (e.g) 512 ratser; giving a positive value: sync start time to this raster since 1.1.1970; in case filtering band F supply -run NN to filter and generate a new run therewith*
-filw xx -nmtx	*instead of changing the band index form i.e. D to E the new ats filename will be ….32x.ats* *or …4x.ats*
	filtering should automatically set start times of ats files to a according start time point raster (see also -raster for overriding this)

option	**explanation**
-fn2 {1,2,-1,2}	*filename two; if -fn2 2 024C01XD.ats is given a second filename 024E01ZD.ats is automaticallygenerated; useful together with the UNIX command find*
-median range	*enable median processing of spectra (takes a lot of memory); range gives a possibleamount in % around the median to return a "mean around the median"; range 0 is the pure median*
-median_coh range	*enable median processing of coherencies (use small values like 0.1 and so on); read manual*
-destep step_len step_height relaxation skewness	*tries to find steps in your time series. Together **-dstw** overwrites your atsdata!*
-despike shoulder_length spike_height relaxation skewness	*tries to find spikes in your time series. Together **-dspw** overwrites your atsdata!; length and relaxation to be given in points, spike height in mV, skewness is dimensionless!*
-stddev min max overlay -wl nn	*test for standard deviation; values are selected if they are between min* median_stddev and max*median_stddev; stddevoverlay is the overlapping interval of the window in pts - specified by -wl*
-cat new_run_number	*cats two atsfiles together; datetimes of files must be in ascending order*
-cut_int -start n -stop m /-use m OR -overwrite	*writes a new truncated ats file with given start_sample and stop_sample as new. LSB value is leaved untuched*
-atm {new merge cat use}	*new ignores an existing file, merge reads an existing atm file and adds new results*
-chan 0 1 2	*read channels 0 1 2 from a multichannel file like .goe*
-convert	*converts given files into atsfiles*

option	**explanation**
-run	*together with convert creates a run number if not existing inside file name; use also with -overwrite option to create new files*
-mrd 9 [-corr]	*changes the reference meridian to 9 deg East*
-utm [-mrd N] [-corr]	*calculates UTM coordinates (together -corr UTM will be stored into header; with -mrd N a meridianwill be forced)*
-gk [-mrd N] [-corr]	*calculates Gauss Krueger coordinates (together -corr UTM will be stored into header; with -mrd N a meridian will be forced)*
-spwad -spwmul	*add or mulitply to spectral output files (not to data itself)*
-resample new_freq	*for sampling frequencies above 1Hz a new higher/lower sampling frequency is generated by spline interpolation; e.g. used to transform from 500 Hz to 512 Hz or 12.5 Hz to 16 Hz*
-median nn	*-returns a median spectra if nn = 0 and uses +-10% of all value AROUND the median if -median 10*
-median_coh nn	*-returns median processing for coherencies; use small values like 0.1*
-corr -wrh	*activates the header correction mode to enable permanent corrections in the atsheader; without -wrh changes will be uses but be written permanently to the file*
-lat 2345 -corr	*sets the latitude to 2345 milli seconds*
-lon 765 -corr	*sets the longitude to 765 milli seconds*
-elev 56 -corr	*sets the elevation to 56 centimeters*
-angle 90 -corr	*changes the angle to 90 degrees (e.g. y or east component)*
-diplen 100 -corr	*changes the dipole length to 100 meters*

Usage

option	**explanation**
-latlon	*-52 09 12.4 N 9 45 22.6 Please find enclosed our current price list and an overview of our terms of delivery and payment.* *-corr -wrh sets latitude and longitude*
-mrd 9	*set the meridian to 9 East (use with -corr -wrh)*
-spos x1 y1 z1 x2 y2 z2 -postodip [-corr -wrh]	*changes the sensor position together with -corr (this is preferred by MAPROS)*
-diptopos -corr	*calculates sensor positions under the assumption that dipole length and angle are correct*
-postodip -corr	*calculates dipole length and angle under the assumption that the sensor position is correct*
-gmtoff 3600 -corr	*set the GMT offset to your local time to 3600s*
-sensorcal MFS07.txt -corr	*changes the sensor cal file name in the header; use the DOS 8.3 format*
-sensor_type EFP05 -corr	*changes the sensor type to EFP06; 6 characters are allowed*
-channeltype Ex -corr	*-changes channel type to Ex Ey Hx Hy Hz; two chars max; tsmp make use of it!*
-adbser or -aduser -corr	*sets the ADB or ADU serial number (together with -corr); use also with -overwrite option to create new files (without -corr)*
-sensser 1234 -corr	*sets the Sensor serial number (together with -corr)*
-resis 1000 -corr -wrh	*changes the resistivity measured by the ADU*
-dcoffset 1.2 -corr	*changes the dc offset (to 1.2 mV)*
-channeltype Hx -corr	*set the channel type to magnetic North*
-channelno 2 -corr	*set the internal channel no to 2 ... that should be an Ey channel (Ex, ...Hz and so on)*
-samplefreq 64 -corr	*set the sample frequency inside the header*
-wtrend	*removes a trend for each fft window (use together with -rect or not)*

option	**explanation**
-sync	*find the same start time of all give ats files*
-sync2	*find the same start time of all give ats files - and stop times; needed for analysis!!*
-synctime 1002384003	*forces all file to start at given datetime, UNIX time format required!!*
-raster 512	*sets the beginning interpr. /filtering to a 512s start raster*
-ccf cstartx cstarty cshifts	*calculates cross- and autocorrelation of time series (see Tips and Tricks section)*
-overwrite	*enables overwiting of ats data source file - see Tips and Tricks*
-run	*proviedes a new run number; eg. for new output files; if -run is the only option you obtain a simply copy of the source file; howver due to conversion from int->double->int there is loss of precision around 0.01%*
-raster nn	*filter to a given time raster of nn seconds (see Tips&Tricks)*
-calib 0.125	*generates a list of calibration frequencies starting with 8s*
-add [mean]	*-adds a given mV to the time series; add mean: remove the mean from time series*
-mul [mean]	*multiplies time series with given mV; mul mean divides time series by mean*
-time_s nn	*shifts start time with nn seconds; nn 10 at a sampling frequency of 2Hz results in a shift of 20 samples (same effect like -start 20);* *with 16s samples and -tc first time value is 16 followed by value – like above* *with* ***-abst 0*** ***-corr*** *header will be corrected (but not written) and ascii output will contain shifted time; time series will be parallel now to a 16s offset station*
-time_s nn -corr	*corrects start time in header with nn seconds*

option	**explanation**
-gensign	*activate generate a signal and write ats file*
-gen_sin f f f ... f	*generate sine frequencies*
-gen_rand scale	*generate a simple random signal and scale (multiply)* *together with -gensign*
-gen_gsl_rand mix0 mix1 ... mix5	*generate special random signal for a 5 channel site – see explanation in text!!!*
-nodata	*data section of files is not read*
-nspw	*suppress spectral output files; needed for inverse FFT*
-wtrend	*removes trend over given window (-wl, -rect)*
-detrend	*removes trend over complete time series*
-out_dir dir	*redirects output file; useful if data is on a CDROM*
-sst nn	*sub stacks in time domain; to be used with -wl*
-wbin [float] -merge	*writes data binary to disk ; float indicates 16bit data; otherwise 32bit*
-pipe	*pipes numbers to the screen*
-corr	*make correction in header data - but no not write*
-wrh	***write header; together with -corr changes in ats file will be permanent***
-firfil 641 3 20 0	*filters a band pass with 3 and 20 Hz*
-firfil coeff freq freq reduct	*example for 4096Hz and 8193 coeff: 0 1000 is low pass, 100 4069 is high pass, 100 1000 is bandpass, 70 40 is notch / bandstop; reduct= 0; ref. to example section*
-prz 0.2	*gives a parzen radius of 0.2; should be 0.05 ... 0.1, 0.2 ... 0.5*

option	**explanation**
-write_trf	*write all interpolated transfer functions to disk: should not be used with -wl 1*
-ovr	*overlapping in points, must be smaller than -wl*
-mt	*magnetotelluric eqns.; not ready yet*
-coh	*calculate coherencies of two time series (-sync2 should be used in case)*
-extra_scale [f, sqr(f), sqrt(f) , 1/f, 1/sqr(f), 1/sqrt(f)]	*for each ats file give:-extra_scale f sqrt(f) if you have two files; first spectra divided by f second by sqrt(f) and so on*
-iso_s nn.dd	*shift ISO date by 0.12 or -0.34 seconds; this ONLY affects the output – nothing else*
-xmul	*supply 10 0 if you want to multiply the raw spectra by 10*
-xadd	*supply 0 10 to change the imaginary part of the raw spectra only*
-isostart ISO 8601 time	*-isostart 2009-08-11T20:00:00*
-isostop ISO 8601 time	*-isostop 2009-08-13T02:00:00*
-calib xx	*generate calibration data, phase multinplied with xx*
-flist file.lst	*read data file with calibration frequencies*

Older version of GMS.exe do not use z-coordinates for dipole length calculation.

If you want to use tsmp inside scripts or in batch mode don't provide an output filename ending with .ats. You might be prompted for overwriting or not (example: tsmp all merge 1.ats 2.ats my_new_data.ats; here you will be asked whether my_new_data.ats shall be overwritten).

4.1 Examples

Command line	**action**
-ascii *.ats	*converts all ats files in that directory to *.dat one column ASCII files containing the data if you files of **different length** (e.g. different bands) use-all *C.ats and *D.ats and so on*
-ascii -tc *.ats	*same as above but column one contains the time, column two the data*
-start 3489 -use 16384 *ats	*reads 16384 samples after skipping the first 3489; converts to ASCII*
-start 0 -use 16384 -median 0 -wl 1024 a.ats b.ats	*calculates 16 spectral windows of the given files;output will be a.dat, a_ampl.dat (stacked spectra), a_ampl_max.dat (all maximum values of every spectrum),a_ampl_min.dat (all minimum values of every spectrum),a_amp.med.dat (median spectrum); same applies to b;and: a_b_coh.dat (spectral coherency between a and b)*
-start 3489 -use 16384 -wl 1024-nspw a.ats b.ats	*s.o., except that time series (a.dat, b.dat) are not written*
-corr -time_s 13 028*.ats	*changes the start time im the atsheader +13 seconds of all ADU 28 time series in that directory.*
-lin_conv 1024 16.666 50 150	*linear convolution with 1024 points length and target frequencies 16 2/3 Hz and 50, 150 Hz; hence that the convolution works **ONLY** if the frequencies are absolutely stable! Minor changes can lead to heavy distortions.*

Command line	**action**
-merge a.ats b.ats c.ats all.dat	*merges all data in to separate columns in all.dat;*
-noascii *.ats	*get log file of all ats files*
to analyze a directory structure use:	*find ./ -type f -name ``*.ats'' -exec tsmp -noascii {} \;*
-out_dir /wrk/spc/	*output files will be written to /wrk/spc*
-out_dir cal	*puts the output into the directory cal below your current working directory*
-corr -spos 10 n n n n n	*changes only the the x1 entry in the header*
-stddev 0.3 2 64 -wl 128	*analyse 128pts windows with 64pts overlapping; values are assumed correct ifthey are between 0.3 and 2* of the median of the overall standard deviation*

Together with the find command you can change a complete directory structure, for example if the sensor was not set correctly with gmsxx.exe:

```
find ./ -type f [-noleaf] -name “*Z*.ats” -exec tsmp -corr
 -wrh -senser 6 -sensortype mfs06 {} \;
```

Median: median calculation is a powerful tool to eliminate outliers. Hence that spectra are logarithmic, median 50 will add ±50% to the median – that is almost nothing. Try 1000 for example. I am sure that you know that the spectra and coherencies do NOT follow a Gauss distribution. The obtained values are only useful for your statistics – do not use them for interpretation.

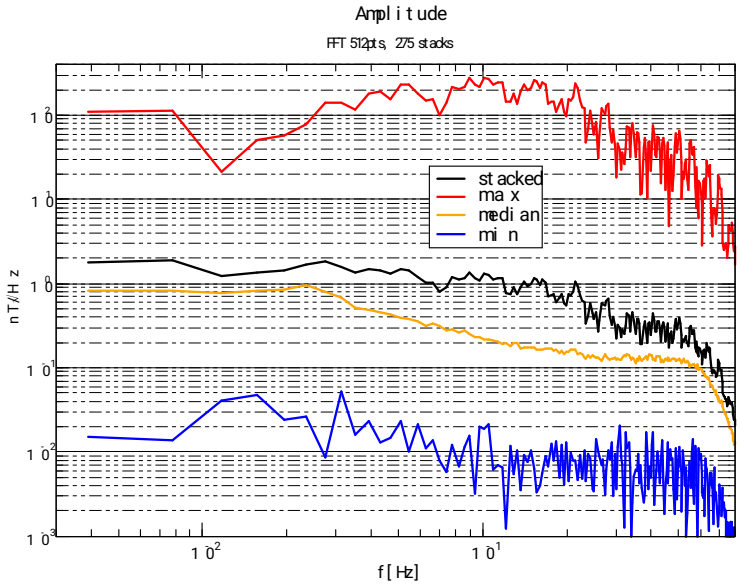


Figure 4.1: Ranges of measured spectra

If have tried to implement these features also for the Windows user. It should work if your Windows compiler identify with __CYGWIN__ or __MINGW32__ or __WIN32__ environment.

Do not use tsmp *.ats if you want manipulate the file (header)s.

tsmp will use the information from the first file! So try tsmp -xxx *C.ats for example or – much better! - the `find` command as shown above!

5 Results from Noise Tests

The scaling in MAPROS and the FFT producing smoother results than tsmp. However, if you stack a recent amount and smooth the spectra afterwards (e.g. Running average or parzen) you should be able to estimate the exact amplitude of the incoming signal.

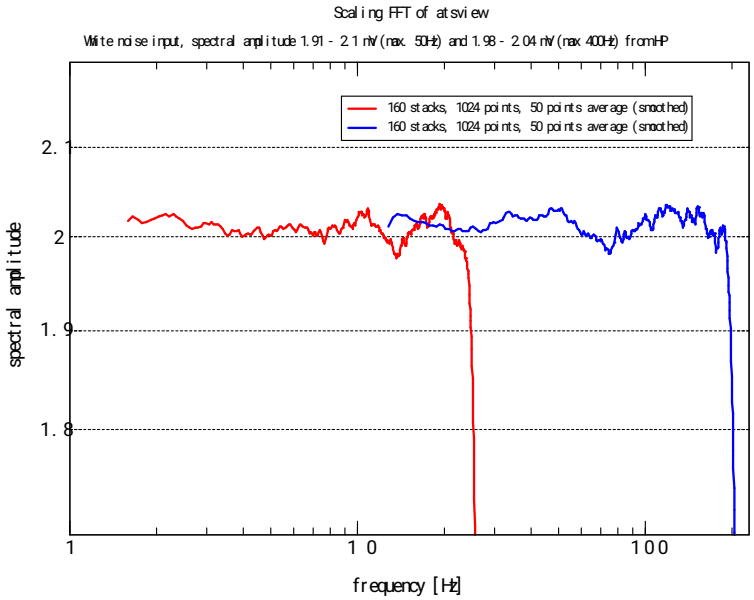


Figure 5.1: 2mV noise from HP spectrum analyser

In the figure above white noise was generated with a HP spectrum analyser. The noise level was estimated by the HP. The length of the analyzed time series was 168340 points with a FFT length of 1024 points and 160 stacks. The results were smoothed with the running avergage tool from xmgrace.

Hence that the resolution of the spectra is a function of your window length:

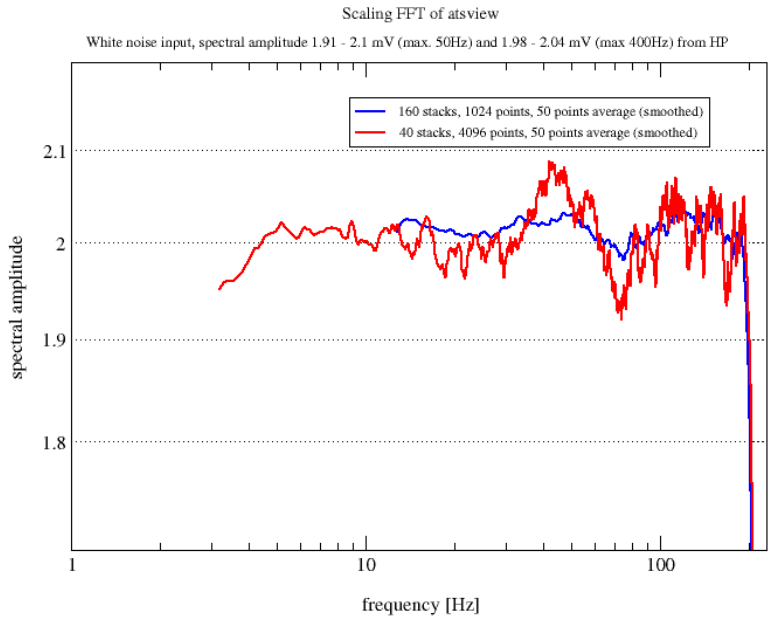
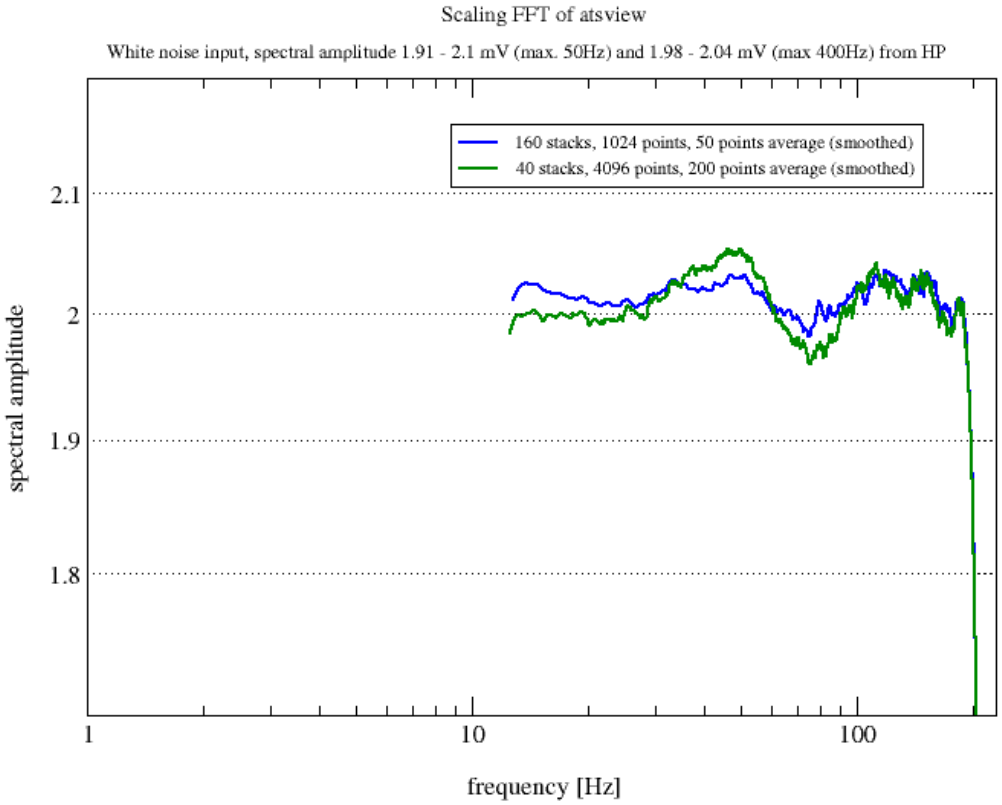


Figure 5.2: 2mV and 2 different window lengths

Here two different length were used – and also a different amount of stacks of course. Even though both spectra are smoothed is obvious that the spectrum calculated with 4096 window length is rougher (because you have less stacks) and has a higher resolution in frequency.



This graph demonstrates that if you change the window length of your FFT and did not change the parzen radius (or here simulated with a averaging on the same frequency interval) you obtain very similar results.

6 Used Equations

6.1 FFT

The FFT (Fast Fourier Transform) of tsmp is in fact a captured FFT routine from FFTW.

The Fourier Transform (FT) should equal to

$$F(i\omega)=\int_{\infty}^{\infty} f(t)e^{i\omega t}dt \tag{6.1.1}$$

and

$$f(t)=\frac{1}{2\pi}\int_{-\infty}^{\infty} F(i\omega)e^{i\omega t}d\omega \tag{6.1.2}$$

Using a discrete FFT (Fast Fourier Transform) these equations slightly change to:

$$F_{FFT}(f_k)\rightarrow\sum_{n=0}^{N-1} f[n]e^{i2\pi\frac{kn}{N}} \tag{6.1.3}$$

$$k = 0,1,2, \ldots N\text{-}1$$

Where f_s = *sampling frequency* and t_d = *discretization intervall* with $t_d = 1/fs$; N is the number of samples and k one of the discrete spectral lines.

$$\omega_k=\frac{2\pi f_s}{N}k \quad with\, k=0,1,\ldots N \tag{6.1.4}$$

or

$$f_k=\frac{f_s}{N}k \quad with\, k=0,1,\ldots, N/2 \;\; for\, the\, FFTW \tag{6.1.5}$$

$$\Delta\omega = \frac{2\pi f_s}{N}, \Delta f = \frac{f_s}{N} \text{ as spectral resolution} \tag{6.1.6}$$

The FFT is a periodical function with the period $\frac{2\pi}{t_d}$ resulting into the sampling theorem that the sampling frequency must be double the highest frequency in the spectrum:

$$f_{max} < \frac{f_s}{2} \vee \omega_{max} < \pi f_s = \frac{\pi}{t_d} \tag{6.1.7}$$

and therefore every ADC has to use a low pass filter before conversion.

Also we obtain *N*/2 independent spectral lines up to the Nyquist frequency plus a DC part of the signal; for 1024 input values we obtain 513 values from the FFT: 1 DC part plus 512 frequencies. Some FFT routines like the FFTW do NOT store values above *N*/2 (so called half sided FFT). However: the FFTW requires a scale of *N* for the inverse – but a scale of *N*/2 for the power spectra as we will see in (6.1.10).

As we can see the summarization does not change the units and we have to divide by the sampling frequency

$$F(i\omega) \Leftrightarrow F_{FFT}[\omega_k] / f_s \tag{6.1.8}$$

to obtain the same result as with the FT. A signal recorded in [V] has the unit [Vs]=[V/Hz]after the transformation because of the integration over the time *dt* or division by f_s.

Finally to make the result independent from *N* the result should be divided by *N*/2 (or N for a full sided FFT):

Used Equations

$$F_{Norm}(f)=\frac{2}{N f_s}\left[\frac{F_{FFT}[f_0]}{2}, F_{FFT}[f_k], \frac{F_{FFT}[f_{N/2}]}{2}\right] \tag{6.1.9}$$

For the half-sided FFT the first (DC) and the last (Nyquist) value have to be divided by *N*. The reason for that is, the all samples have a neighbouring sample to the left and to the right, except the first and the last which have only a sample to their right or left; or in other words: the bandwidth of these two sample is half the size in comparison to the others.

The following equation (using the un-normalized FFT result)

$$F_{Norm}(f)=\frac{2}{N f_s}\left[\frac{F_{FFT}[f_0]}{2}, F_{FFT}[f_k], \frac{F_{FFT}[f_{N/2}]}{2}\right] \tag{6.1.10}$$

is expressing that the energy is conserved (Parseval's theorem) or in other words the energy (power) can be calculated in time domain as well as in the frequency domain. Since the power of a signal is defined as

$$power=\int_{-\infty}^{\infty}|f(t)|^2 dt=\frac{1}{2\pi}\int_{-\infty}^{\infty}|F(\omega)|^2 d\omega \tag{6.1.11}$$

or

$$power=\int_{0}^{T}|f(t)|^2 dt=\frac{1}{2\pi}\int_{0}^{f_{max}}|F(f)|^2 dt \tag{6.1.12}$$

we have to multiply Parseval's equation above ([Parseval])with dt (that is t_d) to obatin he power spectra (except for DC and Nyquist):

$$power(f_k)_{xx}=\frac{2}{N}\frac{|F_{FFT}(f_k)|^2}{f_s} \quad with \quad the \quad units\left[\frac{V^2}{Hz}\right] \tag{6.1.13}$$

from here it follows that from a “naked” FFT routine we have to scale the result with

$$\frac{F_{FFT}(f_k)}{\sqrt{N/f_s}} \; with \;\; the \;\; units\left[\frac{V}{\sqrt{H}}\right] \qquad (6.1.14)$$

Once again: use equation (6.1.10) to find out whether your FFT needs a scaling with N or N/2 (it will be N/2 in most cases even though the authors write something different).

The conclusion is that the FFT can be used in MT especially because our time series are limited by the recording itself (or a window function). But then the spectra have to be recalibrated to fit the theory of our unlimeted MT signal.

Let t be your time series vector, which is an array of double, an wl you window length. T is sliced into L windows, each window is detrended and (if not rectangular window function is chosen) a Hanning window is applied. Then the FFTW is used to obtain an array of complex numbers spc. From wl real input values we get *pts = wl/2* complex numbers in spc. spc is calibrated with $spc = \frac{spc * 1}{\sqrt{f_{sample} * pts}}$. If a transfer function trf is given we obtain $spc = \frac{spc}{trf}$.

Finally if you use a windowing function the result has to be multiplied with the inverse of the integral of that window; if a rectangle window is used nothing is to be done; if a standard Hanning window is used the result has to be multiplied with 2 (Hamming 1/0.54, Blackmann 1/0.42). If your window function has an integral of 1, no correction is necessary.

tsmp uses the function

$$hw(n) = 2\left(1 - \cos^2\left(\frac{\pi n}{N}\right)\right) n = 0, \ldots window\ length\ N \qquad (6.1.15)$$

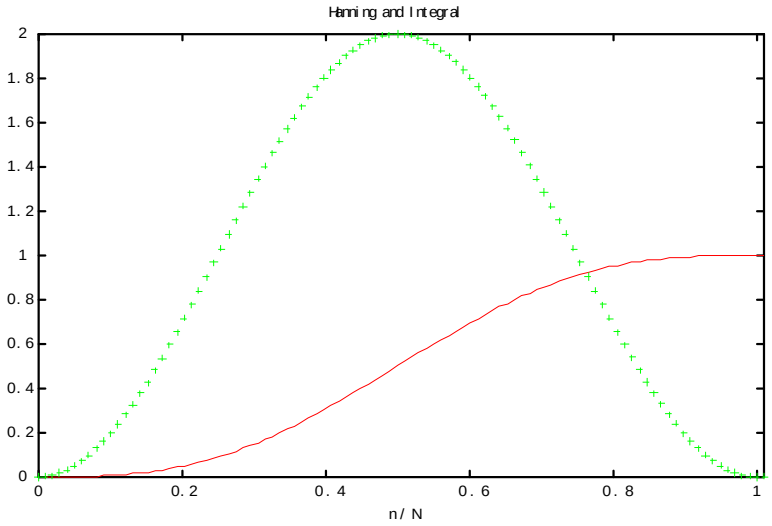
with its integral

Used Equations

$$\frac{1-\cos(\pi n)^2 \quad \left(1/\sin(\pi n)^2\right) \quad (2\pi n \sin(2\pi n))}{4\pi} \qquad (6.1.16)$$

$$n = 0\ldots 1 \quad \textit{scaled onto window length}$$

and the **raw spectra are scaled with** $\sqrt{2/\left(f_{sample} \cdot window_{length}\right)}$. .



weighting of the Hanning window (dots) and the integral of that function

The following figure illustrates the properties of the used FFT quite well:

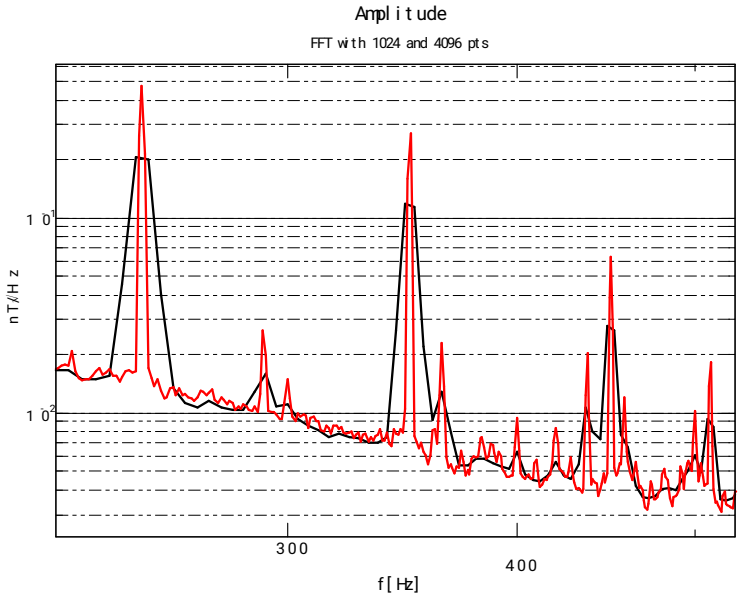


Figure 6.1: properties of the FFT

the “so called” noise is independent from the window length of the used FFT; the sine waves from the power line frequencies however are changing to half width and double height and will become a delta peak for an unmlimited FFT window.

6.2 Choosing your Window Type

tsmp offers you the rectangular window and the Hanning window, the general properties are listed below:

Used Equations

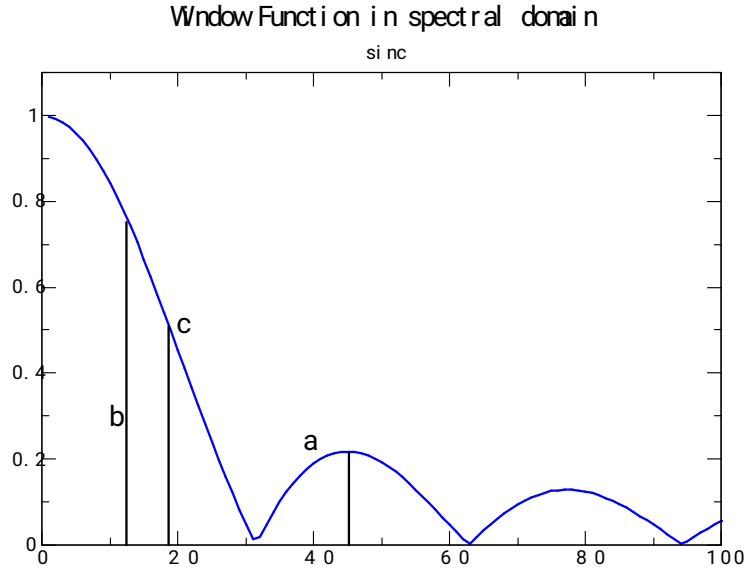


Figure 6.2: Discretization Error

Three different parameters are of interest:

a) a) ratio of main maximum to neighboring maximum

$$a = \frac{neigh}{main}$$

b) b) discretization error

$$b = \frac{amplitude(f_{sample}/2)}{amplitude(f=0)}$$

c) c) width

$$3\text{dB} = \frac{amplitude(f=0)}{amplitude(f_{width})}$$

window	**a**	**b**	**c**
rectangular	-13 dB	0.64	0.45 Δf
triangle	-27 dB	0.81	0.64 Δf
Hanning	-32 dB	0.85	0.72 Δf

In other words: the rectangular window gives a good resolution with small errors (b,c). But the neighboring maxima are quite high (a). The Hanning window is mostly used, (b,c) are acceptable and neighboring maxima well suppressed.

If your signal has a strong decay like $e^{-\omega t}$ only the rectangular window can be used.

6.3 Example

Assume a sample rate of 512 Hz and an input signal in mV.

If you an run a FFT with a window length of 1024 points the result are 512 complex numbers.

Your resolution (bandwidth) in the frequency domain is
sample rate/window length = 512 Hz/1024 = 0.5 Hz

Your first complex number is related to the DC part of the signal, the last represents $512 \cdot 0.5\ Hz = 256\ Hz$ the highest possible frequency in your spectrum, the Nyquist frequency. The units of the x-axis are Hz and on the y-axis are $mV/\sqrt{Hz}$.

6.4 Coherency

Consider the spectra X and Y and their conjugates $\bar{X}$ and $\bar{Y}$ and $\langle | \rangle$ for their stacked values.

The coherency is defined then

$$coh^2 = \frac{\langle X\,\bar{Y}\rangle\langle \bar{X}\,Y\rangle}{\langle X\,\bar{X}\rangle\langle Y\,\bar{Y}\rangle} \qquad (6.4.1)$$

If you use the Terms of cospectrum and quadrature:

$$\begin{aligned} X\,\bar{Y} &= C_{xy} + iQ_{xy}\,;\, \bar{X}\,Y = C_{xy} - iQ_{xy} \\ C_{xy} &= \Re(X)\Re(Y) + \Im(X)\Im(Y) \\ Q_{xy} &= \Im(X)\Re(Y) - \Re(X)\Im(Y) \end{aligned} \tag{6.4.2}$$

$$coh^2 = \frac{\langle C_{xy}^2 \rangle + \langle Q_{xy}^2 \rangle}{\langle X^2 \rangle \langle Y^2 \rangle} \tag{6.4.3}$$

tsmp writes *coh* to disk if spectral calculation is enabled and only two input files are given. You will need more than one stack to get a result different from one (more degrees of freedom) because the coherency is formulated as quotient between predicted and measured value.

6.5 Cross- and Autocorrelation

The cross- and autocorrelations are defined as:

$$ccf\,(shift) = \sum_{i=0}^{wl-1} x(i)\,y(i+shift) \tag{6.5.1}$$

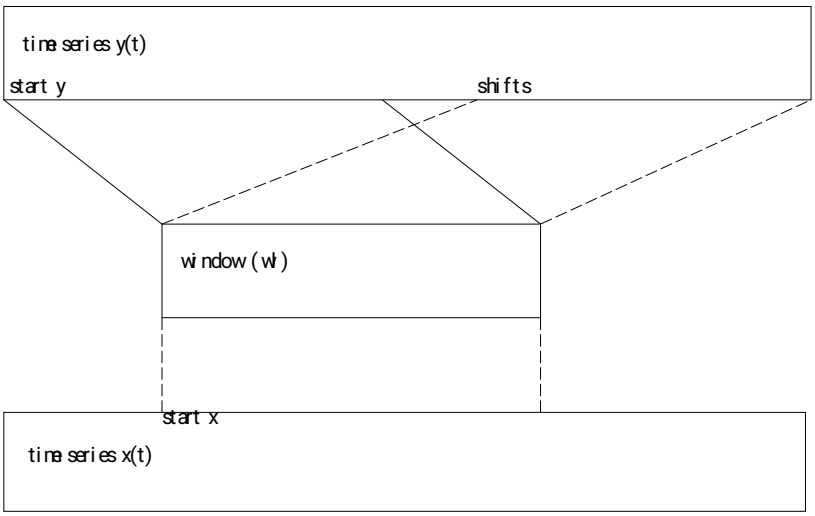


Figure 6.3: Cross Correlation

6.6 Standard Deviation

Mean is defined as

$$\bar{x} = \frac{1}{N} \sum_{i=0}^{N-1} x_i \tag{6.6.1}$$

standard deviation as

$$\sigma = \sqrt{\frac{1}{N-1} (\bar{x} - x_i)^2} = \sqrt{\frac{1}{N-1} \left(\sum_{i=0}^{N-1} x_i^2 - \left(\sum_{i=0}^{N-1} x_i \right)^2 / N \right)} \tag{6.6.2}$$

(where the second term is faster and with less round off noise).

6.7 Detrend

The detrended time series is calculated as follows:

Used Equations

$$x_{dt}[i] = x[i] - \bar{x} - \left(\frac{2\mathrm{i}}{N-1} - 1\right) \cdot \frac{\sum_{j=0}^{N-1}\left(\frac{2\mathrm{j}}{N-1} - 1\right)x[j]}{\sum_{j=0}^{N-1}\left(\frac{2\mathrm{j}}{N-1} - 1\right)^2} \qquad (6.7.1)$$

7 Apply Transfer Functions

After using the FFT you get $mV/\sqrt{Hz}$ as your **spectral amplitude**.

To say something about the earth`s magnetic field one has to apply the transfer functions of the sensors to the signal obtained in *mV*.

The transfer function of the metronix coils is measured in $\frac{V}{nT\ Hz}$ with a Solatron. tsmp scales this to the *mV* range as stored in the ats files.

If you generate your own transfer function please use the metronix format and scaling.

Dividing your FFT result with the transfer function of the magnetic field sensors leads to

$$\frac{mV}{\sqrt{Hz}}\frac{nT\ Hz}{mV}\frac{1}{Hz}=\frac{nT}{\sqrt{Hz}} \tag{7.1}$$

as units or your **calibrated amplitude spectra** for your y-axis.

7.1 Noise

Noise level (as well as coherency) will be calculated automatically if only two files are given (this make sense for parallel tests).

I use *noise = (1 - coh) * amplitude spectra*. Hence that I do not use coh^2.

7.2 Cal Files

Cal files can have the metronix format:

Apply Transfer Functions

```
Zylinderspule, MFS 05 No. 001


FREQUENCY    MAGNITUDE      PHASE
(Hz)      (V/(nT*Hz)     (deg)


Chopper on
1.0000E-01  1.9975E-01  8.8324E+01
1.0000E+00  1.8979E-01  7.5202E+01
1.1220E+00  1.8934E-01  7.3797E+01

Chopper off
1.0000E+00  2.7926E-02  -1.5219E+02
1.1220E+00  3.2516E-02  -1.5549E+02
```

Hence that for some compilers `+1.0000E-01  +1.9975E-01 +8.8324E+01` might not a valid format. Remove leading "+" signs if your dump file test_calibv.dat shows errors.

Hint: Hence that MAPROS needs <cr> <lf> (\r\n) at the end of a line. Unix might change the <endl> into a simple <lf> (\n).

Or provide a simple three column ascii file with frequency *Hz*, magnitude $\frac{V}{nT\ Hz}$ and phase in *deg*. You have to make sure then that you use the correct calibration function with chopper on or off.

If -trf auto is enabled tsmp searches for a file name given in the ats header; if the frequency range is greater than given in the calfile tsmp uses the built in transfer function outside the calfile range. If the sensor is unknown **Akima spline** is used inside and outside the frequency range of the calfile.

7.3 Built in Transfer Function

The routine which is calculating the spectra can also generate transfer functions for MFS06 and MFS05 – which might be used for **frequencies < 512 Hz only** (because chopper on assumed):

$$F(f)_{sensor} = 0.8 \frac{V}{nT} \quad \frac{P_1}{1+P_1} \quad \frac{1}{1+P_2}$$
$$with\ P_1 = i\frac{f}{4\,Hz};\ P_2 = i\frac{f}{8192\,Hz};\ f = Frequency \tag{7.3.1}$$

One can also estimate the sensor transfer function for low frequencies with

$$Phase = 90° - atan\left(\frac{f}{4\,Hz}\right) \tag{7.3.2}$$

$$Amplitude = 0.2 \cdot \frac{1}{\sqrt{1+\left(\frac{f}{4\,Hz}\right)^2}} \tag{7.3.3}$$

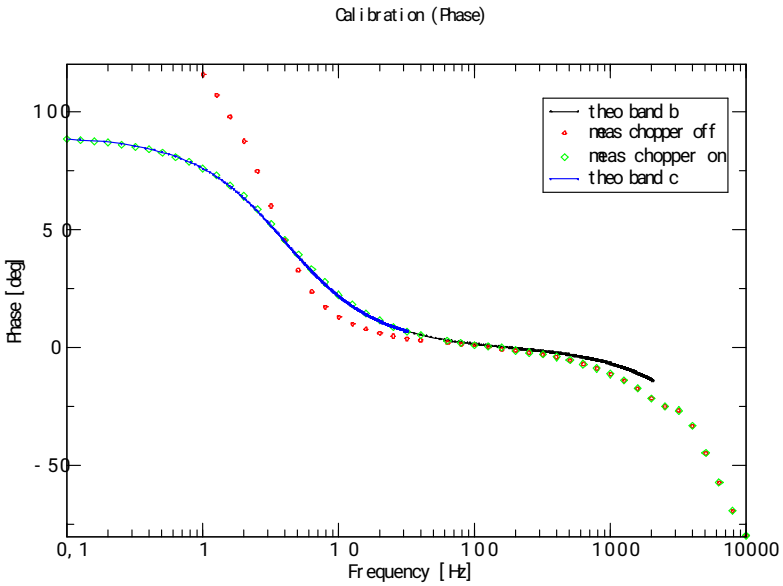


Figure 7.1: Built in and Measured Transfer Function (Phase)

7.4 ADU-06 Transfer Functions

Also the ADU-06 has transfer functions; they need not to be used for MT but for displaying the correct spectra for single channels:

$$F = \frac{1}{1 + \frac{f}{33800} i} \quad \text{if gain is on} \qquad (7.4.1)$$

(Radio filter at 33kHz)

$$F = \frac{1}{1 + \frac{f}{846} i} \quad \text{f\_sample = 40kHz} \qquad (7.4.2)$$

(846 Hz high pass for 40kHz sampling frequency)

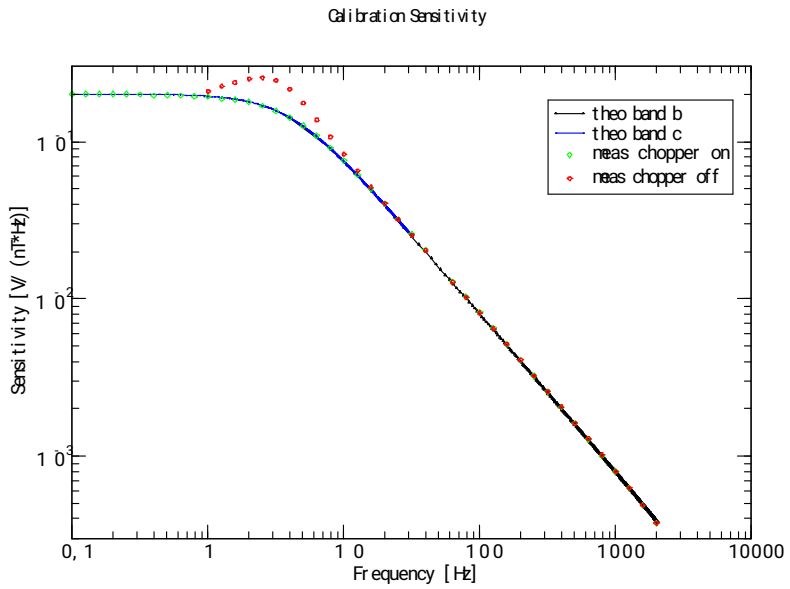


Figure 7.2: Built in and Measured Transfer Function (Sensitivity)

Looking at the calibration function we see for example for frequencies higher than 10Hz and 10V as maximum output of the coil:

*10 V / (0.08 * 10 Hz) = 12.5 nT or 10V / (0.008 * 100 Hz) = 12.5 nT* and so on is the maximum signal strength.

The maximum range *nT* is almost constant here.

For lower frequencies the maximum range varies with the frequency:
*10V / (0.2 * 0.1 Hz) = 500 nT and 10V / (0.2 * 0.01 Hz) = 5000 nT*
and so on.

7.5 Hardwired Transfer Function

As one can see above – the theoretical transfer function and the measured transfer function with chopper on are fitting perfectly.

However at sampling frequencies higher than512 Hz (256 Hz interpretation) one must apply a measured transfer function to obtain correct results. Use your transfer functions or if not available the -hwxxx options.

8 ADU-07

8.1 LF Board

8.1.1 Sample Frequencies

128	265	512	1024	2048	4096

8.1.2 Gains

Gains can be combined separately

pre	1	2	4	8	16	32	64
post	1	2	4	8	16	32	64

8.1.3 Filter

Filter Name	corner frequency		

8.2 HF Board

Sample Frequencies

8192	16384	32768	65536	121072	262144	524288

8.2.1 Gains

pre	1	8	
post	1	8	64

8.2.2 Filter

9 Units

$$[B] = \frac{N}{As\, m/s} = \frac{N}{Am} = \frac{Vs}{m^2} = T$$

$$V = \frac{Nm}{As}$$

tsmp works with *mV* and *nT* in the time series and *Hz* for the frequencies. The calibration file format for Metronix coils if *f[Hz], Amplitude* $\frac{V}{nT\ Hz}$ *Phase[deg]*.

10 File Formats

10.1 ATS Header

(LP32 memory model, please refer to [sec: system]); FLOAT also known as SINGLE or REAL. Red: New in version 0.73; green: used since 0.73.

ADR	Byte	Type	Name	Info
000H	2	INTEGER	HeaderLength	Length of Header in Byte
002H	2	INTEGER	HeaderVer	Version Number of Header (*100)
004H	4	LONGINT	Samples	Number of Samples
008H	4	FLOAT	SampleFreq	Sample Frequency
00CH	4	LONGINT	DateTime	Start Time of Measurement
				(Sec. since 1.1.70)
010H	8	DOUBLE	LSBVal	LSB unit in (gain incl.)
018H	4	LONGINT	GMTOffset	Offset of local time to GMT in Sec.
01CH	4	BYTE	reserved	Original Sample Freq
020H	2	INTEGER		serial number of ADU06
022H	2	INTEGER		serial number of ADB06
				(ADC board)
024H	1	BYTE		channel number (0..7)
025H	1	BYTE	SensorChopper	1 = on, 0 = off
026H	2	CHAR		channel type (Ex, Ey, Hx, Hy, Hz)
028H	6	CHAR		Sensor type (MFS05, BF4, ...)
02EH	2	INTEGER		serial number of sensor
030H	4	FLOAT		x1 coordinates of 1. Dipole [m]

ADR	Byte	Type	Name	Info
034H	4	FLOAT		y1
038H	4	FLOAT		z1
03CH	4	FLOAT		x2 coordinates of 1. Dipole [m]
040H	4	FLOAT		y2
044H	4	FLOAT		z2
048H	4	FLOAT		e-field dipole length [m]
04CH	4	FLOAT		angle (0 = north) [degrees]
050H	4	FLOAT		probe resistivity [Ohm]
054H	4	FLOAT		DC offset voltage [mV]
058H	4	FLOAT		Internal gain amplification (1 or 30; ADU-07: pre-gain)
05CH	4	FLOAT		ADU-07: post-gain
060H	4	LONG		Latitude [msec]
064H	4	LONG		Longitude [msec]
068H	4	LONG		Elevation [cm]
06CH	1	CHAR		Lat / Long Type:
				'U' user defined
				'G' internal GPS clock
06DH	1	CHAR		type of additional coordinates:
				'U' UTM
				'G' Gaus-Krueger (Germany)
06EH	2	INTEGER		reference meridian
070H	8	DOUBLE		x coordinate
078H	8	DOUBLE		y coordinate
080H	1	CHAR		GPS / CLK status:
				'G' GPS locked
				'C' CLK synchronized

ADR	Byte	Type	Name	Info
				'N' CLK not synchronized
081H	1	BYTE		Approximate accuracy of GPS / CLK :
				9 means accuracy of 10^{-9}
082H	2	INTEGER		Offset UTC and GPS in Seconds (2006: 14 seconds)
084H	12	BYTE	SystemType	ADU06, EMIMT24, etc.
090H	12	CHAR		survey header file name
09CH	4	CHAR		type of measurement: MT ', 'CSAMT'
0A0H	12	CHAR		log file of system self test
0ACH	2	CHAR		result of self test 'OK' or 'NO'
0AEH	1	BYTE	SystemChopper	1 = chopper on, 0 = chopper off; (unused for ADU)
0AFH	1	BYTE	reserved	
0B0H	2	INTEGER		number of calibration frequencies in file
0B2H	2	INTEGER		length of frequency entry (32 byte)
0B4H	2	INTEGER		version of calibration format (*100)
0B6H	2	INTEGER		start address of calibration information
				in header (400H)
0B8H	8	BYTE		LF Board Filters
0C0H	12	CHAR		file name of ADU06 cal file
0CCH	4	LONG		date/time of calibration
0D0H	12	CHAR		file name of sensor calibration
0DCH	4	LONG		date/time of calibration

ADR	Byte	Type	Name	Info
0E0H	4	FLOAT		Powerline Freq.1
0E4H	4	FLOAT		Powerline Freq. 2
0E8H	8	BYTE		ADU-07 HF Board Filters
0F0H	4	FLOAT		CSAMT transmitter frequency
0F4H	2	INTEGER		CSAMT time series blocks
0F6H	2	INTEGER		CSAMT stacks / block
0F8H	4	LONG		CSAMT block length
0FCH	4	BYTE		BoardType ADU07: LF, HF
100H	16	CHAR		Client
110H	16	CHAR		Contractor
120H	16	CHAR		Area
130H	16	CHAR		Survey ID
140H	16	CHAR		Operator
150H	112	CHAR		reserved
150H	8	CHAR		Coordinates Type (UTM)
158H	4	LONGINT		Zone Number
15CH	1	CHAR		Letter Designator
160H	8	DOUBLE		Northing
168H	8	DOUBLE		Easting
170H	4	LONGINT		UTM Reference Meridian
174H	12	CHAR		Ref. Ellipsoid
1C0H	64	CHAR		Weather
200H	512	CHAR		Comments

ADR	Byte	Type	Name	Info
400H	32			calibration frequency 1
400H	4	FLOAT		frequency
404H	4	FLOAT		amplitude
				e-field [V]
				h-field [nT/V]
408H	4	FLOAT		phase [°]
40CH	4	FLOAT		accuracy of amplitude [%]
410H	4	FLOAT		accuracy of phase [+/- °]
414H	12	BYTE		reserved
420H	32			calibration frequency 2
...				...
400H +				calibration frequency n
(n-1) * 20H	32			
400H +				
n * 20H	4 *			
	SAMPLES	LONGINT (int32)	DATA	time series data

Detailed Filter Settings, bits for LF:

Byte	Addr.	7	6	5	4	3	2	1	0
0	0x0B8				ADU07_LF_LP_4HZ	ADU07_LF_RF_4	ADU07_LF_RF_3	ADU07_LF_RF_2	ADU07_LF_RF_1
1	0x0B9								
2	0x0BA								
3	0x0BB								
4	0x0BC								
5	0x0BD								
6	0x0BE								
7	0x0BF								

Detailed Filter Settings, bits for HF:

Byte	Addr.	7	6	5	4	3	2	1	0
0	0x0E8								ADU07_HF_HP_1HZ
1	0x0E9								
2	0x0EA								
3	0x0EB								
4	0x0EC								
5	0x0ED								
6	0x0EE								
7	0x0EF								

10.2 EMI Header

EMI uses a 4096 byte ASCII header followed by binary data. Even though you can read the header you will change the size by editing it and binary data will not be read in correct any more. However EMI uses the undocumented keyword “DataSet:” before the binary section starts. It will depend on the software what finally happens....

10.3 EMI Time series

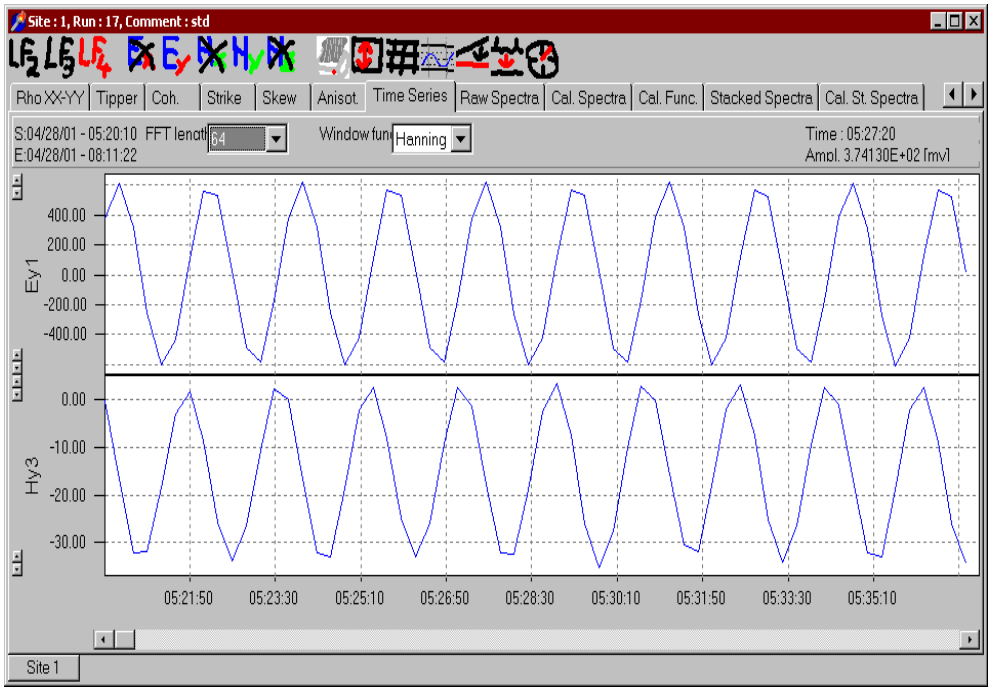
The time series have different endings depending on the sampling rate. For each sampling rate a system calibration file has to be read in.

extension	sampling rate
t01	1000 Hz
t02	500 Hz
t03	100 Hz
t04	60 Hz
t05	50 Hz
t06	25 Hz
t07	12.5 Hz

11 Performing a Test

Assuming that you are not sure about your results obtained with your ADU you would perform some tests.

Here we look at data sent to metronix with the information that a signal with 7.3 nT in H and 470 mV in E was generated with a 100 s period, recorded with 64 Hz (LF2):



E and H of test signal

`tsmp -nosacii -wl 1024 -trf theo <filename>` produces a log file; here we see for H channel

```
evaluated samples: 642
voltage peak peak: 40.5546mV
max: 4.2383  min: -36.3163 mV
mean:-16.0099
over all standard deviation: 13.2093
voltage peak peak / std deviation : 3.07016
(ratio is: square wave 2, triangle wave sqrt(12) = 3.46, sine
wave 2*sqrt(2) = 2.83, noise ca. 6-8;)
```

and E

```
evaluated samples: 642
voltage peak peak: 1232.55mV
max: 609.153  min: -623.4 mV
mean:-8.19573
over all standard deviation: 436.59
voltage peak peak / std deviation : 2.82314
....
```

We can see that the statiscal analysation shows that a sine wave was used (voltage peak peak / std deviation).

Also we assume that the given values in nT and mV showing the effective range: if we multiply the values for E (which do not need a transfer function) with $2\sqrt{2}$ we obtain 1232 mV_{pp} / $2\sqrt{2}$ = 435 mV_{eff}.

The spectral analysis of the H field results in:

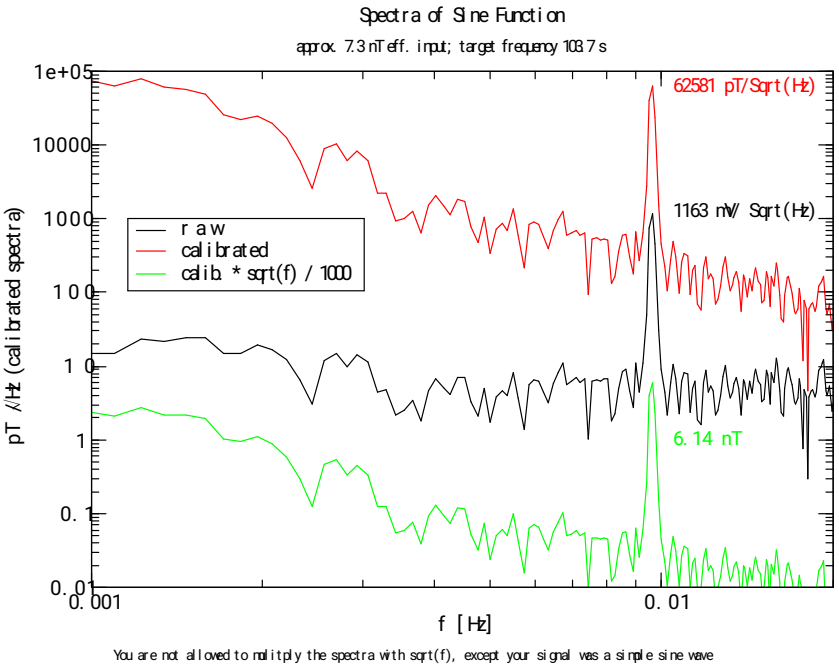


Figure 11.1: Spectra of Sine Function

The result is 62.581 $nT/\sqrt{Hz}$ at 103.7 s or multiplied with the square root of the frequency 6.14 nT (what we can do here assuming a sine wave).

The other way to obtain the same result is looking at the time series window, reading that the peak/peak value is about 36\, mV. Using a basic formula for very low frequencies (below 0.1\, Hz) for the coil sensitivity $S = \frac{0.2\,V}{nT\,Hz} \cdot meas frequency$ (or refer to then figure of the built in transfer function) we obtain $H_{p/p} = \frac{V}{S} = \frac{0.036\,V \cdot nT \cdot Hz}{0.2 \cdot 0.01\,Hz} = 18\,nT_{p/p}$ and therefore $18/2\sqrt{2} = 6.36\,nT_{eff}$.

In other words: the ADU meets the expected input. The frequency resolution of the 512 point FFT is 1.2 Hz @ 100 s, the nearest frequency to 100 s was 99.9 s and was not met.

Also the precision of the induction coil is << 1%. We assume that the source was not exactly calibrated. On the other hand on can say that the system works. If for example the chopper amplifier would not have switched you would not see that signal at 100 s.

12 Using tsmp for Sensor Test

The most common test is a parallel sensor test. With all sensors you should obtain the same results. The easiest way does not lead to a success: subtracting the time series from each other:

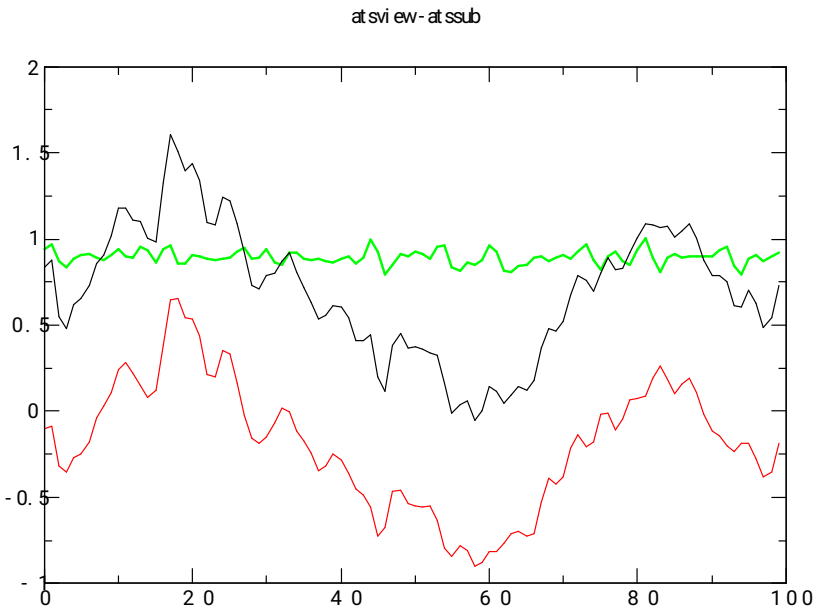


Figure 12.1: Subtracted Time Series

You always get offsets which a are showing residuals in the time domain. Also the coil is calibated in the frequency domain.

Since MT is interpretation in the frequency domain we don't care about that.

With the command tsmp -nspw -trf auto file1.ats file2.ats we start the interpretation. Amplitudes, noise and coherencies are written to disk:

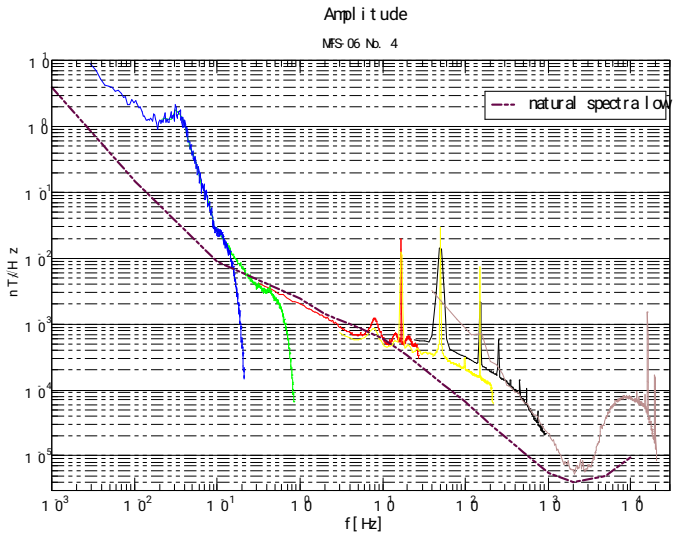


Figure 12.2: MFS-06 Mag. Field Amplitudes

The amplitudes showing the “expected” behavior: decreasing with higher frequencies. The doted line shows the expected amplitudes for the low natural field spectrum. The “drop down” of the curves is caused by FFT when you are going closer to the Nyquist frequency.

Here it is recommend to change to a higher band (LF2 $\rightarrow$ LF1) instead of interpreting at the band limits. The peaks are related to the man made noise ($16\frac{2}{3}$,50,150...Hz and radio transmitters above 16 kHz).

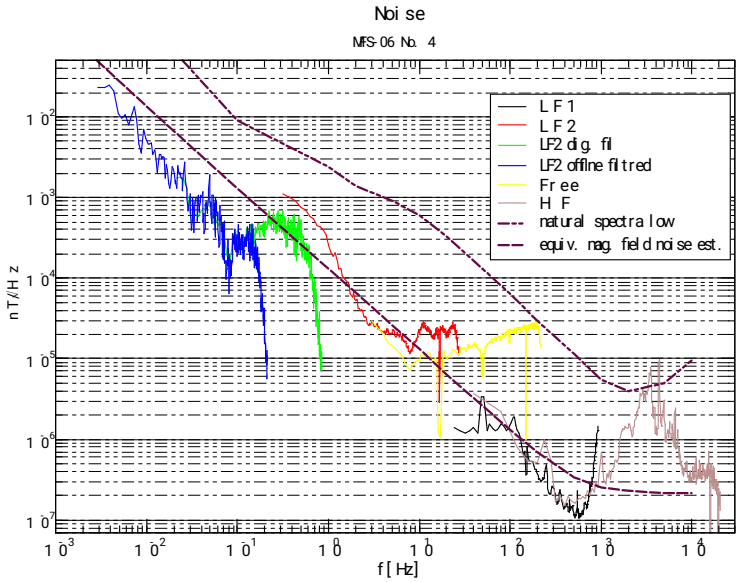


Figure 12.3: Noise Chart

The noise chart shows that the MFS-06 is able to record an MT signal even when the natural field is low.

The ramp at 10 Hz for the 512 Hz and 4096 Hz sampling rate is caused by the chopper amplifier. Hence that the increasing noise is higher than the theoretical coil function but still lower than the low natural spectrum.

The coherency now shows what is not visible in the subtracted time series: the coherency is almost 1. Also visible: coherency decreases at the edges of the bands and the typical gaps at 3 kHz and 5 s which makes live hard for MT interpretation.

With the option -ll n (lower limit) and -ul n (upper limit) you can change the amount of frequencies in the FFT output file to reduce the band overlap:

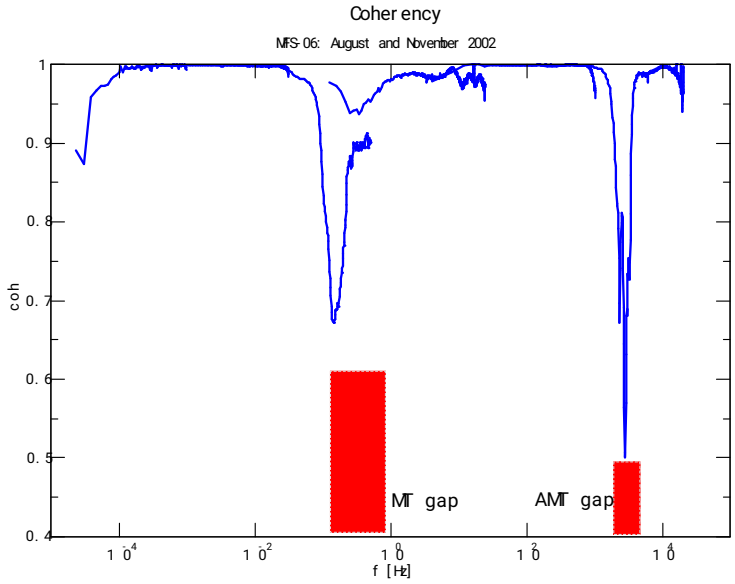


Figure 12.4: Coherency

In the figure above the coherencies for the low frequencies were calculated from survey data; the site spacing was 8 km.

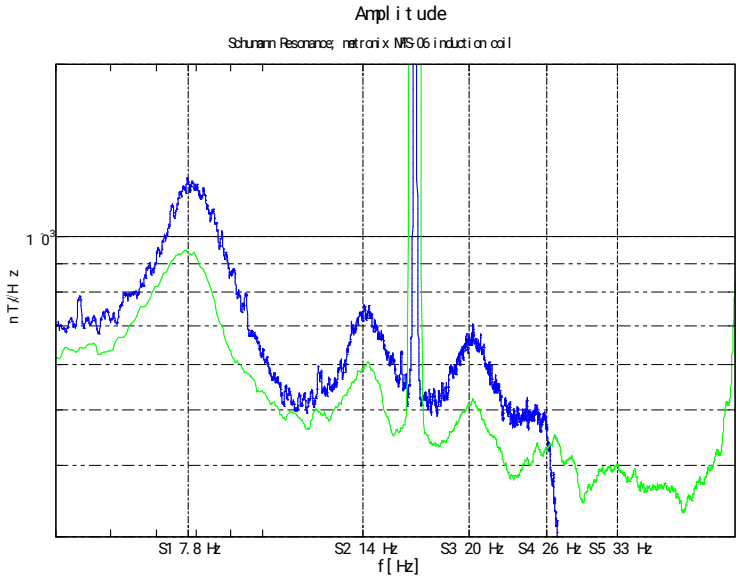


Figure 12.5: Schumann Resonances

A detail of the amplitude graph clearly shows the Schumann resonances (in older books you read that they can not be recorded with induction coils ... take this as a quality example of the MFS coils)

13 Destepping

Be ensured that this algorithm destroys your data! Make a backup!

It can happen that you had bad contact to your electrodes and you get the following effect:

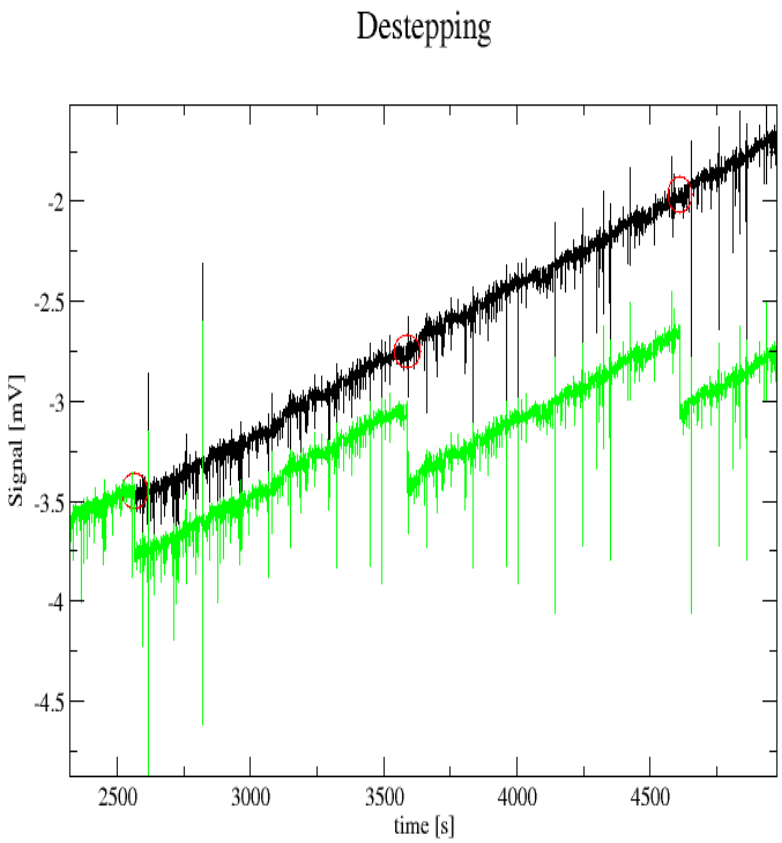


Figure 13.1: Destepping

There might be a possibility to remove them using the command:

```
tsmp -start 136500  -nspw -dstw -destep 40 0.1 1 0.8
```

where -dstw write a new ats file to the disk (with 136500 samples less!) and -destep tries to find step with a “shoulder – length” of minimum 40 samples, a minimum step height of 0.1 mV using one point for “relaxation” and a “skewness” of 0.8 for both sides of the step.

For tests write an ascii file with:

```
tsmp -start 136500  -tc file.ats
cp file.dat orig.dat // make a copy
tsmp -start 136500  -tc -destep 40 0.1 1 0.8 file.ats// test
your parameters
xmgrace file.dat orig.dat file.steps // view your result
tsmp -start 136500  -nspw -dstw -destep 40 0.1 1 0.8 //
overwite your atsfile
```

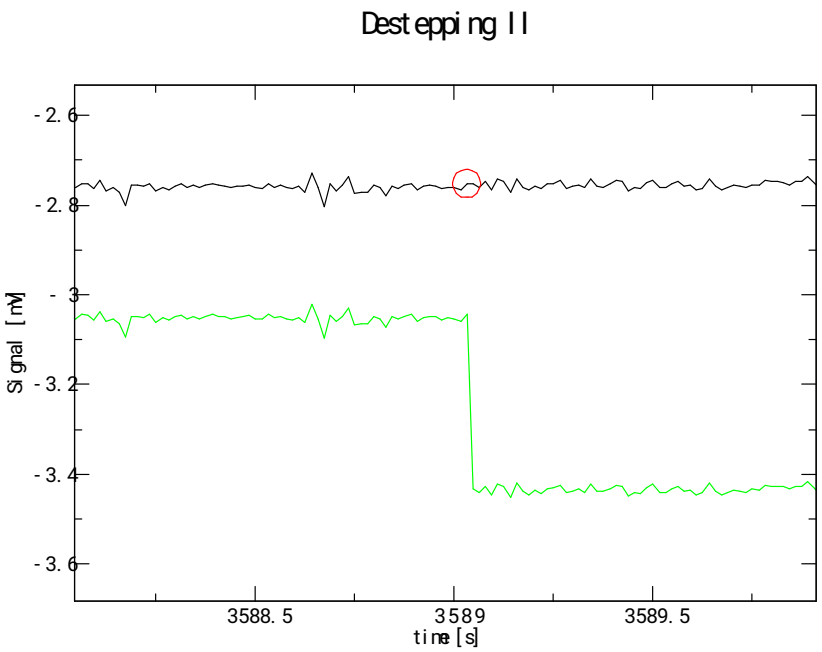


Figure 13.2: Destepping Parameters

The step size is easy to understand – it is the minimum what you guess that it is a step and not measured data.

You might prefer to use the -atm merge or -atm new option instead of -dstw option if you only have a few steps to exclude from processing.

14 Despiking

Be ensured that this algorithm destroys your data! Make a backup!

This algorithm tries to find out where spike are:

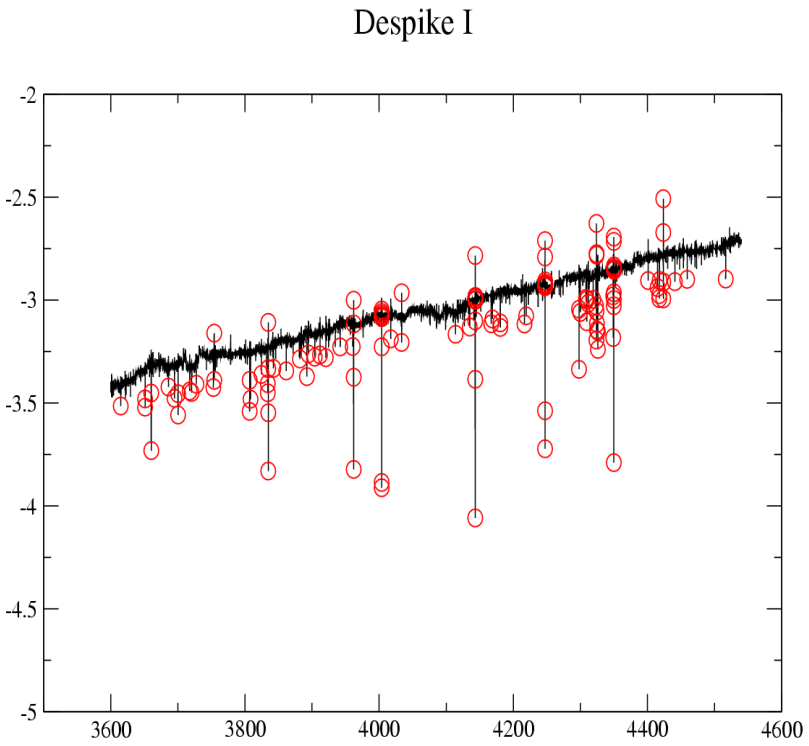


Figure 14.1: Despike

A spike is defined as a sudden impulse which is extremely short and does not belong to a natural signal. On both sides on the spike the signal should have the same level (defined by the skewness paramater which should be in the range of 1.1 – 1.2 and the “ shoulder length” over which this parameter is checked).

The minimum spike hight must be defined – try a conservative setting to be sure not to remove the natural signal from your time series:

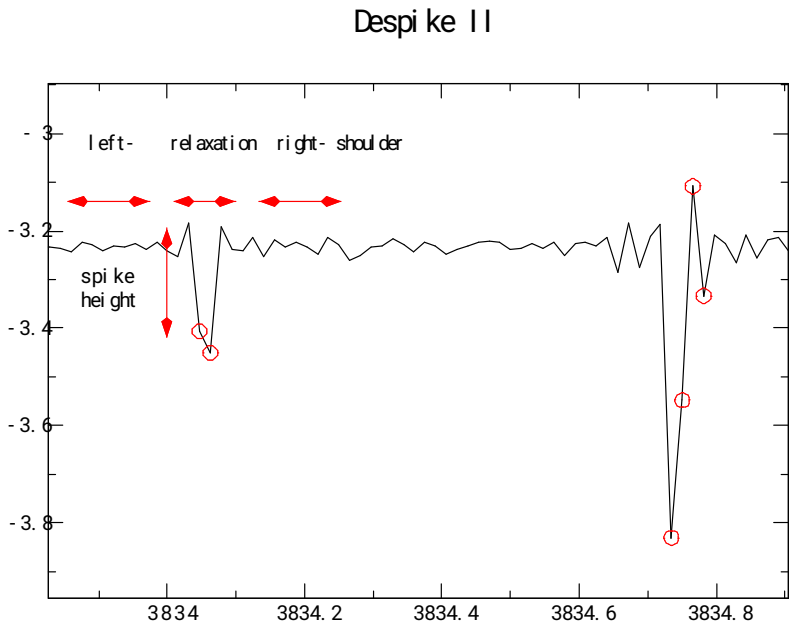


Figure 14.2: Despike Parameters

The last parameter is called relaxation. That is in general the width of the spike. Because the algorithm is using a moving average I try to avoid taking an average over the spike itself.

Use the same procedure as described in the destepping chapter (without activating -dspw here) before you overwrite your atsfile with:

```
tsmp   -nspw -dspw -despike 10 0.05 8 1.2 file.ats //
 overwrites your data!


tsmp   -nspc -nspw -despike 10 0.05 8 1.2 -atm new
 018C01XC.ats // creates a corresponding atm file for mapros
```

(despike with 10pts shoulder length, 0.05 mV spike level, 8pts relaxation

and anisotropy of max 1.2)

The detected outliers will be replaced by “normal” values from the left shoulder. Do use -noscale_e if you want to compare your results with mapros (otherwise spike values interpreted as mV/km! If you want a written index of detected spikes do not use and supply -tc; hence that also the corrected time series data is written to disk; repeat without despike option to see the original time series data.

If you use -atm merge or -atm new instead of -dspw spike will be excluded from processing from MAPROS; -dspw stands for write the despike data.

15 Standard Deviation

However, despiking works only reliable with the electric fields. A coils does not produce sharp spikes, as shown below.

Here we calculate standard deviations using

tsmp -nspw -stddev 0.3 6 64 -wl 128 037C01XC.ats

and getting a message like

```
atsdata::stddev_array -> stddev min: 0.877842   stddev max:
466.381
atsdata::stddev_array -> stddev mean: 5.00348   stddev
median: 4.02622  stddev: 9.41619
atsdata::stddev_array -> 1831 window tags selected FOR
processing and 5740 NOT

main -> std dev analysed windows: 13310
main -> writing 037C01XC.stddev ...
main -> done
main -> writing 037C01XC.stddev_used ...
possible use of stacks: 6655

******** start spectra ********

init(1) of 1

v_spectra::init -> 886 stacks initialisied
v_spectra::calc_sp -> Hanning window and detrending used
....
```

Use -tc if you want to plot the time series together with the standard deviation vector and the vector of used segments.Use instead -nspw to suppress the output of ascii files as much as possible, e.g. When you repeat the calculation several times and want to compare the different result in the spectra.

We slice the time series into windows of 128 point with an overalapping of 64 points. For every 128 point a standard deviation is calculated. The median of all standard deviations is calculated. -stddev 0.3 6 says that events are selected which are between 0.3 and 6 times the median of all standard deviations (that is stddev median: 4.02622 here).

Standard Deviation

Unlike using destep and despike we can not write a new atsfile. If we want to use the analysis for later processing we use -atm new or -atm merge to store the results in the atm file.

Hence that especially the lower limit allows you to exclude data with low energy or bit noise.

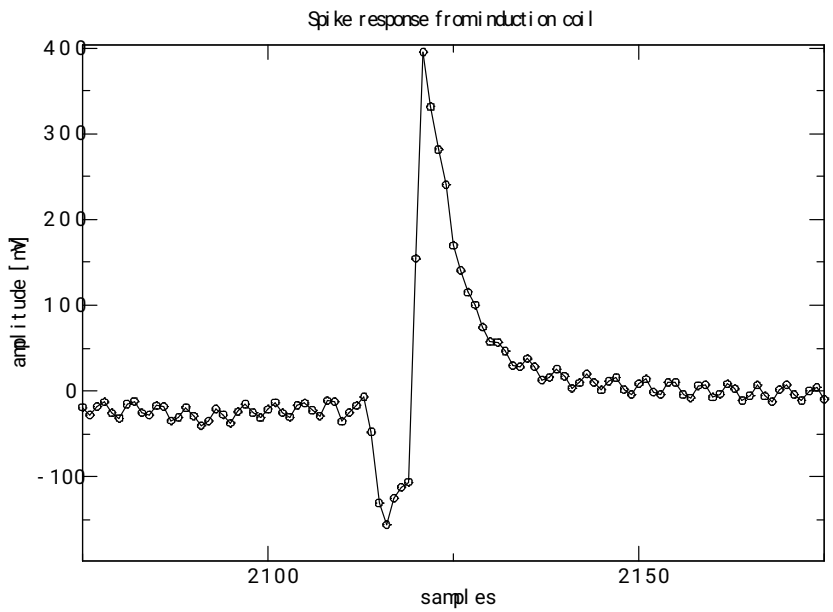


Figure 15.1: Spike Response (Coil)

General Note

Hence that nor destepping neither despiking can “repair “ a corrupt time series.

16 Mapros

Hence that MAPROS is a database oriented program. The headers are stores in a database. So if you manipulate ats files you have to re-import them to make sure that MAPROS take notice from the changes.

Also MAPROS locks the files; in other words you can not manipulate atm files while MAPROS is running on the same data.

17 System Dependent Variables

datatype/32 bit UNIX	*byte*	*bit*	*range*	*addressable*
short int	2	16	-16,384 -> +16,383	(16 kB)
unsigned short int	2	16	0 -> +32,767	(32 kB)
unsigned int	4	32	0 -> +4,294,967,295	(4 GB)
int	4	32	-2,147,483,648 -> +2,147,483,647	(2 GB)
long int	4	32	-2,147,483,648 -> +2,147,483,647	(2 GB)
signed char	1	8	-128 -> +127	
unsigned char	1	8	0 -> +255	
float	4	32	climits	
double	8	64	climits	
long double	12	96	climits	

ILP 32 variables; floats are defined in the corresponding header files of you C++ compiler

Since ats files are generated on a PC the variables will have that size. In emergency case one can pick up the LSB value from the header and than start to read the file after 1024 bit; he has to make sure that the 32 bit long int are converted correctly.

Tsmp tries to find out whether your machine uses high byte order (big endian, network byte order, SPARC, MC68000) with the configure script. If so, all values will be converted from little endian (INTEL, VAX) to that format automatically. However: you must run a test with the delivered ats file to check the results.

Tsmp should also run on LLP64 and LP64 machines, NOT on LP32 (DOS, Windows 3.xx). Critical variable is DateTime which is long and 32bits.

On ILP 64 bit machines more work has to be done.....

Some machines (CRAY) seem to use 32 bit short and 64 int there; might be a possibility to port that.

Datatype	**LP32**	**ILP32**	**ILP64**	**LLP64**	LP64
	2/4/4	4/4/4	8/8/8	4/4/8	4/8/8
char	8	8	8	8	8
short	16	16	16	16	16
int	16	32	64	32	32
long	32	32	64	32	64
pointer	32	32	64	64	64

Programming Models

Since I am using UNIX/LINUX I am using currently the ILP32 model. The atsheader uses the LP32 model. For that one has to use short instead of int wherever int was given in the header.

18 ADU-07

New file names as:

039_V01_C08_R753_TEX_BH_16384h.ats

where

039 ADU serial

XML **V**ersion 01

Channel 08

Run 753

channel **T**ype Ex

Board H, M, L

512 h/s **h**ertz / **s**econds

19 Tips and Tricks

19.1 Getting an Overview of Recorded Time schedules

19.1.1 I have a set of recordings at different sites

which where sometimes parallel. How to find out?

Go Up to the timeser directory in MAPROS.

Use:

```
find ./ -name “???A???C.ats” -exec tsmp {} \;
```

to genereate log files of all LF2 band for example (or use “???A*.ats”) for all bands.

Then use

```
find ./ -name “???A???C.log” -exec grep --binary-files=text
 excel {} \;
```

to grep the excel section from the log file (the binary option is optional if you have grep refuses to scan the log files).

The result looks like:

```
excel: 0107_run_03     7.12.2001 9:18  7.12.2001 18:7
excel: 0107_run_26     3.12.2001 17:15 3.12.2001 19:31
excel: 0107_run_27     3.12.2001 17:915 3.12.2001 19:31
excel: 0108_run_03     6.12.2001 14:0  6.12.2001 22:49
excel: 0108_run_27     4.12.2001 17:45 4.12.2001 20:35
excel: 0109_run_02     7.12.2001 11:42 7.12.2001 20:31
excel: 0109_run_03     7.12.2001 11:42 7.12.2001 20:31
excel: 0109_run_04     10.12.2001 11:59 10.12.2001 20:48
excel: 0109_run_27     6.12.2001 13:15 6.12.2001 16:56
```

put this into a file and import to Excel or StarOffice.

In Excel you choose Date into Number; provide a column which now

contains length = (stop-start). Then choose diagram type “balance beam” (Schwebebalken) start, length.

An example might look like this:

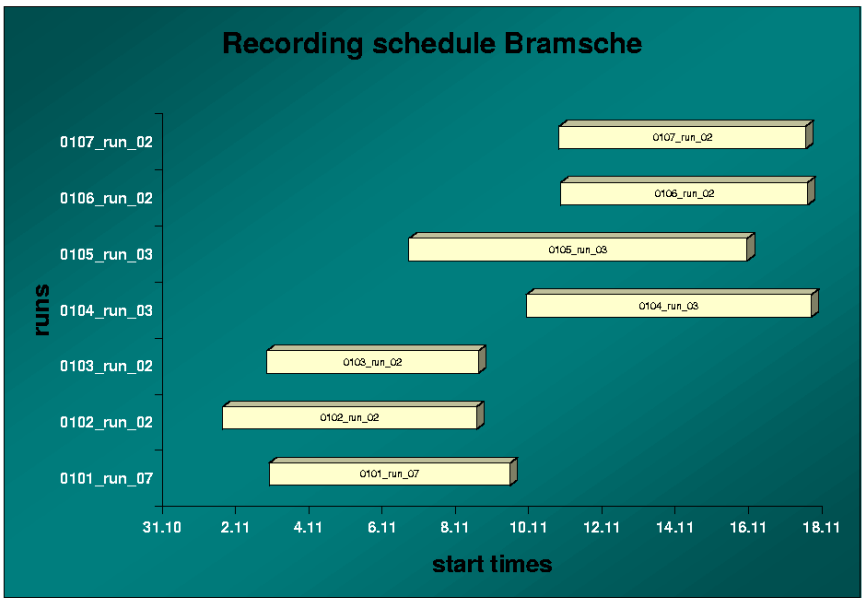


Figure 19.1: Recording Schedule

19.1.2 I want to display Hx, Hy Hz in the same plot window

use `-add mean` to remove the mean of the time series

use `-detrend` to enhance the scaling (if you do need the trend)

use `-tc` for getting the x-axis in seconds from start; use `-tc -abst 0` to plot with UNIX time stamp; use `-abst -12` to shift with -12s; use `-tc -abst 0 -dateformat ISO` to plot a date with year, month day and so on; can be read by xmgrace (set X-axis format from general to date). Hence that xmgrace hangs if the time interval in ISO format is to small (ref. 19.1.9)

19.1.3 Changing E-Field Positions

You can change the positions of the electrodes by using

```
tsmp -corr -wrh -spos 0. -42.2 0. 0. 42.2 0. -postodip ???
 B*.ats
tsmp -corr -wrh -spos -51. 0. 0. 48.3 0. 0. -postodip ???
 A*.ats
```

The switch postodip calculates the dipole length and direction from North to East. Otherwise – if dipole length and angle are correct but the sensor positions are not correct (Mapros uses the sensor positions!) use

```
tsmp -corr -wrh -diptopos ???A*.ats ???B*.ats
```

19.1.4 Upgrading the Header to Version 0.73

Header versions from 0.73 contain UTM and Gauss Krueger position information. The ADU however writes 0.72.

use -corr -wrh -utm -gk to calculate both positions simultaneously, put these coordinates into the header and upgrade. Together with -mrd 12 for example you can force a meridian for the calculation. Hence that Gauss -Krueger coordinates can use 3, 9 ,12 degrees and so on, UTM however 3, 9, 15. Giving -mrd leads to two different meridian lengths: 12 for Gauss-Krueger and 9 for UTM coordinates.

19.1.5 Working With Multiple Files

On Unix you can use the command: find ./ -name “*.ats” -exec tsmp -corr -wrh -utm -gk {} \; to change all headers; or simply use tsmp -options *.ats when you are working on a single directory.

You also can use wildcards like this:

```
find ./ -name “???[AB]????.ats” -exec tsmp -corr -wrh
 -sensortype EFP06 {} \;
```

will change the header entry for all electric channels.

Try with [CDE] and MFS06 for the magnetic.

```
Find ./ -name “???[A]????.ats” -exec tsmp -corr -wrh -spos
 -45.0 0 0 48.4 0. 0. -postodip -angle 0 -diplen 0
{} \;
```

sets the positions entries for the electrodes and (here pure North) an calculates automatically dipole length and angle to North.

```
Find ./ -name “???[C]????.ats” -exec tsmp -corr -wrh -sensser
 124 {} \;
```

19.1.6 Changing Calibration

If sensor type, serial numbers are not specified one can easily change the header entries with

```
find ./ -name “???[C]????.ats” -exec tsmp -corr -wrh -sensser
 124 -channeltype Hx -sensortype MFS06U{} \;
```

where channel type is max 2 characters and sensor type is max. 6 characters. From sensor type and serial numer tsmp will automatically search for the correct calibration file in the working directory or in /usr/local/share; you can use than -trf auto. Use -sensorcal ABCDEFGH.IJK for programs reading this entry.

19.1.7 Working with CDROMs

When you are working on a write protected drive use -out_dir /home/mtx/data to redirect all output to your working directory. Only options which overwrite your ats files will not work then.

19.1.8 Customizing XmGrace

Simply format your style, lets say log-log for spectra and log-lin for your coherencies; name axis and so on.

Then save the parameters. Later call xmgrace together with this parameter file or even better: put an alias into your .bashrc file:

Aliases : xl=’xmgrace -par ~/bin/lx.par’; xll=’xmgrace -par ~/bin/lxy.par’.

19.1.9 Absolute Date

tsmp provides different date formats. The option -tc activates output in seconds in the first column, starting with 0s (relative time). Together with -abst you get the something similar like UNIX time (secons since 1970) but as doubles and not as integers. For low frequency recordings over several days you also can use -tc -abst 0 -dateformat ISO; here the first column of your ascii file contains a ISO date string which can be read by xmgrace. Hence that this does not work with short term recordings because xmgrace needs at minimum 24hrs to label the x-axis.

In xmgrace do edit, prefernces, date hint: ISO; in case change inside the date hint field '12:00:00' to '24:00:00'. Then set the properties of the x-axis to the preferred date format.

19.1.10 ASCII files

With most output (except spectra) will be suppressed.

With -nspw suppresses the conversion of the ats file into ascii; selection vectors are written.

When you make a analysis of the standard deviation, spikes or steps than you should generate an ascii file with a time column (-tc). Repeating your analysis the you need NOT to write this file again and again, use -nspw instead.

If you want to compare the results of different analysis in the frequency domain can be used; the spectral section will automatically take care of you exclusions/inclusions.

Use the “-rda sample_freq column year month day hour min sec” option to create an ats file from ascii data:

```
tsmp -rda 512 1 2000 12 24 9 15 0 -wrh -corr -atsoutfile
 099C01XF.ats input.dat
```

for a file recorded with 512 Hz at Christmas 2000 at breakfast time 9:15:00 ill be converted in 099C01XF.ats. Column 1 (that is the first here; not C

convention) is used of the ascii file conversion.

The -corr option allows you to fill in a more complete header:

```
tsmp ..... -corr -sensser 114 -aduser 12 -samplefreq 512
 -sensorcal MFS07.txt .....
```

Hence that MAPROS et al using the filename!! to determine sample frequency, run number, E/H channel; here: Hx.

19.1.11 Output Data

tsmp offers you ascii and binary output; for binary output you use:

```
tsmp -sync2 -merge -detrend -tc -wbin -abst 0 002A02AE.ats
 036A02AE.ats out.bin
```

this writes a binary (64bit, 8byte) output: unix_date (as double!) file1[0] file2[0] unix_date file1[1] file2[1] ....; both time series might have different start and stop times; a trend is removed.

```
tsmp -merge -wbin float 002C02XE.ats 002D02YE.ats
 002E02ZE.ats 002A02AE.ats 002B02BE.ats out.bin
```

does the same without detrend and writing a time; this results in a simple raw conversion of Hx Hy Hz Ex Ey into a binary file. The option -float indicates 32 bit data.

The option -merge works even with one file.

```
xmgrace -source pipe -nxy “bintoa -bytes 8 -cols 3 out.dat”
```

19.1.12 Converting

Converting files to the ats format should work quite automatically. To generate a valid ats files we need virtual ADU and Run number, sampling frequency and and channel type. Using an EMI file you only need to provide run number and ADU number

```
tsmp -run 12 -aduser 98 28153286.t05
```

Hence that -convert does not make use of any scaling: simple transfer from one format to the other. If you want include system transfer functions refer

to [sec:Real-B-Field]

19.1.13 Display Parallel Recordings

I have time series at two sites which are almost parallel – how to display them: use -tc -abst 0;

19.1.14 Remote Reference / Coherency / Crosscorrelation /Raster

For these options it might be necessary to re-filter the time series again. Reason: If the sampling period is 16 s and the start times differs with 3 s the results of these processings are not correct. Since the lowest recorded band of the ADU is 2 Hz data can always be filtred again. Use:

```
tsmp -filw 32 -raster 512 *D.ats
tsmp -filw 32 -raster -1 *E.ats
```

The first option grids the start time to a 512 s raster

all 16 s sampled data have now start times of n * 512 s since 1970. For the lowest band E this is not possible because of the filter length (the time shift is 3768 s when you filter the 16s band). But for all filtered bands the offset will be the same while disabling the raster.

19.1.15 Filtering Free Bands

While filtering “Free Bands” the band shift does not apply. Solution:

```
tsmp -filw 32 -run 29 -noscale_e *F.ats
```

will write the filtered values to run 29 in this case.

19.1.16 Coherency Test

When you have carried out a recording with parallel oriented sensor exactly provide two filenames and a window length and tsmp automatically calculates coherency and noise of the tested sensors (tsmp -wl 1024 -sync2 1.ats 2.ats). Use the atm or stddev options to exclude parts of the time series from spectral calculation. When using sensors of different type you must use one of the -trf options! See also [sec: raster].

19.1.17 Real B Field

I want to see the real B field instead of milli volts.

Here we have to transfer into the spectral domain with a rectangular window(!) and use the dc part of the fft then apply the transfer function of the coil and then transform back into the time domain. A new run number (two digits) has to be provided if you want the result to be written into a new ats file (and always switch -nspw) use:

```
tsmp -nspw -wl 512 -rect -back -trf auto -run 99 *.ats
```

hence that the window length of 512 works like a high pass pass filter! More precise you can seperate H and E with:

```
tsmp -nspw -rect -wl 1 -back -run 86 -dcfft -trf theo
 *C01*.ats *D01*.ats *E01*.ats
tsmp -nspw -rect -wl 1 -back -run 86 -dcfft *A01*.ats
 *B01*.ats
```

it also works on any subset of your data:

```
tsmp -nspw -rect -start 4000 -use 234567 -wl 1 -back -run 86
 -dcfft *.ats
```

Re-writing of the E field calibrates the E-field to mv/km; this can also be done in time domain with:

```
tsmp -run 86 -overwrite -detrend *A01*.ats *B01*.ats
```

You also can also transform the whole time series at once with the reserverd key -wl 1. (uses ALL points for one FFT) :

```
tsmp -nspw -wl 1 -rect -back -trf auto -run 99 -detrend *.ats
```

to get more safety you might use -detrend for detrending the complete time series. Do not use -wtrend in these functions.

If you converting from other MT systems rather than ADU-06:

```
tsmp -wl 1024 -rect -back -trf auto -run 99 -wtrend -resample
 512 01250011.t02
```

here also the software tries to read the transfer functions; the resample

option trys to adjust the sampling frequency of the original time series.

If you want to write scaled time series as ascii data only use combinations with:

```
tsmp -nspw -wl 1 -rect -back -trf auto -detrend -ascii *.ats
```

Be aware that a 30MB file will be expanded to approx. 500MB in main memory.

19.1.18 I want to check files from a read only device

Use for example

```
find ./ -type f -name "*XD.ats" -exec tsmp -out_dir
 /home/bfriedr -gk -utm -nodata {} \;
```

to write the output log files in a different directory.

Here a temporary UTM and Gauss Krueger coordinates calculation is switched on.

19.1.19 What is Linear Convolution

Linear convolution can be used for time domain filtering. This will work only if the frequency to be removed is very stable. The algorithm is NOT perfect but can be used for comparing ats data with data from analogue, filtred recordings.

You have to understand that any filtering applied to ats time series will reduce the quality. You can use a filter for displaying data but NEVER for processing them. The following command convolutes the time series with 1024 pts windows and tries to remove 16, 50 and 150 Hertz:

```
tsmp -lin_conv 1024 16.666 50 150 [-run 99 -overwrite]
 888D22YC.ats
```

(optional you can write a new ats file).

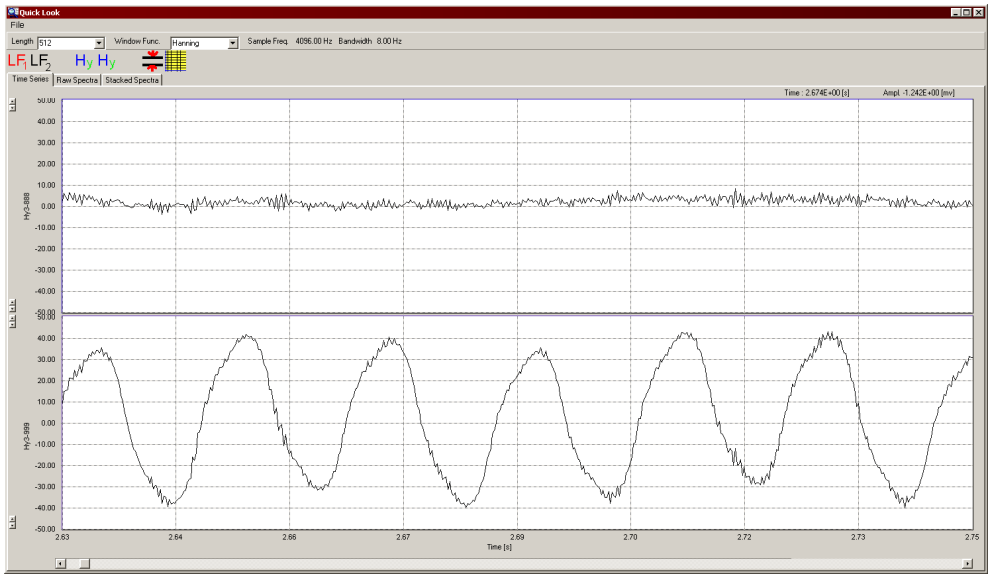


Figure 19.2: Linear Convolution (bottom orig. time series)

If we compare now:

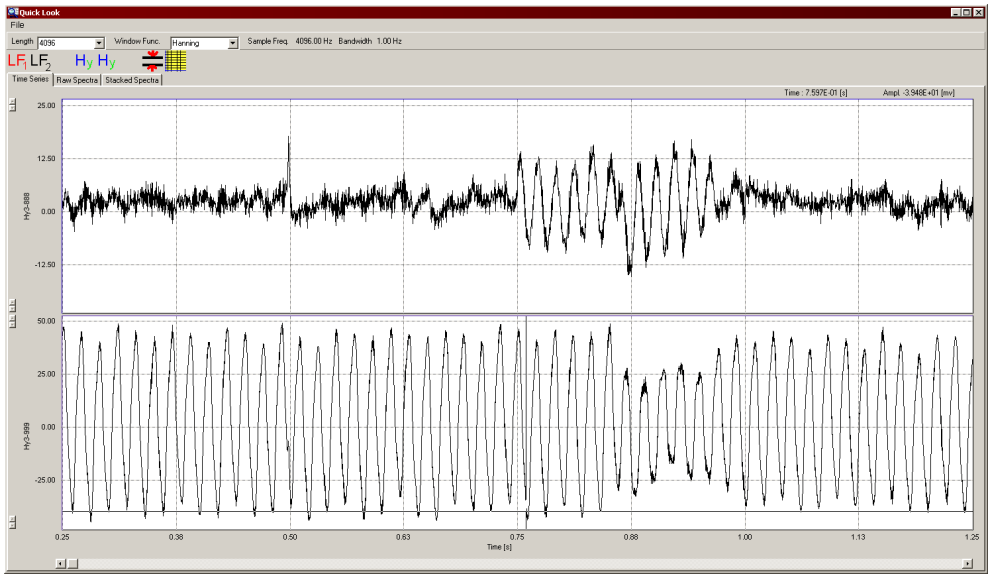


Figure 19.3: Convoluted with shifts

In can happened that some segments are not stable or and offset between two windows are visible

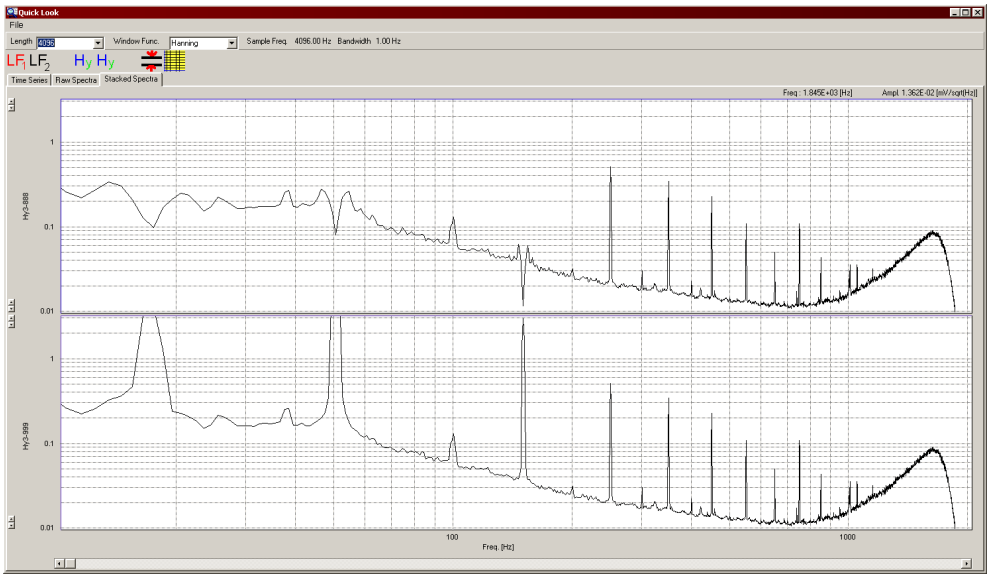


Figure 19.4: Spectra after Convolution

If everything works well the spectra should look like this. Hence that for any spectral interpretation there will be absolute no advantage in using this filter.

19.1.20 Auto- and Cross-correlation

With:

```
tsmp -ccf 0 0 20000 -wl 8192 002C02XD.ats 036C02XD.ats
```

a cross correlation will be calculated and a file with the ending _ccf.dat written (see also [sec: raster]). The first 0 0 indicate that both time series will be read in starting at sample 0; the ccf will be calculated over the first 20000 points (0 would indcate that the maximum possible points will be used; that can take several 20 minutes on a GHz Pentium).

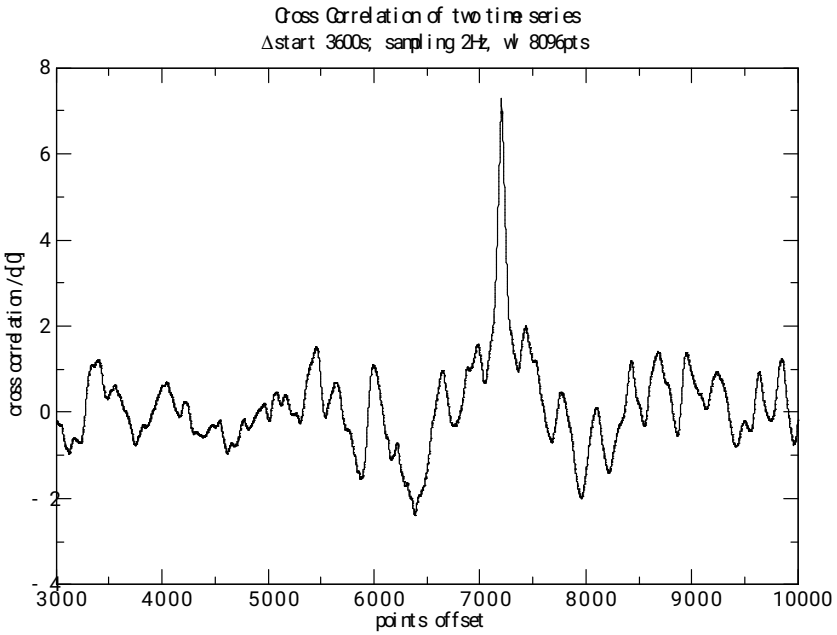


Figure 19.5: Cross Correlation a)

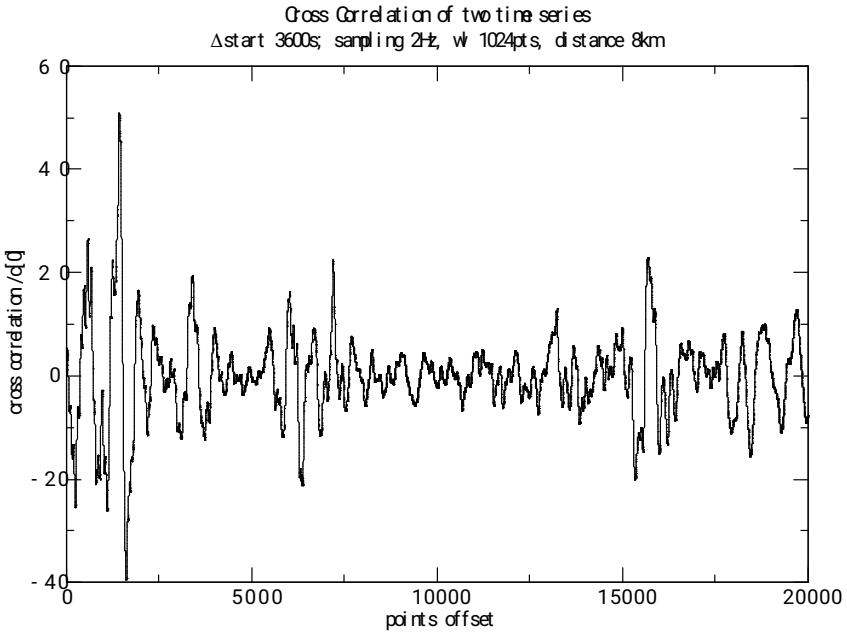


Figure 19.6: Cross Correlation b)

Hence that the cross correlation very strongly depends on the coherency length. As longer you chose the window length -wl as more reliable the result of the ccf will be. The second example with a window length of 1024 points used shows a misleading result! The offset of the time series in reality is 1 h = 3600 s = 7200 samples in this example here. A indicator of the right window length seems to be that the values of ccf are small.

The cross- (auto-) correlation is constructed that way that the second file is moving on (20000 points in the given example above); **if start time of file1 is earlier than start time of file2** you should get the expected result by using tsmp ... file2.ats file1.ats

This function can be used if you coming to the conclusion that two time series might not be synchronized – or to synchronize a time series which was started without GPS.

19.1.21 Truncate and Shift

Assume two systems were not running synchronously.

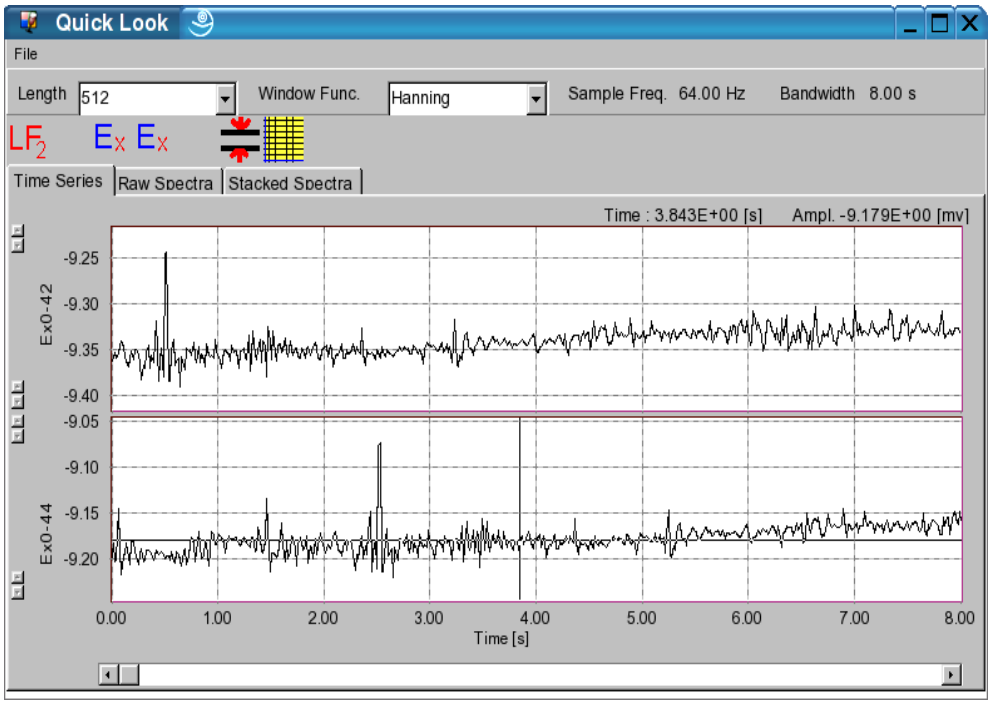


Figure 19.7: Time Shift

When using quicklook the result will look like this.

Hence that quicklook does not evaluate the start time in the header.

To correct a shift of 2 seconds at a sampling frequency of 64 Hz do:

```
tsmp -start 128 -overwrite 044a01ac.ats
```

(tsmp corrects the start time automatically, so re-do that)

```
tsmp -corr -wrh -time_s -2 044a01ac.ats
```

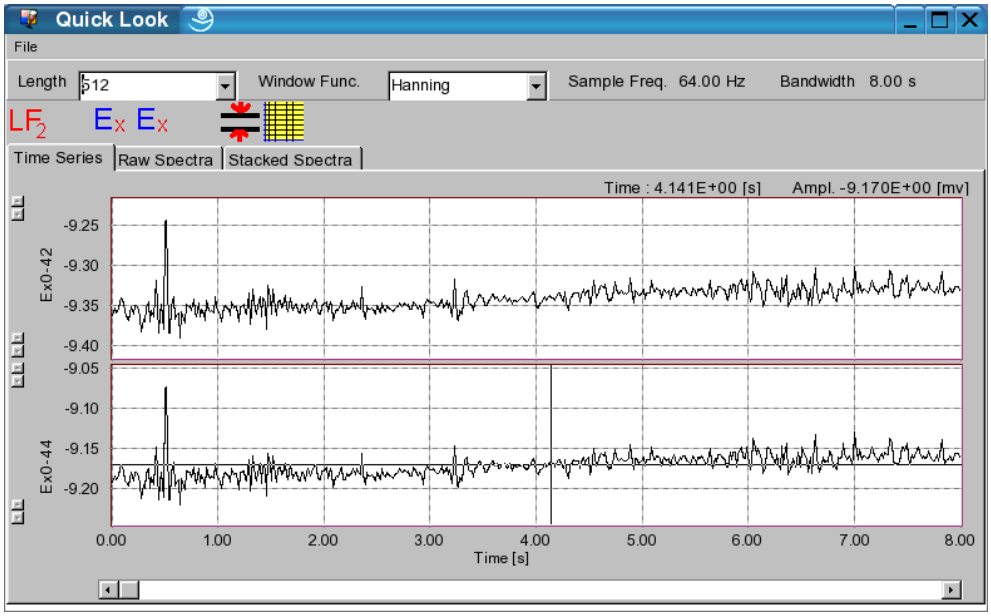


Figure 19.8: No Time Shift

If you are not sure wheather the shift is exactly on or mor full seconds you have to use the cross correlation function to find out the amount of shifted samples:

```
tsmp -ccf 0 0 1000 -wl 8192 042a01ac.ats 044a01ac.ats
```

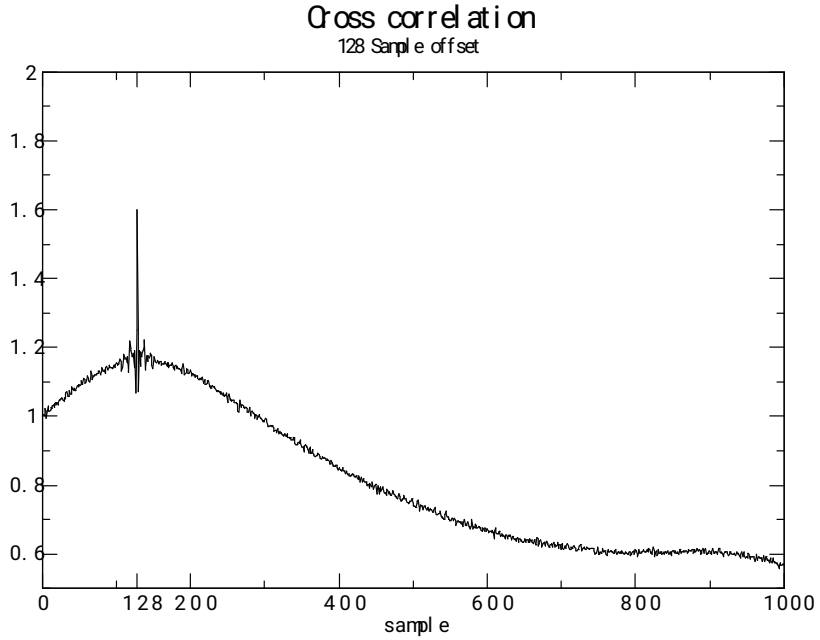


Figure 19.9: CCF of 2s Time Shift (128 Samples)

This gives you the offset between two file in SMAPLES; ref to 19.1.20

19.1.22 Mixing ATS Files

Another purpose can be that you have two stations running but one with a chopped E-field.

You want to process one ADU with the electric field of the other, files have different start and stop times. Here I take ADU 002 for magnetic and ADU 09 for electric field. copy all corresponding files into one directory, and:

```
tsmp -corr -wrh -aduser 002 *39*ats
mv 039A03AD.ats 002A03AD.ats
mv 039B03BD.ats 002B03BD.ats
tsmp -sync2 -run 4 *.ats
```

The -run 04 indicates that the ouput will be written to a new run.

19.1.23 Calibration

Sensor (in progress)

The ADU-06 provides a calibration facility. We are looking for a calibration function with the units $\frac{V}{nT\ Hz}$.

First we submit a job with “CAL Sensor” switched on. Then we submit a job with “CAL Internal” switched on. Both jobs with the same sampling rate and the same amount of samples.

At the first job a signal is send through the calibration coil of the MFS with the calibration constant $c = 4\frac{nT}{V}$ and the resonse function is measured.

At the second job the sam is done internally (coil does not need to be disconnected). If we divide first by second we obtain the transfer function of the coil:

$$T_{coil} = \frac{U_{sensor}}{U_{internal}} \cdot \frac{10}{1010} \cdot \frac{1\mathrm{V}}{4\,nT} \tag{19.1.1}$$

the result has to be normalised by frequency to obtain the same units like in the Metronix calibration files.

The factor is a result from the internal wiring:

$$U_{measured} = U_{input} \cdot \frac{10\,k\,\Omega}{1010\,k\,\Omega} \qquad \text{where } U_{input} = 2.5\,V_{pp} \tag{19.1.2}$$

If the calibration frequency was 0.125 Hz the command will be (../site001/ contains “Cal Sensor” data)

```
tsmp -wl 4096 -calib 0.125 -ul 20 ../site001/036C02XC.ats
 036C02XC.ats
```

```
tsmp -wl 20480 -calib 512 -ul 1 ../site001/036C02XA.ats
 036C02XA.ats
```

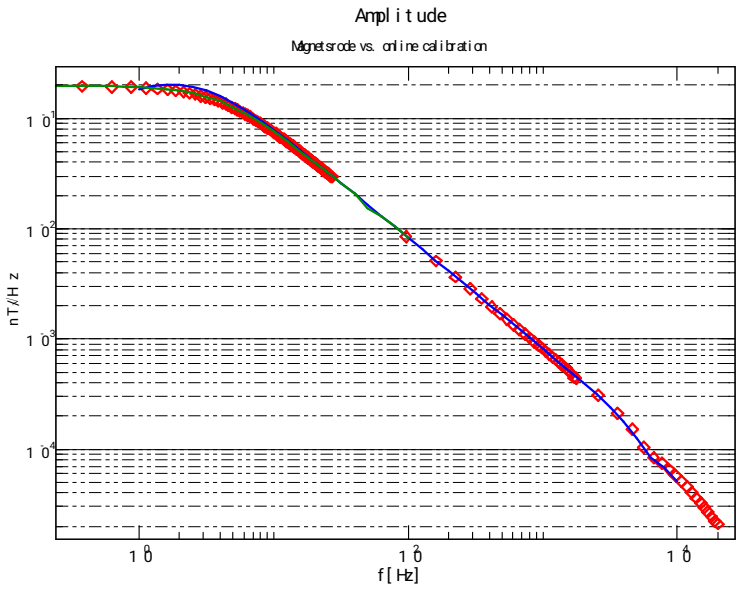


Figure 19.10: Online vs. MTX Calibration

Hence that if you use for example 32 Hz as calibration frequency the response will correspond to the odd harmonics: 96, 160, 224 Hz. To pick up these frequencies you have to choose the correct window length, which should be at minimum half of the sampling rate at the HF band and more for lower frequencies.

Make sure that you use the calibration frequency which was used during measurement – otherwise the result is unusable.

Board

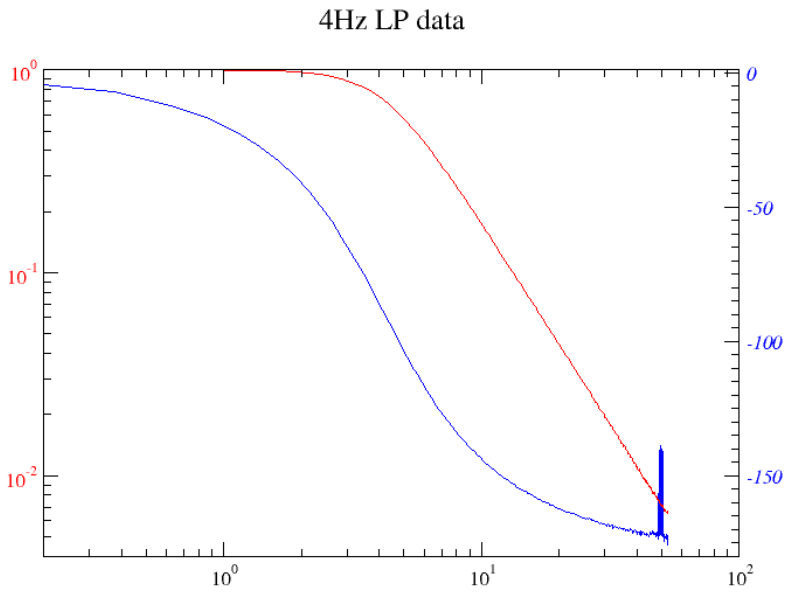
Two jobs have to be submitted of same sampling frequency and timing.

A) LF 4Hz

```
tsmp -wl 16384 -flist 8.lst -spectra_div -calib 57.2974693618
 -ul 6500 -ll 2 125_V01_C00_R000_TEx_BL_512H.ats
 125_V01_C01_R000_TEy_BL_512H.ats
```

The file 8.lst contains a list of frequencies such as .125 0.375 0.625 …. up to 219.375 in this demo. (1/8 Hz base frequency and their odd harmonics up to the frequency you want). The calib factor is here 180/π.

The result is here:



I have shown the "forbidden" frequencies where you can't correct even if you have a transfer function.

19.1.24 Overwrite

Activating this option will OVERWRITE the ats data file at the end of the time series section of tsmp. If you have subtracted or added a value or applied a convolution the new values will be written on disk.

Original data is lost now.

If you have used tsmp -start 128 -overwrite 036C02XA.ats tsmp overwrites 036C02XA.ats; but this file has now 128 samples less at the biginning and the start time has been shifted corresponding to the deleted samples.

Better make exercises with -out_dir option and write data into a different directory

If you do not have a copy of your original data ....

19.1.25 Crop

To cut or crop data segments out of one recording use:

```
tsmp -run 07 -start 0 -use 4880000 -noscale_e *.ats
```

or

```
tsmp -cut_int -start 1024 -stop 40098 *.ats
```

The second command leaves the header untouched.filw

19.1.26 Foreign Files and Transfer Functions

The standard header of ATS time series format does not contain the system type (data logger name). Especially when data from other loggers has been converted into ats file format this can become a problem.

Best way to solve this problem:

```
tsmp -corr -system_name linear -aduser 001 *.ats
```

for data without system transfer functions.

If you do not want to change the header use the options like -trf auto -nosystrf during processing.

If you have a system transfer function you must correct the header and put in a filename (8.3 format) into the header:

```
tsmp -corr -system_name linear -aduser 001 *.ats
```

19.1.27 Using the FIR Filter

Applying any filter will decrease the quality of the spectra or at least limit the range! For any MT application you should not apply any high, low, bandpass or bandstop (notch) filter.

For later usage of time series the filter length should be set to n*2s=n*samplerate+1; For LF1 that is a minimum of 8193.

Otherwise the time series has not a full second offsett and can not be used for remote referencing together with non-filtered timeseries anymore.

Filtering can take a tremendous amount of time because some $10^{9\text{-}12}$ operations have to be carried out.

Filtering is a typical application for a batch job.

All shown examples will use the option -out_dir ./filter to write the results into a directory filter which has been created by the user manually before. If this option is skipped the inputfile will be overwritten!!

To filter LF1 band with 4096 Hz sample rate with notch filter around 50 Hz use

```
tsmp -firfil 16385 55 45 0 -out_dir ./filter -overwrite
 055C03XB.ats
```

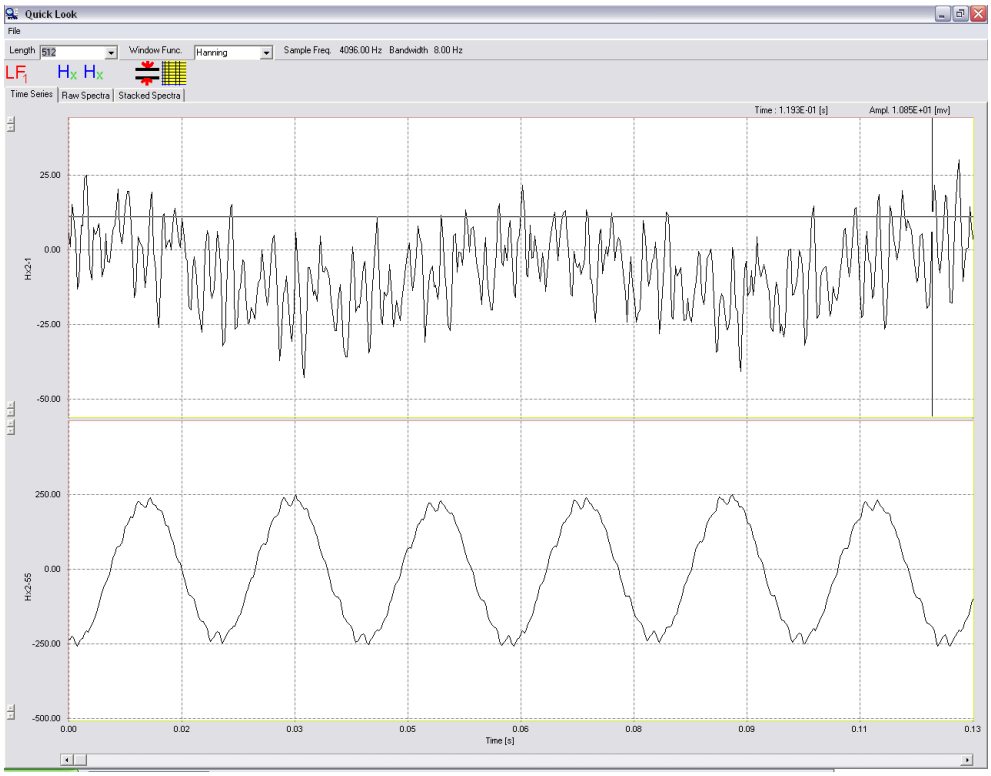


Figure 19.11: LF1 Time Series(Notch)

Hence that the notch filter option is characterized by length high low. Since the 50 Hz are strong we use a length of 16385 (that will cause 2 seconds offsett at start and stop time) and suppress frequencies between 55 and 45 Hz.

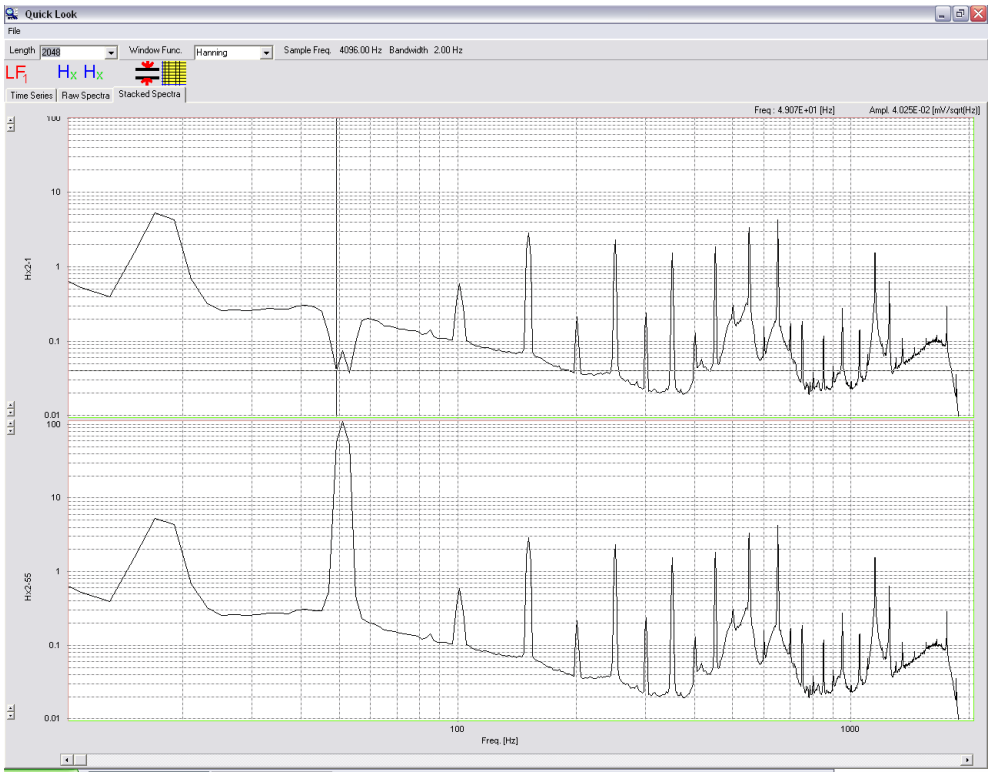


Figure 19.12: LF1 Spectra (Notch)

To use the filter as a bandpass use

```
tsmp -firfil 16385 1 1000 0 -out_dir ./filter -overwrite
 055C03XB.ats
```

The frequency range is now limited between 1 and 1000 Hz.

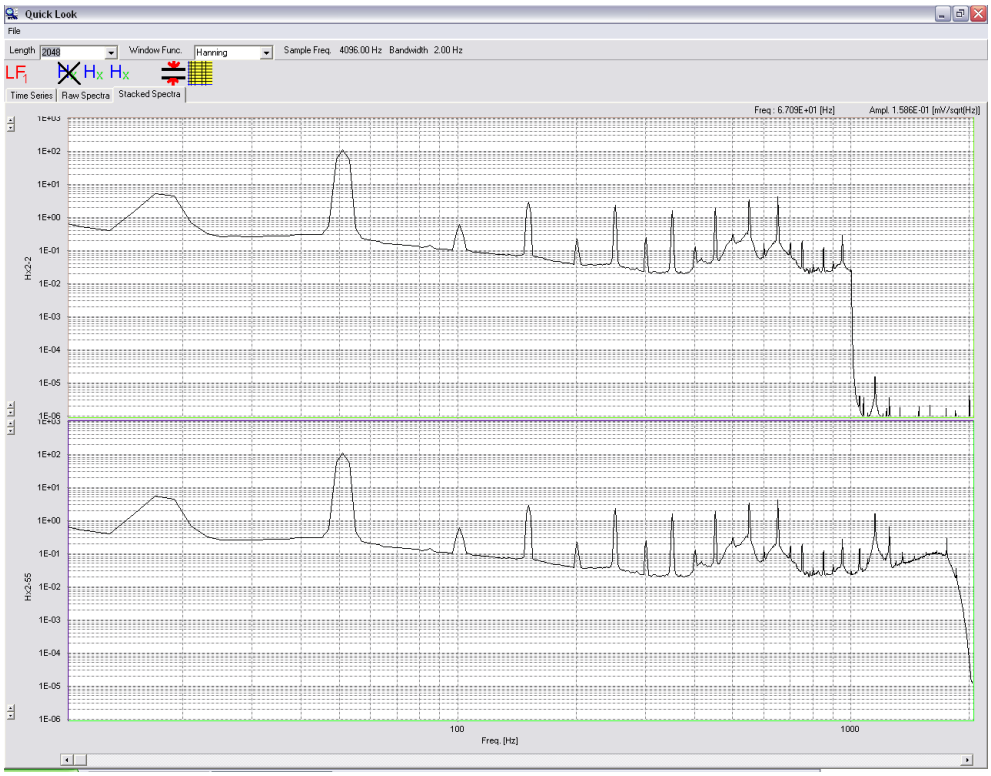


Figure 19.13: LF1 Spectra Low Pass Filtered

For Lower Bands like LF2 (C) and LF3 (D) the lower boundary becomes visible:

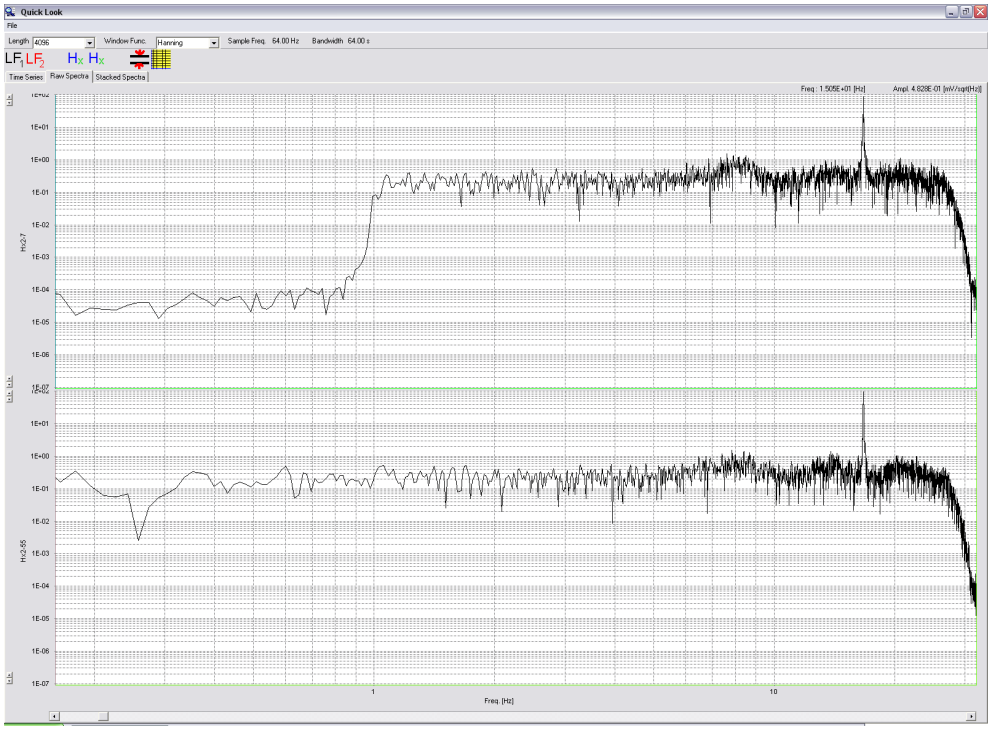


Figure 19.14: Spectra LF2 High Pass

Band	SampleFreq	FilterLength
B	4096	4097, 8193, 16385
C	64	1025, 8193, 16385
D	2	
Free	512	4097, 8193

Take odd filter length and make sure that the filter length corresponds to a even length of seconds (2,4,8...etc). As more points are used for the filter as better the filter quality will be.

19.1.28 Special Generated Signals

With the -gensign option you enable several signal and noise generators of tsmp.

To generate a simple sine wave do

```
tsmp  -gensign -gen_sin  8 2 0.0625 -samples 102400
 -channel_type Hx -samplefreq 64 -datetime 1000 -sensser 3
 -aduser 005 005C03XC.ats
```

Hence that you are responsible for the correct combination of filenames, sampling frequencies etc.

For more files use

```
tsmp -gensign -gen_sin  8 2 0.0625 -samples 102400
 -channel_type Hx -samplefreq 64 -datetime 1000 -sensser 3
 -aduser 005 005C03XC.ats 005D03DC.ats
```

etc.

E.g. you have to correct channels and other information inside the ats header if you want to continue with mapros:

```
find ./ -name $1C\*ats -exec /home/bfriedr/bin/tsmp -wrh
 -corr -aduser $2 -system_name ADU06 -channel_type Hx
 -sensor_type $4 -sensorcal 'SENSOR.CAL' -systemcal '
 ' {} \;
```

etc.

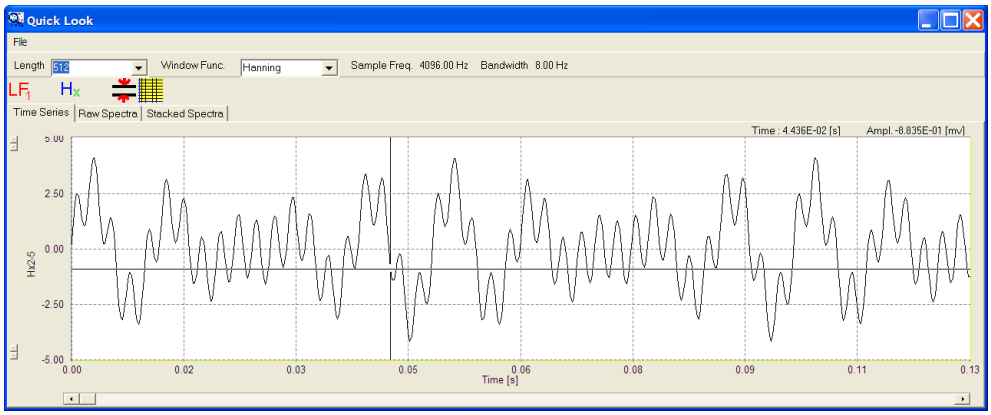


Figure 19.15: Artificial Sine Signal

Hence that you see the bit-noise from the fft calculation of your CPU.

Hence that the signal is normalized to an amplitude of 1 at a bandwidth of 1Hz – that is 4096 window length in the picture above (sample rate was

4096Hz).

Inside tsmp there are two functions included to visualize the transfer function of the MFS-06 and the FGS-01. When a MT processing is carried out with these special time series one should see the transfer function itself.

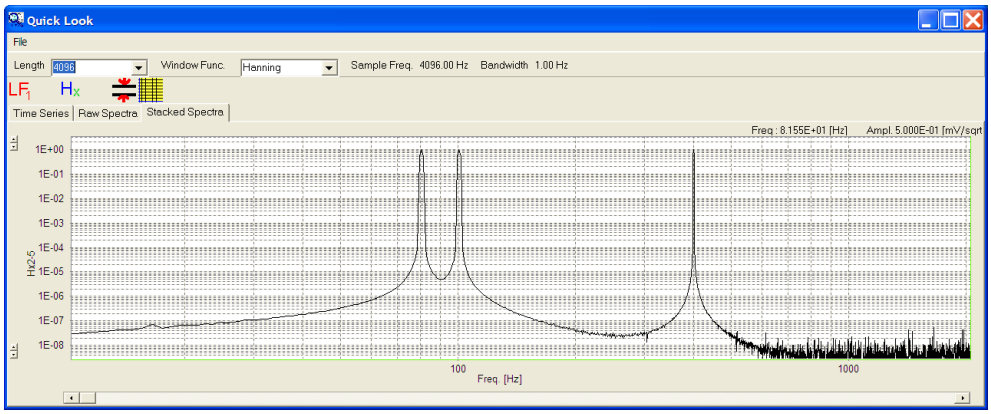


Figure 19.16: Spectra of Artificial Sine Signal

Use

```
tsmp -gensign  -samples 102400 -samplefreq 4096 -datetime
 1000 -sensser 1 -aduser 001 -gen_rand_gsl $1 $2 $3 $4 $5
 -mt -prz 0.1 $Ex $Ey $Hx $Hy $Hz
```

where

`$1 ... $5` are the noise mixer values and

`$Ex .. $Hz` the corresponding filenames;

in this examples use

`1 2 11.31 22.61` for coil noise and
`1 2 0.0014142136 0.0028284272` for fluxgate noise

and

`001A01AB.ats ... 001D01ZB.ats` as filenames.

The noise values are following Uli's calculations and normalized to 1Ωm @ 1000 Hz.

Tips and Tricks

Using mV/km for E and nT for H we obtain:

$$\rho_a = \frac{\mu_0}{\omega}\left|\frac{E}{B}\right|^2$$

using mV/km and nT

$$\rightarrow \frac{4\pi\, 10^{-7}\,\frac{Vs}{Am}}{2\,\frac{\pi}{s}}\left|\frac{10^{-6}\,\frac{V}{m}}{10^{-9}\,\frac{Vs}{m^2}}\right|^2 = 2\cdot 10^{-7}\,\frac{Vs^2}{Am}\left|10^3\,\frac{m}{s}\right|^2 = 0.2\,\frac{V}{A}m \tag{19.1.3}$$

therefore

$$\rho_a = \frac{0.2}{f}\frac{E^2}{H^2} \rightarrow 1 = \frac{0.2}{1000}\frac{E^2}{H^2} \rightarrow \frac{E}{H} = \sqrt{5000} \tag{19.1.4}$$

Due to the fact that the transfer function of the flugate is proportional to 0.1 mV/nT we obtain

$$E = 707.1\,H$$

or 1 1/707.1 → 1 0.0014142136 as mixer values (19.1.5)

For a induction coil MFS-06 we get 800 mV/nT @ 1 kHz we get

$$E = 70.71/800\,H \tag{19.1.6}$$

19.1.29 Simple Noise

You can generate simple random noise with a shell scripts:

```
#! /bin/bash
#./tsmp -gensign -gen_rand 181.019335984 -samplefreq 2048
 -samples 409600 -datetime 1000 -sensser 1 -aduser 001
 -sensor_type mfs06 -sensorcal '' 001C01XB.ats
rm *ampl*.dat
rm theo*.dat
rm raw*.dat
files='001C01X'
b2='C'
b3='D'
b4='G'
ampl='_ampl.dat'
./tsmp -gensign -gen_rand 32. -samplefreq 64 -samples 409600
 -datetime 1000 -sensser 1 -aduser 001 -sensor_type mfs06
 -sensorcal '' $files$b2.ats
./tsmp -gensign -gen_rand 5.65685424949 -samplefreq 2
 -samples 409600 -datetime 1000 -sensser 1 -aduser 001
 -sensor_type mfs06 -sensorcal '' $files$b3.ats
./tsmp -gensign -gen_rand 1 -samplefreq 0.0625 -samples
 409600 -datetime 1000 -sensser 1 -aduser 001 -sensor_type
 mfs06 -sensorcal '' $files$b4.ats
#
./tsmp -wl 1024  -ll 6 -ul 20 001C01X?.ats
cp $files$b2$ampl raw$b2.dat
cp $files$b3$ampl raw$b3.dat
cp $files$b4$ampl raw$b4.dat
#
./tsmp -wl 1024 -trf theo -ll 6 -ul 20 001C01X?.ats
cp $files$b2$ampl theo$b2.dat
```

```
cp $files$b3$ampl theo$b3.dat
cp $files$b4$ampl theo$b4.dat
```

Hence that I scale the spectra from band to band by √Hz and change the sampling frequency by 32. This will generate bands which will appear constant (bandwidth is changing and amplitude is changing).

Figure 19.17 Noise and calibration

19.1.30 Down Sampling

Mainly FIR filter will reduce the sampling rate by even factors like 2x 4x and so on.

With a spline-resample you can generate "almost any" sampling frequency; even it is possible to generate 64Hz from 32Hz sampling – in the sense that there is a sample every 1/64s.

In my example I will generate a 50Hz sampling which often used in seismics – but almost unusable in EM.

```
tsmp -resample 50 -noscale_e -run 32 file.ats
```

Generates a file with 50Hz sampling rate. In my example I use 128 Hz and 64 Hz as souces.

To plot files with different sampling rates into one plot you must generate a plot including a time column. E.G. you use the ISO date format because it can contain values less that a second – and that is needed here.

```
tsmp -start 0 -use 1024 -tc -abst 0 -dateformat ISO -ascii
 50_128.ats
tsmp -start 0 -use 1024 -tc -abst 0 -dateformat ISO -ascii
 -iso_s -0.14 50_128.ats
```

As you can see I am facing the problem that the spline cause a small delay (run time length of a filter) which I correct with -iso-s.

Figure 19.18 Resample: artefacts at high end

Figure 19.18 shows that the spectra becomes acceptable at less than sample_f / 4. Where as the 50 Hz generated from 128 Hz is good up th the 16 Hz peak.

Figure 19.19 Resample: usable overlapping at the lower end

Coming down to lower frequencies the result is surprisingly good.

In the time domain we estimate that as more high frequencies are existing as more that chance will exist to miss the peaks. In general it is a good idea to to something like resample 256 Hz to 200 Hz and than filter by 4x.

Figure 19.20 Resample: time domain comparison

19.1.31 ISO Time and cat

You can concatenate files with

```
tsmp -cat 03 olderfile.ats newer_file.ats
```

A new file with run number 03 or 003 will be created from the first filename.

To extract data by time – and not by sample number – use:

```
tsmp -ascii -abst 0 -tc -dateformat ISO -isostart 2009-08-
 12T20:00:00 -isostop 2009-08-13T02:00:00
 067_V01_C00_R003_TEx_BL_32H.ats
```

Hence that presently this option performs no plausibility check.

20 Non-MT Measurements

The so-called chopper amplifier with a frequency of 2048 Hz is the basic separator between high- and low frequency measurements.

If the chopper is switched on you can sample with 4096 Hz and the first low noise data of your spectra will be obtained at 1024 Hz or 512 Hz (with ADU06).

By subsequent filtering one can obtain all lower frequencies from this recording. However: The ADU07 has in addition a 4 Hz low-pass filter which can be switched on. Here the 50/60 Hz and 16 2/3 Hz can be suppressed and the gain amplification can be set higher; the results in the MT dead-band (1s - 10 s) shall become better now.

If the chopper is switched of you are limited in recording lower frequencies. At frequencies around 30 Hz the data becomes noisy; around 100 Hz and higher the data becomes better than the data with chopper on.

The MFS06 has a little one-pole low-pass at 8192 Hz; for recordings at higher frequencies the MFS07 can be used up to 50 kHz.

The MFS06 records down to 4096 s and – with good magnetic field activity down to 10,000 s; the MFS07 is limited to 1000 s; for frequencies below 1 Hz the MFS06 shall be preferred. Hence that a coil does not suddenly stop to work: the noise level slightly rises more and more.....

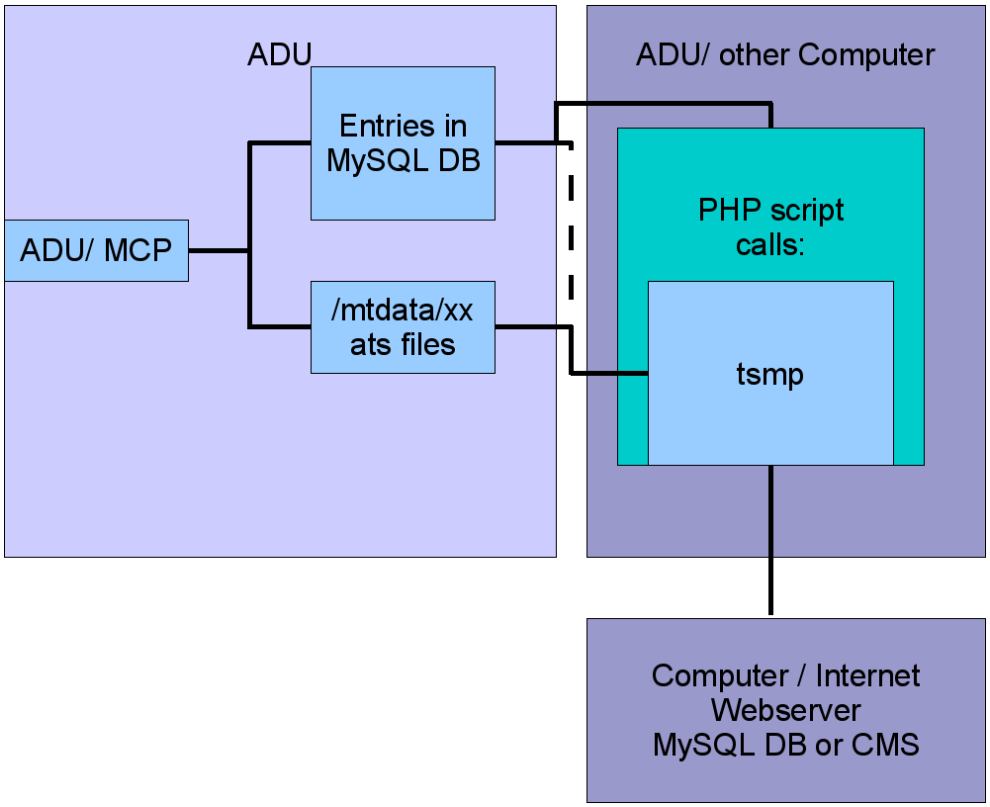
For ultra high frequencies above 1 kHz the SHFT02 can be used -up to 400 kHz.

21 Database Processing

(programming in progress...)

tsmp is able to write the processing results into a MySQL database.

The flow chart illustrates the idea:



A PHP script checks the status of the ADU. If data is available tsmp is called. The script can run on the ADU or on a different computer and connect via port 3306 to th ADU's MySQL database.

The same applies to the tsmp software. It can run on the ADU or read the ADU's data from the samba share /mtdata. tsmp can publish the data again on a MySQL database – either again on the ADU or on a computer connected to the Internet.

At the present the programming focus lies on the installation that the php script and tsmp are installed on a remote computer.

```
-- Database: `tsdata`
--
CREATE DATABASE `tsdata` DEFAULT CHARACTER SET latin2 COLLATE
latin2_general_ci;
USE `tsdata`;
-- Table structure for table `ex`
CREATE TABLE `ex` (
  `ex` double NOT NULL default '0'
) ENGINE=MyISAM DEFAULT CHARSET=latin2;
-- Table structure for table `ey`
CREATE TABLE `ey` (
  `ey` double NOT NULL default '0'
) ENGINE=MyISAM DEFAULT CHARSET=latin2;
-- Table structure for table `hx`
CREATE TABLE `hx` (
  `hx` double NOT NULL default '0'
) ENGINE=MyISAM DEFAULT CHARSET=latin2;
-- Table structure for table `hy`
CREATE TABLE `hy` (
  `hy` double NOT NULL default '0'
) ENGINE=MyISAM DEFAULT CHARSET=latin2;
-- Table structure for table `hz`
CREATE TABLE `hz` (
  `hz` double NOT NULL default '0'
) ENGINE=MyISAM DEFAULT CHARSET=latin2;
-- Table structure for table `iso8601`
CREATE TABLE `iso8601` (
  `iso8601` char(20) NOT NULL default '0'
) ENGINE=MyISAM DEFAULT CHARSET=latin2;
-- Table structure for table `unix`
CREATE TABLE `unix` (
  `unix` bigint(20) NOT NULL default '0'
) ENGINE=MyISAM DEFAULT CHARSET=latin2;
```

22 Programming Notes

Most time series data contains the UNIX time stamp – a long integer, seconds since 1970. 977645700 corresponds with 24. Dec. 2000 9:15 in Germany and 8:15 GMT time.

The UTC time (satellite time differs; actually in the year 2003 the offsett is 13s).

ATS is using GMT time. Reading in a file uses relative time – starting with 0; hence that most systems start on a full second. The DateTime variable inside tsmp will be set – read from the header and modified to the corresponding start position. With 512 Hz sample rate a -start 1024 read all data from 0 to 1023 and starts at position 1024; DateTime is increased +2 sec. If you use -start 121 the DateTime will contain a round up error.

Using -abst uses the full time information and the output file does not start with 2 but with 977645702 if -tc was switched on. All start and and stop/use points are saved internally with skip_samples_read and n_samples and in main the size can be asked with atd.nsamples().

23 Units

Symbol	Unit	Meaning
E	V/m	electric field
D	As/m^2	electric flux
B	$nT = 10^{-9}\ Vs/m^2$	magnetic flux
H	A/m	magnetic field
ε_0	$8.8542 \cdot 10^{-12}\ As/Vm$	electric permittivity
μ_0	$4\pi \cdot 10^{-7}\ Vs/Am$	magnetic permeability
ρ	Ωm	resistivity
σ	S/m	conductivity
τ	S	conductance
S	S	cumulative conductance
Z_{ij}	m/s	magnetotelluric impedance
C_{ij}	m	complex penetration depth
γ	$10^{-9}\ T = 1\mathrm{nT}$	Gauss, Gamma, non SI unit

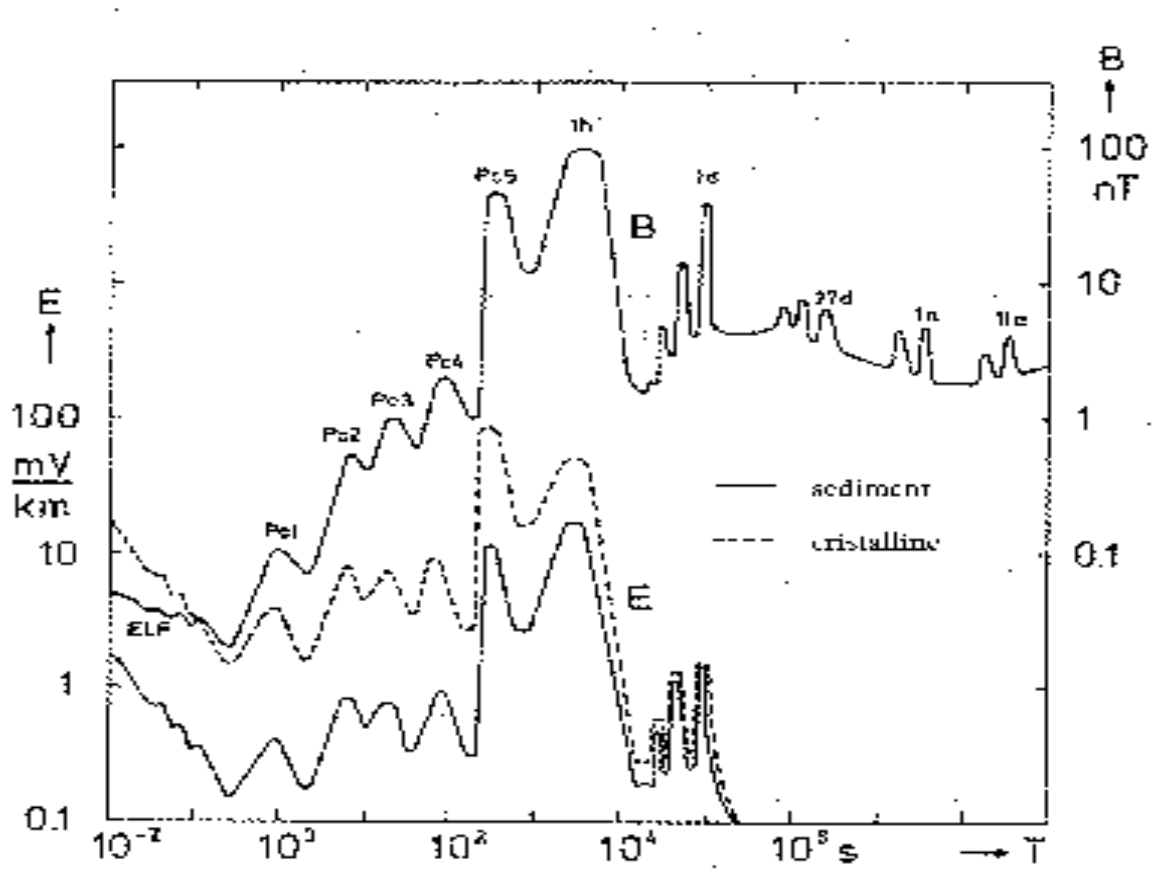
Symbols used in SI system

	Abbr.	**Period/Freq.**	**B [nT]**	**Source location**
Irregular variations				
high frequent emission	VLF	10 µs-1 ms	<0.1	magnetosphere,
and whistler spherics		3 kHz		lighting
low frequent emissions	ELF	3 kHz-1 Hz	<<0.1	atmosphere,
and atmospherics	ULF	>1 s		lighting
SCHUMANN resonance		5/39 s ...	<0.1	wave guide earth-ionosphere
continuous pulsations	pc1-5	0.2-600 s		magnetosphere
irregular pulsations	pi1-2	1-150 s		-``-
solar flare effects	sfe	10 min	10	ionosphere, D&E layer
sudden storm commencement	ssc	2-5 min	10-100	magnetosphere
polar sub storm	bay	30-120 min	20-100	ionosphere of the
without recovery phase				auroral zone
polar electrojet	PEJ	30-120 min	20-100	magnetosphere
polar magnetic storm	DP	5-120 min	20-100	-``-
with recovery phase	(PEJ)	5-120 min	20-100	-``-
recovery phase of	Dst	3 d	100	ring current
polar magnetic storms	DS	1 d	100	ring current
equatorial electrojet	EEJ	all freqs.		ionosphere at equator
Regular variations				
solar daily	S	1 d	20	E-layer in

Units

	Abbr.	Period/Freq.	B [nT]	Source location
variation				ionosphere
- on quiet days	Sq	1 d	20	-``-
lunar daily variation	L	1 d	2	E-layer in ionosphere
semiannual variation		6 months	5	ring current
annual variation		1 a	5	ionosphere
sunspot cycle variation		11 a	20	ring current

External sources of natural excitation (Schmucker, lecture notes winter term 92/93)<tab: source_schmucker>



Chapter 23

Spectra of magnetic field and the responding electric field in ground
(Lecture notes of Weidelt)

24 Remarks Using the FFTW

The FFTW tries to be fast and efficient – but the usage is difficult.

We do a FFT with 7 real points with indices 0,1,2,3,4,5,6; the result is 0, 1,2,3,3,2,1 where 0 is the DC component, the 3 real and 3 imaginary indices.

As we set up the resulting vector as N/2+1the result has 4 complex numbers.

Now we take – that is mostly the case – an even amount of input – 8 numbers with indices 0,1,2,3,4,5,6,7; the resulting vector N/2+1has now 5 complex numbers (that is not really a contrast to the rule that if we put 1024 values into a FFT we get 512 complex values back). The result is 0,1,2,3,4,3,2,1 – 8 values to be stored in 5!! complex numbers: 0 as DC component containing a real value only, then 1,2,3 business as usual, and the 5^{th} complex contains the Nyquist frequency with index 4; also a real number is stored inside the complex result.

Hence that we need to sort out our complex array if we want to transform back – complex[0,...,5] have to be put back as in[0] = real(c[0]), in[1] = real(c[1])...

in[4] = real(c[4]), in[5] = imag(c[3]) and so on.

25 Copyright

bernhard.friedrichs@coopertools.com