Wavelet Analysis of Transients and Unmodeled GW signals

Presented by S.Klimenko

University of Florida

- Outline
  - What are wavelets?
  - Detection of transients & unmodeled sources with wavelets.
  - Wavelet Analysis Tool
  - Current status
  - Plans & Conclusion

LIGO Data Analysis at UF

- LSC White Paper - plan of work on
  - detector characterization
  - development of detection algorithms
  - provision of reduced data sets
- One of the UF group commitments is:
  - Development of Wavelet Analysis Tool for
    - data compression/reduction
    - transient signal characterization
    - unmodeled GW sources detection
  - WAT will be part of LIGO/LSC Algorithm Library
- “Wavelet” people at UF:

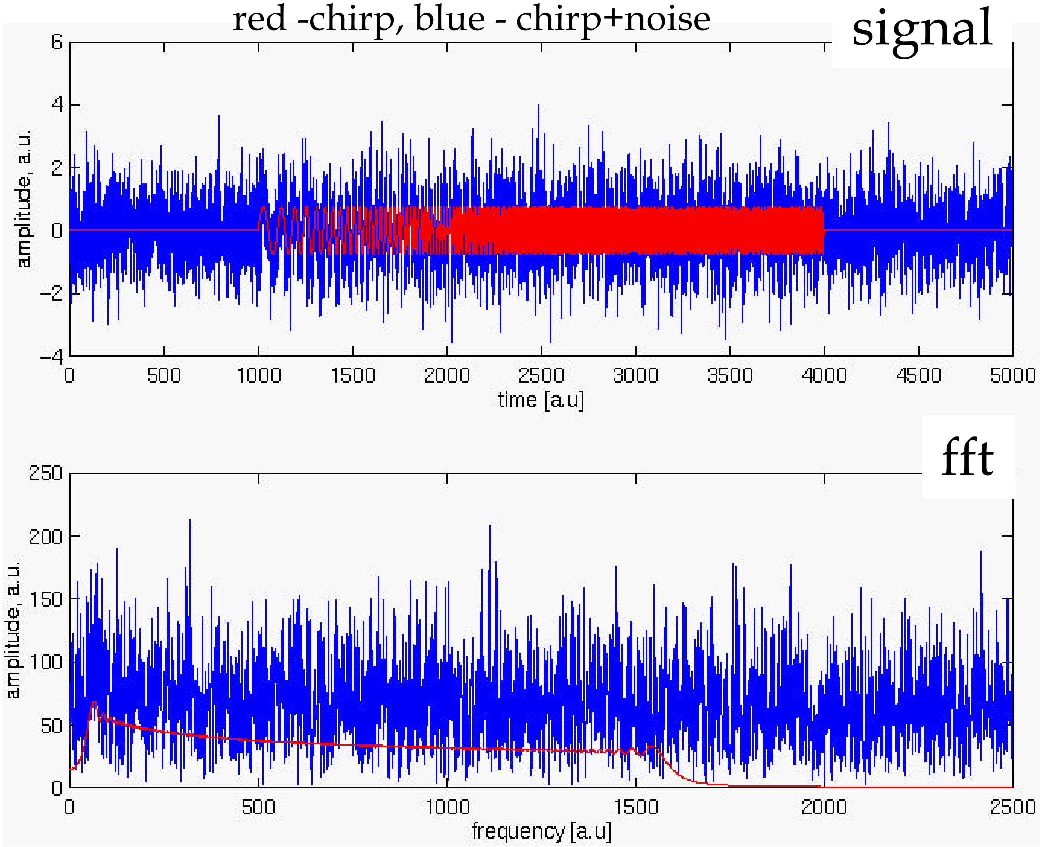
  S.Klimenko, G.Mitselmakher, A.Sazonov, B.Whiting

Wavelets

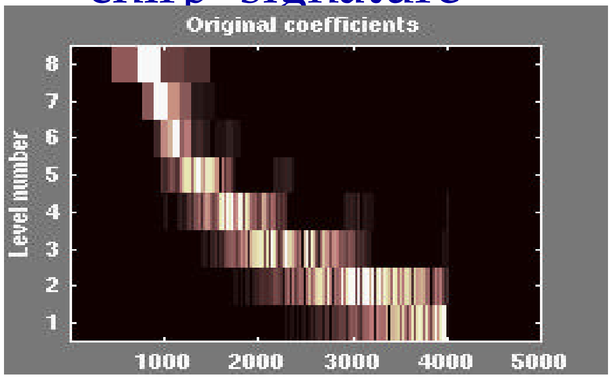
- What are wavelets?
  - set of basis functions $\Psi_{jk} = a^{j/2}\,\Psi_0(a^j x-k)$; Ψ_0 -mother wavelet
  - used in a way similar to Fourier Transform:
    $w_{jk} = \Sigma_i\, f(x_i)\, \Psi_{jk}(x_i)$ - digital wavelet transformation of $f\,(x_i)$
  - local in time & frequency domains (in contrast to Fourier Transform)
    real signals are finite in time!
- Why wavelets?
  - wavelets are convenient for pattern recognition
    - widely used in image and signal processing.
    - can be used for GW signal and non-gaussian noise identification.
  - allow simple description of signal with minimal number of waveforms
  - mathematics of wavelets is well developed, algorithms are flexible and fast.
- Very promising technique for LIGO data analysis

Example of wavelet use (db6 wavelet)

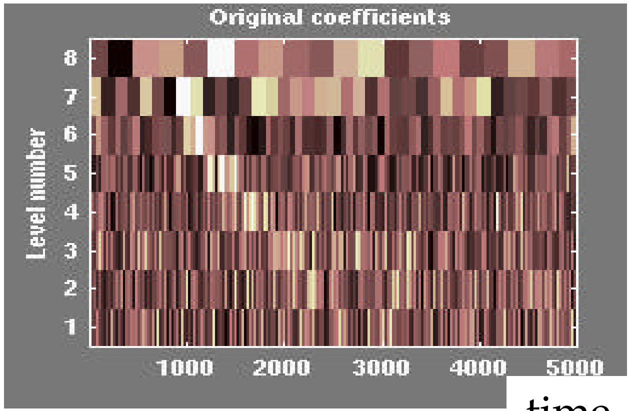
- Chirp u + Gaussian noise n: (SNR= <u^2>/<n^2> = 0.25)



chirp 'signature'



wavelet domain



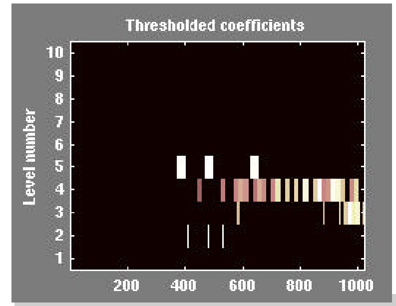
time

- GW signal or transient can be identified by it's signature

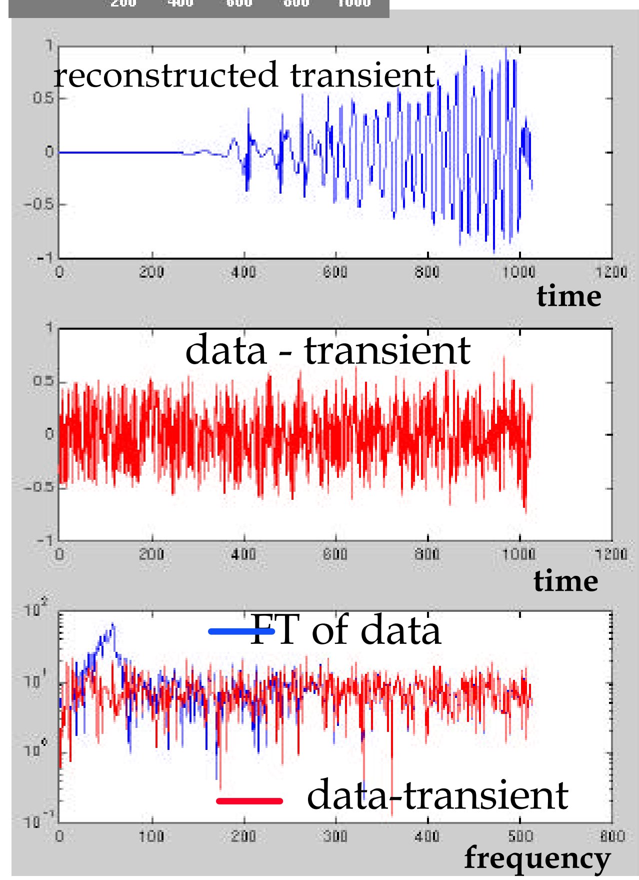
S.Klimenko

Transient analysis

- Transient is characterized by it's signature in wavelet domain
- transient identification:
  identify transients by statistical analysis of data in wavelet domain. All types of wavelets can be used. Optimal wavelets retain maximum of transient energy with minimal number of coefficients. Transient's signature may be unknown in advance
- transient reconstruction:
  If identify transient with orthogonal (bi-orthogonal) wavelet, it can be rebuilt in time domain using reconstructed signature and subtracted from the original data.



reconstructed transient's signature



S.Klimenko

Filtering with Wavelets

- Optimal (Wiener) filter F_{jk} in wavelet domain

$$S_n(s'(nD) - s(nD))^2 = S_{jk}(u'_{jk} - u_{jk})^2$$

$$u'_{jk} = w_{jk}\, F_{jk}\,; \qquad F_{jk} = u^2_{jk}/(u^2_{jk} + n^2_{jk})$$

  s,u - uncorrupted signal, s',u' - filtered signal, n - noise, w - data

  s',s - time domain, u',u, n,w - wavelet domain

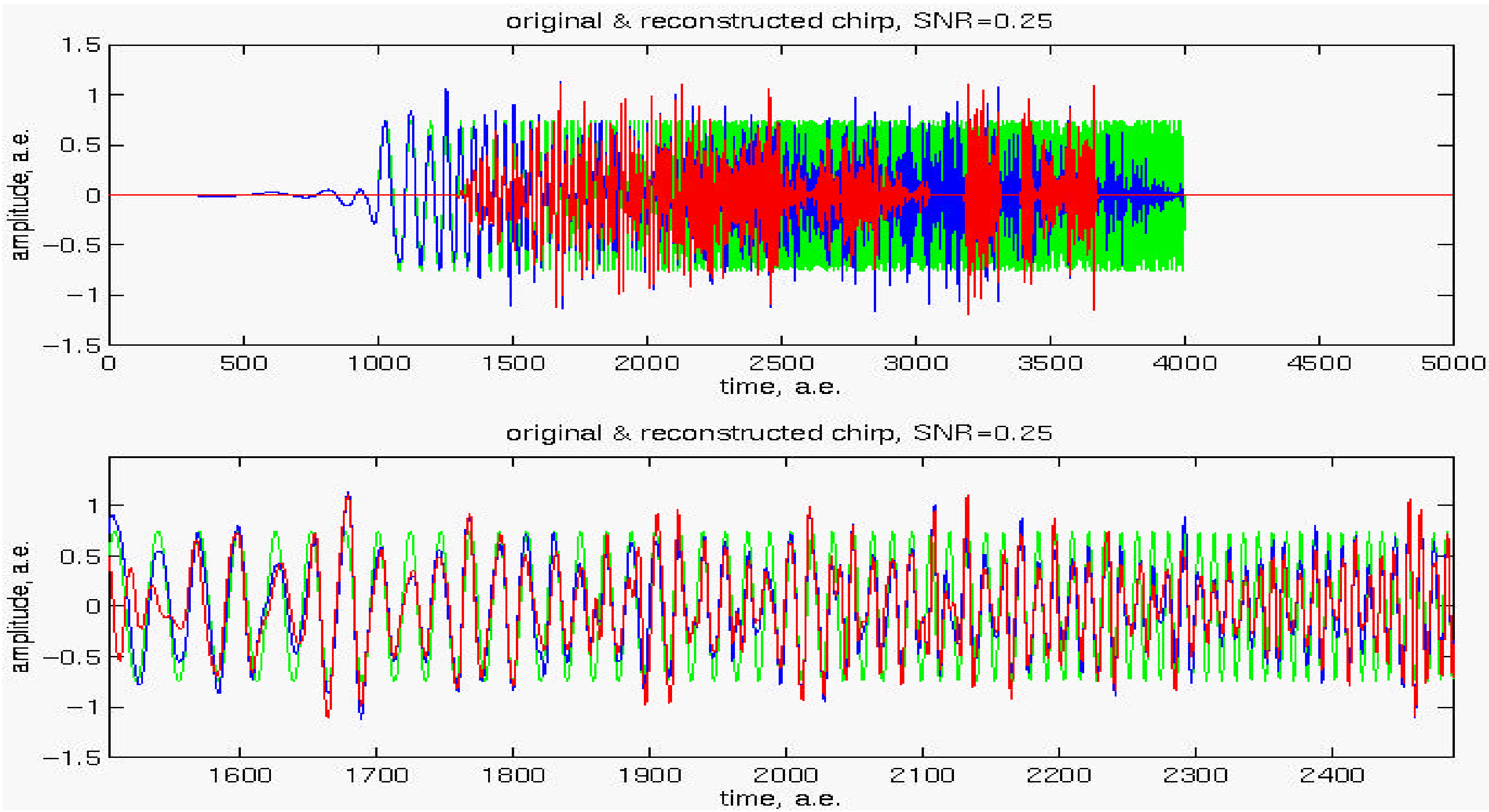
  - F_{jk} requires detail information about signal u and noise n.

- Practical filter
  - limited information about signal and noise
  - signal signature may be unknown (e.g. unmodeled GW signals)
  - example: wavelet threshold filter (WTF)
    - set threshold on w_{jk}^2 / s^2; s^2 - noise variance.
    - n_{jk}^2 / s^2 - has χ^2 distribution in case of Gaussian noise.
    - no *a priory* information about signal is used

S.Klimenko

Example of chirp reconstruction

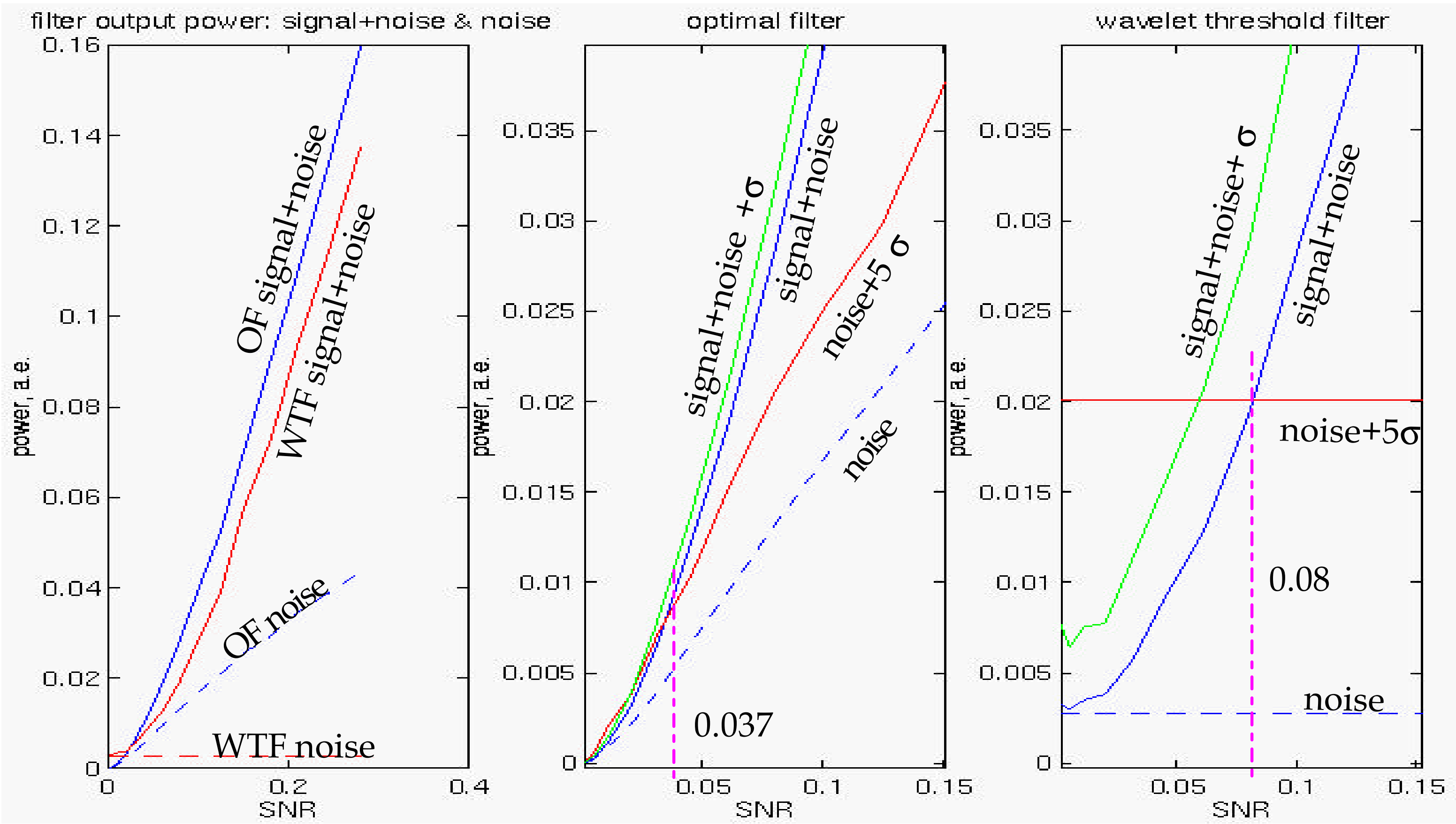
- reconstructed chirp signal: blue- optimal filter (OF), red - wavelet threshold filter (WTF), green -original chirp. SNR=0.25



S.Klimenko

Comparison of optimal & threshold filters

- detectable SNR: 0.035 (db6 OF) vs 0.08 (db6 WTF) (preliminary result)
- no information about signal is used by WTF!



S.Klimenko

Unmodeled GW signals detection

- UGW signal detection is "*looking for things that aren't noise*" (LSC White Paper, Unmodeled Sources)
  - ➢ detection of non-Gaussian (nG) noise
  - ➢ sorting out nG noise using environmental monitor data
  - ➢ analysis of residual nG noise trying to find "*things that aren't noise*"
- UGW signals selection with wavelets
  - ➢ We can consider a UGW signal as a "transient" (or nG noise) with unknown signature.
  - ➢ Wavelet algorithms for transients detection can be used to select UGW signals. More specific algorithms can be developed if needed.
  - ➢ Bank of unidentified "transients" (residual nG-noise) is a good data sample to search for UGW sources.
  - ➢ Analysis of signals from multiple detectors: looking for correlation of wavelet signatures.

Comparison with other UGW methods

- Power monitoring:
  - ➢ wavelets work in a similar way measuring excess of nG noise over G noise with rms σ_j:

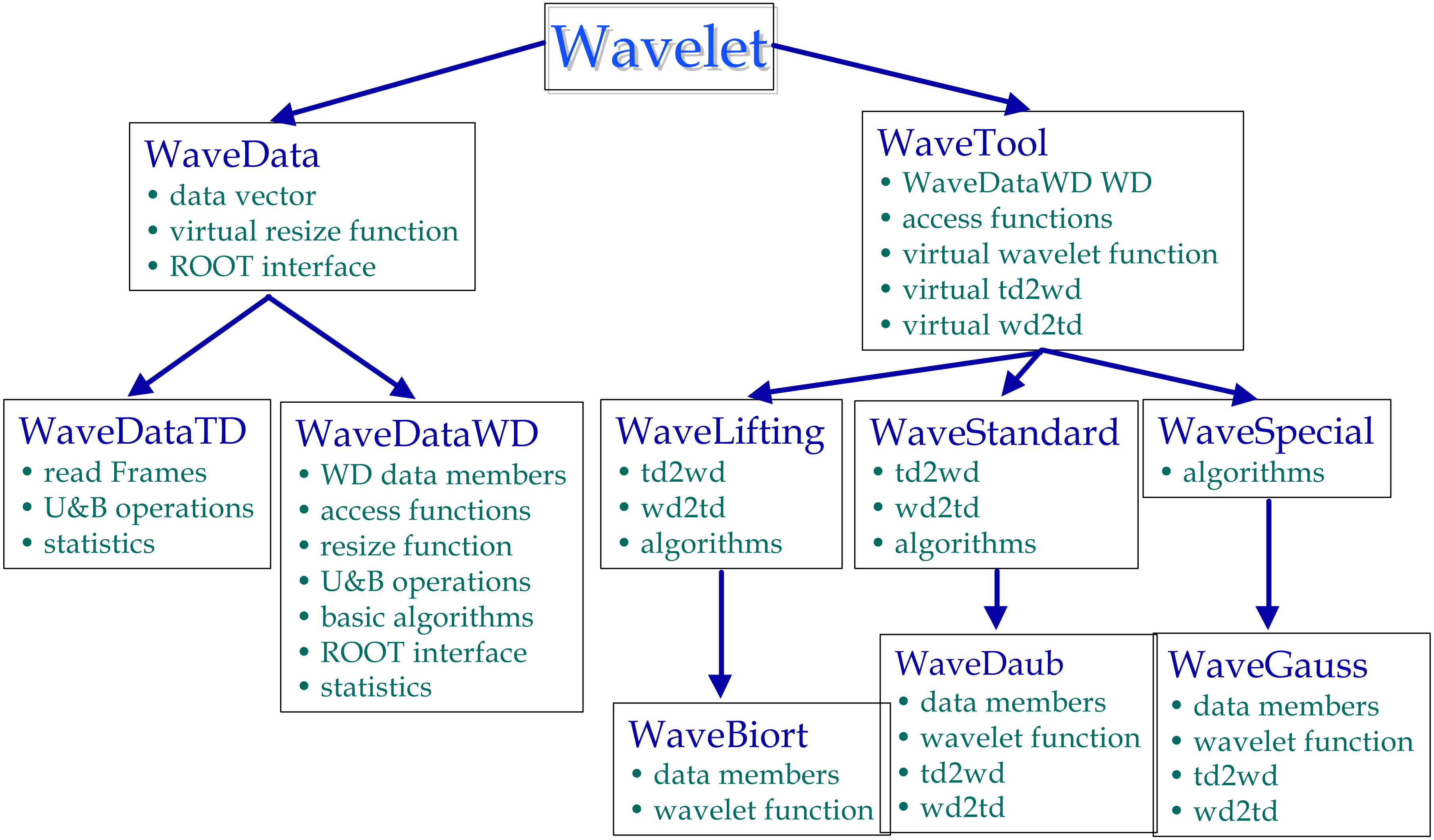
$$\chi^2 = -2 \ln L = \sum_{jk} w_{jk}^2 / \sigma_j^2$$

  - ➢ different types of wavelets can be used to get better SNR.
- Time-frequency method:
  - ➢ wavelets give time-frequency representation of data
  - ➢ signal signature recognition is used to identify events
  - ➢ different wavelets can be used for different signals to get better SNR
  - ➢ wavelet algorithms can be very fast
- Pulse matching:
  - ➢ wavelet is a "bank of profiles". Using Gaussian wavelets we can get a bank of Gaussian profiles.
  - ➢ Wavelets offer general pulse matching technique that can be optimized to search for different types of the UGW sources.
- **Wavelets are intrinsically different or supplement existing methods**

Wavelet Analysis Tool

- Toolbox to construct data triggers/filters and process LIGO data.
- WAT components:
  - ➢ wavelet class library (C++):
    - ➢ wavelet domain data structure and functions
    - ➢ wavelet transformations (Doubechie's, Mayer's, Fast Wavelet Transforms)
  - ➢ interfaces to LDAS (Frame format) and GUI (ROOT)
  - ➢ build in set of wavelet algorithms for data analysis
- Why we need WAT?
  - ➢ can be used to process large amount of data
  - ➢ provides class library for development of new wavelet algorithms
  - ➢ will agree with LLAL requirements

Wavelet Class Library



S.Klimenko

Current Status

- Wavelet Analysis Tool development
  - ➢WAT structure is determined
  - ➢can read Frame data (need switch from Fcl to FrameCpp)
  - ➢Fast Wavelet Transforms (lifting wavelets) (implemented)
  - ➢Gaussian wavelets (in progress)
  - ➢data reduction algorithms (in progress)
  - ➢transients identification (investigating)
  - ➢interface to ROOT (investigating)

Plans

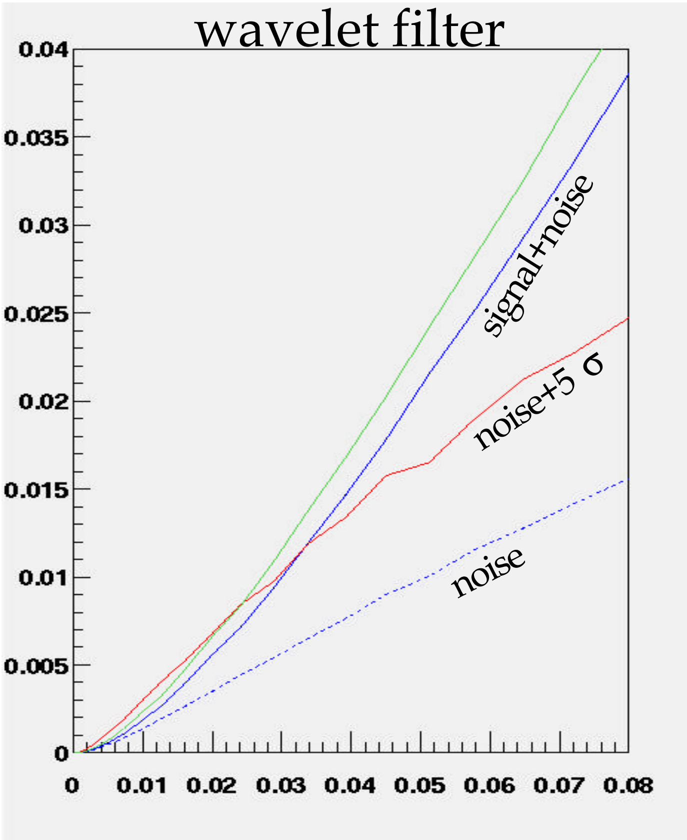
- Short term plan (Aug 2000):
  - develop first version of WAT
    - wavelet class library
    - interfaces to LIGO data and ROOT
    - set of Fast Wavelet Transforms and Gaussian wavelets
  - preliminary algorithms for transient analysis and data reduction
  - consider specific wavelet software for UGW analysis
- Long term plans (2000-2002)
  - 2000 - initial WAT
    - develop simple and fast wavelets & wavelet algorithms to process large amount of data at earlier stage of analysis, including UGW signals detection
  - 2001 & 2002 - final WAT
    - develop more sophisticated and hence less time efficient wavelets and wavelet packets for final stage of data analysis.
    - use wavelets to analyze LIGO data

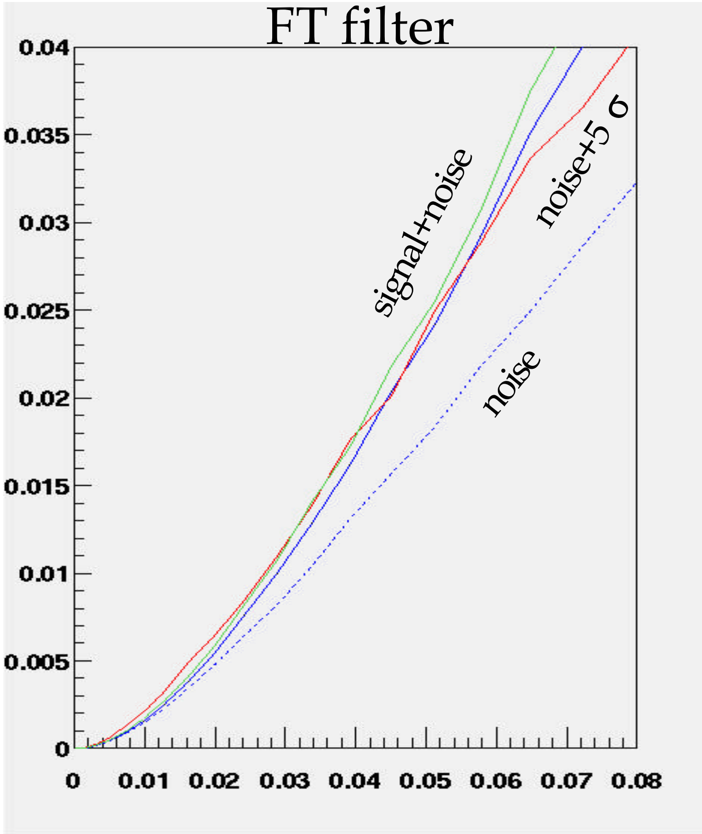
Conclusion

- Wavelets can be used to construct data triggers and filters to sort out GW and noise pulses and produce reduced data sets.
- Flexible Wavelet Analysis Tool is needed
  - Different wavelets and algorithms need to be used for different tasks
  - Different wavelets and algorithms will be used at initial and final stages of analysis.
- UF group is working on wavelet algorithms and wavelet software development

Optimal filters in wavelet and frequency domains

- Optimal 3rd order lifting wavelet filter
- Optimal filter in frequency domain





Wavelet Analysis & Line Removal

Presented by S.Klimenko

University of Florida

- Outline
  - Wavelets
    - Wavelet Analysis Tool
    - Current status
    - Plans & Conclusion
  - Line removal
    - Algorithm
    - Results
    - Conclusion

LIGO Data Analysis at UF

- LSC White Paper - plan of work on
  - detector characterization
  - development of detection algorithms
  - provision of reduced data sets
- One of the UF group commitments is:
  - Development of Wavelet Analysis Tool for
    - data compression/reduction
    - transient signal characterization
    - un-modeled GW sources detection
  - WAT will be part of LIGO/LSC Algorithm Library
- "Wavelet" people at UF:

  S.Klimenko, G.Mitselmakher, A.Sazonov, B.Whiting

Current Status

- Wavelet Analysis Tool development
  - ➢WAT structure is determined
  - ➢can read Frame data (need switch from Fcl to FrameCpp)
  - ➢Fast Wavelet Transforms (lifting wavelets) (implemented)
  - ➢Gaussian wavelets (in progress)
  - ➢data reduction algorithms (in progress)
  - ➢transients identification (investigating)
  - ➢interface to ROOT (investigating)

Plans

- Short term plan (Aug 2000):
  - develop first version of WAT
    - wavelet class library
    - interfaces to LIGO data and ROOT
    - set of Fast Wavelet Transforms and Gaussian wavelets
  - preliminary algorithms for transient analysis and data reduction
  - consider specific wavelet software for UGW analysis
- Long term plans (2000-2002)
  - 2000 - initial WAT
    - develop simple and fast wavelets & wavelet algorithms to process large amount of data at earlier stage of analysis, including UGW signals detection
  - 2001 & 2002 - final WAT
    - develop more sophisticated and hence less time efficient wavelets and wavelet packets for final stage of data analysis.
    - use wavelets to analyze LIGO data

Conclusion

- Wavelets can be used to construct data triggers and filters to sort out GW and noise pulses and produce reduced data sets.
- Flexible Wavelet Analysis Tool is needed
  - Different wavelets and algorithms need to be used for different tasks
  - Different wavelets and algorithms will be used at initial and final stages of analysis.
- UF group is working on wavelet algorithms and wavelet software development

Coherent Lines Removal

- Wavelets algorithms can be used for time-frequency analysis of transients and/or short bursts of GW.
- Strong line interference produces a significant non-Gaussian noise masking other non-Gaussian components.
- Effective, simple and fast line removal algorithm is necessary for wavelet analysis.

Line Removal Algorithm

- DFT of data of N samples:
  - basis of orthogonal Fourier functions:
    $F_k(n)=e^{-2\pi i\, n\, k/N}$, $k,n = 0,..,N-1;$ $\delta_{ij}=\Sigma_n F_i(n) F_j(n)$
  - sampled harmonic signal: $L(n)=a\, e^{-2\pi i\, n\, f/fo}$,
    - f - harmonic signal frequency
    - f_o - sampling rate
  - if $f/f_o=k/N$, $L(n) = a_k F_k(n)$ - one of the basis Fourier functions.
- Removing of line and its harmonics $\{L_k(n)\}$.
  - resample data with sampling rate f_S: $f_S/f = int(f_o/f)+1$
  - select data sample length: $N = k\, f_S/f$, $k=1,2\ldots$
  - extract interference signal: $I(n) = \Sigma_k L_k(n)= \Sigma_k a_k F_k(n)$
  - re-sample $I(n)$ back & subtract from original data

Data re-sampling

- Sample rate converting
  - reconstruction: $s(nD) \rightarrow s(t)$; $D=1/f_0$
    ST: $s(t)$ perfectly represents frequencies that are less then $f_0/2$
  - sample data at new sampling rate: $s(t) \rightarrow s'(nD')$; $D'=1/f_s$
- Data interpolation filter
  - wavelets
  - other interpolating techniques
  - currently nth order polynomial interpolation filter is used (lifting wavelets use the same filter)
- Up-sampled ($f_s > f_0$) data used to find interference due to harmonic lines

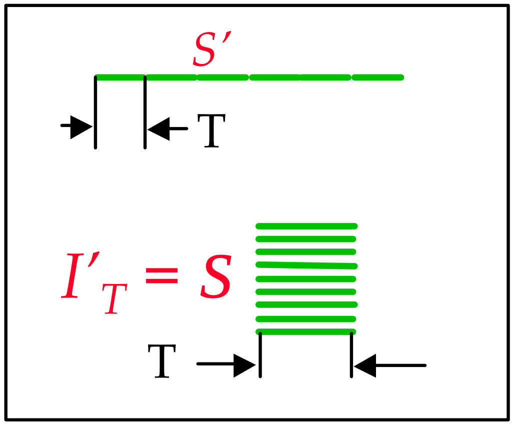
Line Interference Signal

- line extraction

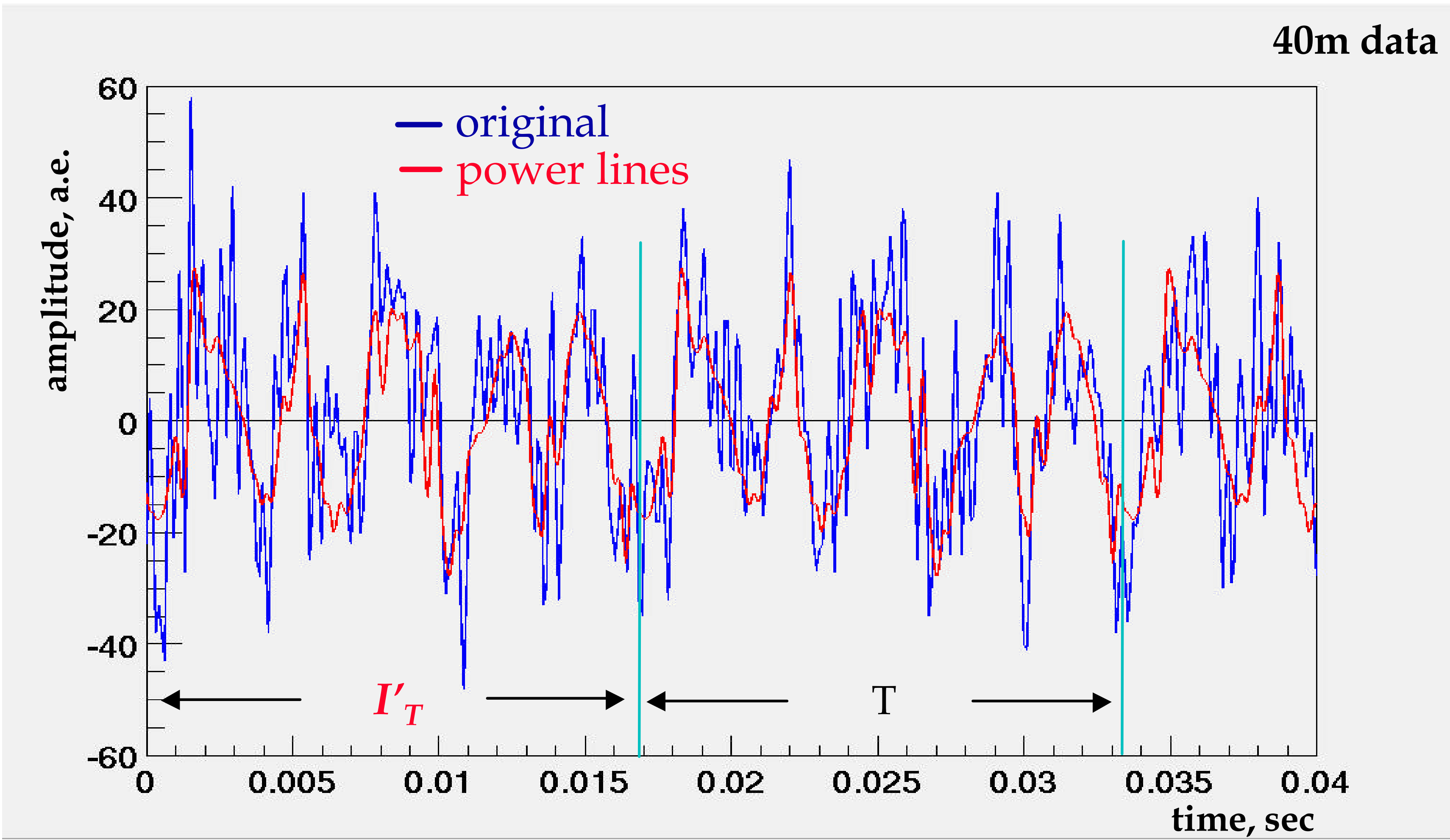
$$I'(n) = S_k\, a_k F_k^{*}(n)\ f_k$$

  - ➢ $a_k = \Sigma_n\, s'(n)\, F_k(n)$ - Fourier coefficient for kth harmonic
  - ➢ f_k - optimal filter; $f_k = 1$, if neglect noise for kth harmonic

- fast line extraction in T domain $(f_k = 1)$:
  - ➢ T - period of fundamental harmonic
  - ➢ I'_T - one period of $I'(n)$ for all harmonics
  - ➢ save I'_T along with filtered signal to recover original signal
  - ➢ signals $s'(n)$-$I'(n)$ and $I'(n)$ are orthogonal by definition
  - ➢ for optimal filtering a_k can be found by Fourier transform of I'_T

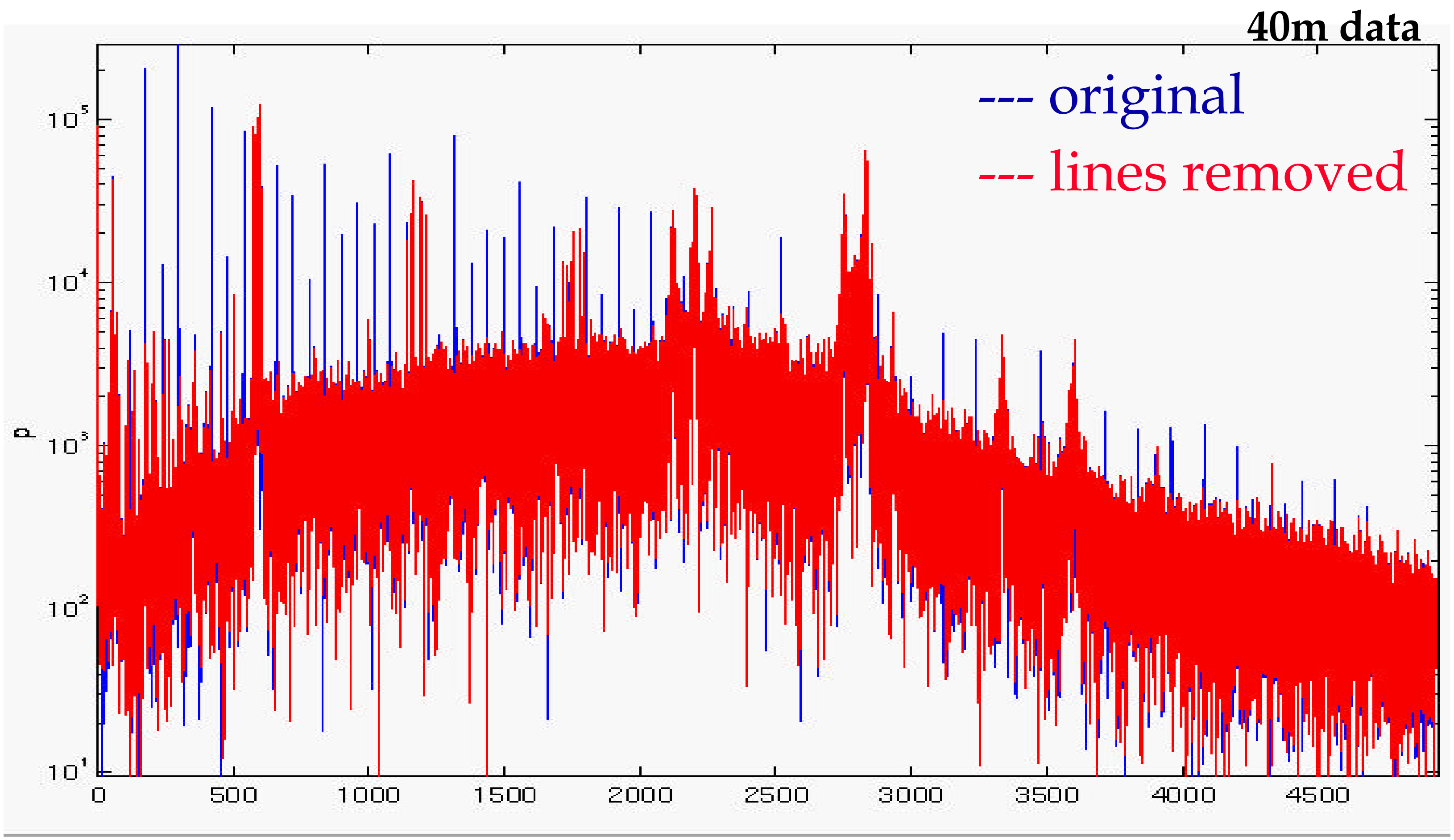


Power Lines Interference



Power Lines Removal

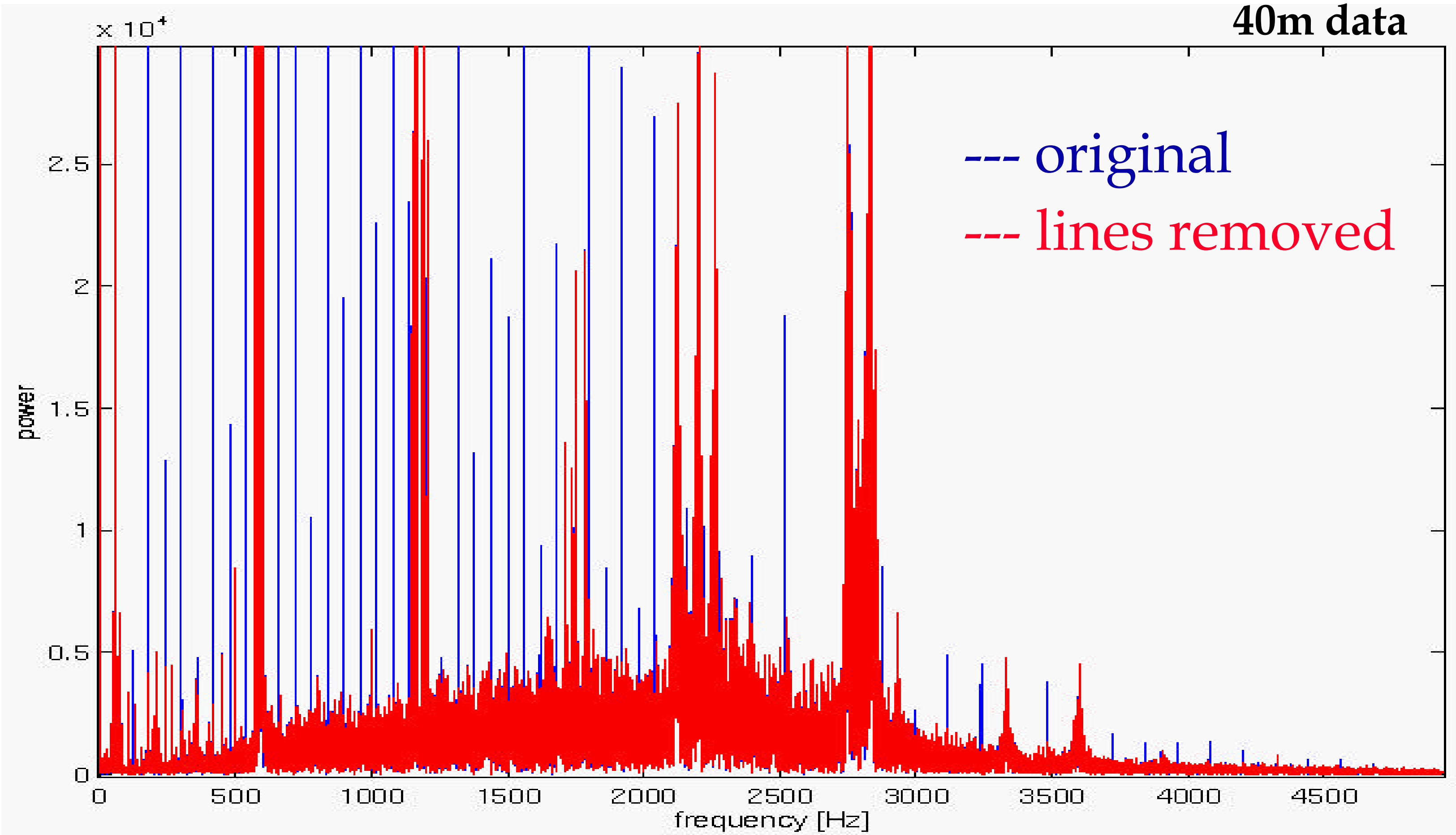
- Blue - Fourier spectra with power lines, red - Fourier spectra with power lines removed. (T=5sec)



S.Klimenko

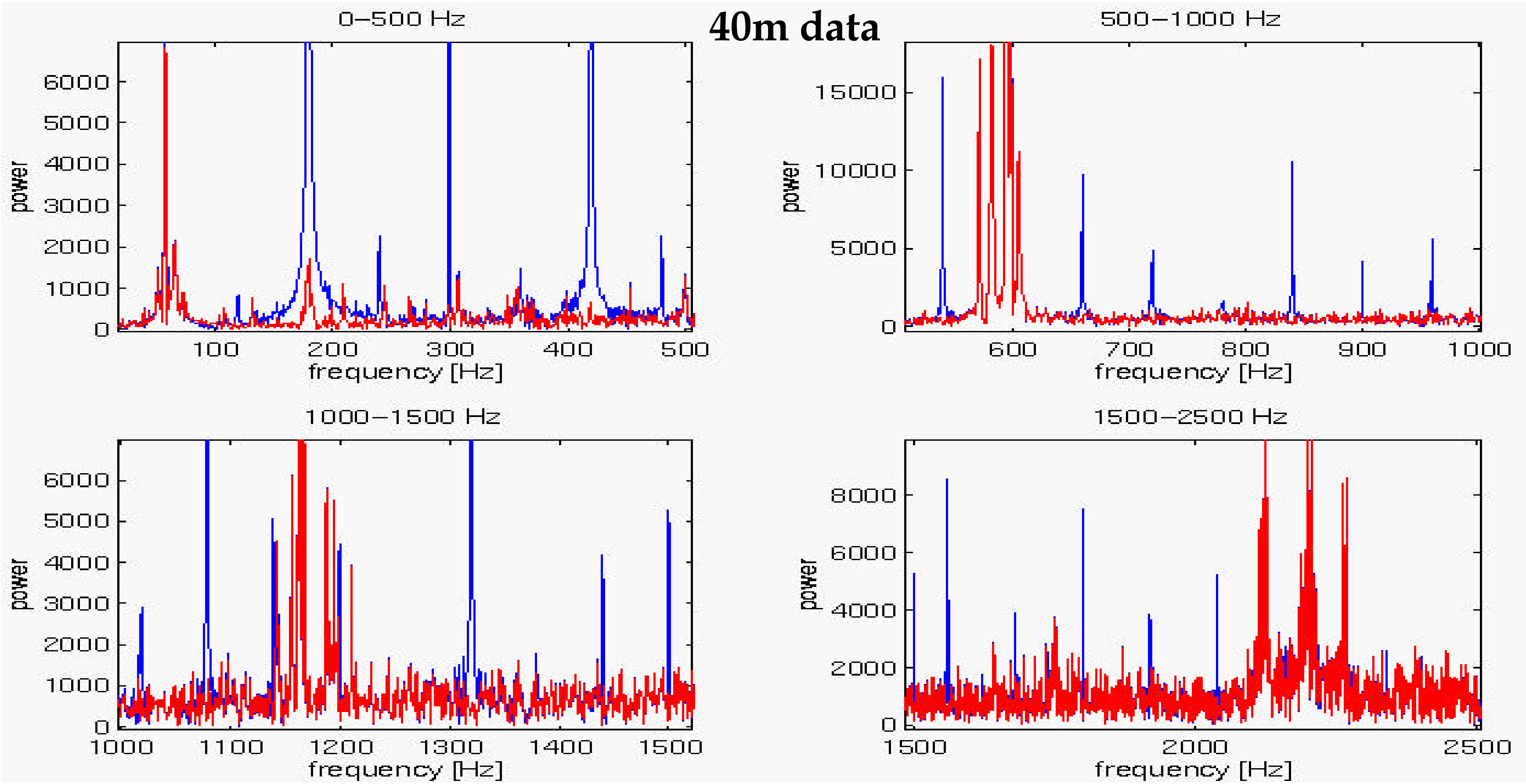
Power Lines Removal

- blue - Fourier spectra with power lines, red - Fourier spectra with power lines removed. (T=5sec)



Power Lines Removal

- Sample of 10000 points, (1sec), blue - data with lines, red - lines removed

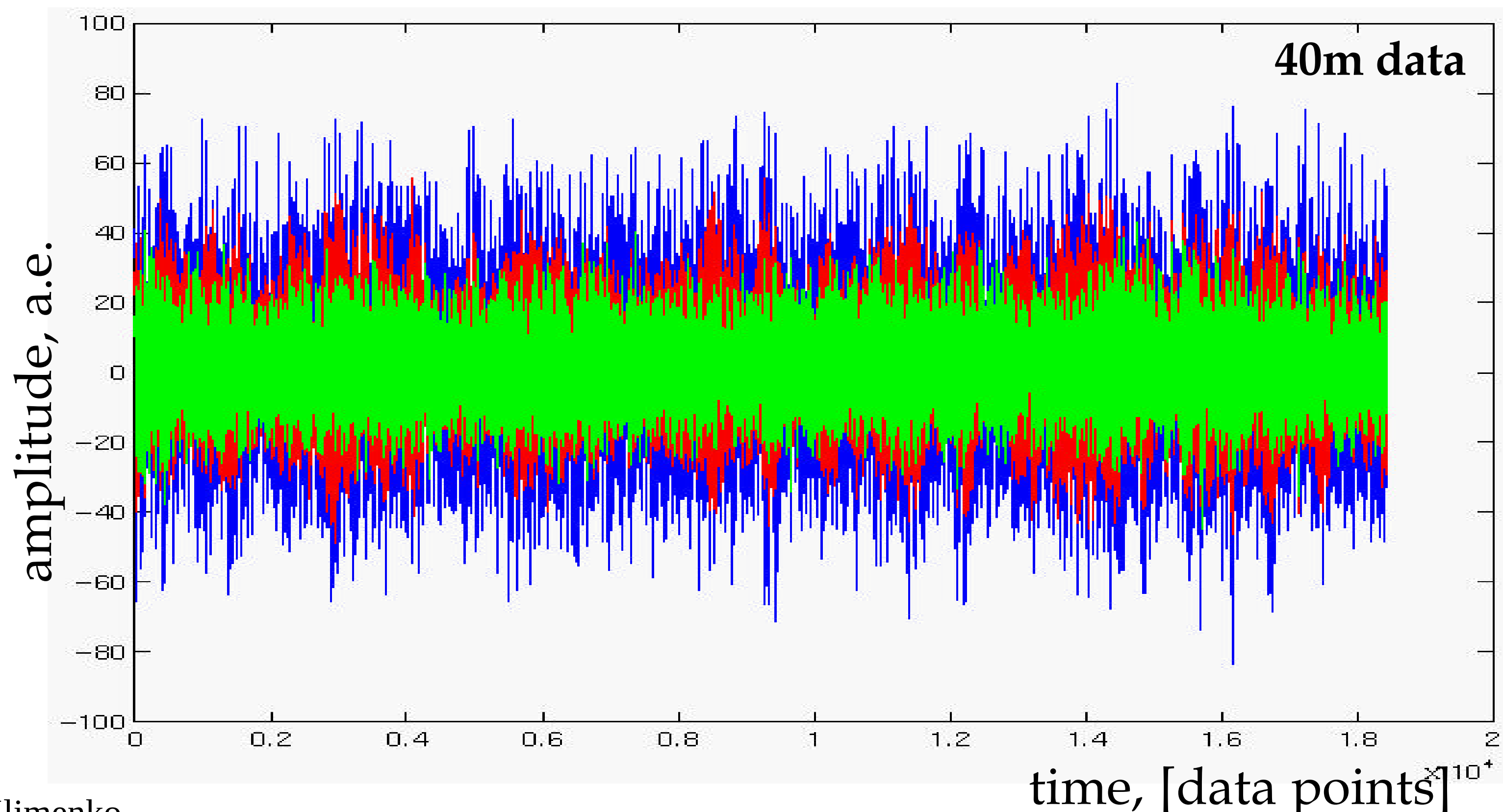


Power Lines

#	frequency	A	A/N	#	frequency	A	A/N	#	frequency	A	A/N
1	60.00190	0.191	0.24	26	1560.0494	0.776	16.16	51	3060.0969	0.0109	0.278
2	120.0038	0.096	9.45	27	1620.0513	0.203	1.15	52	3120.0988	0.0858	2.285
3	180.0057	4.074	40.71	28	1680.0532	0.453	2.87	53	3180.1007	0.0172	1.007
4	240.0076	0.261	14.82	29	1740.0551	0.164	1.00	54	3240.1026	0.1175	5.884
5	300.0095	7.223	107.37	30	1800.0570	0.706	15.49	55	3300.1045	0.0393	2.658
6	360.0114	0.078	0.37	31	1860.0589	0.204	2.20	56	3360.1064	0.0138	0.276
7	420.0133	2.344	23.78	32	1920.0608	0.684	6.87	57	3420.1083	0.0148	0.717
8	480.0152	0.299	7.33	33	1980.0627	0.116	1.07	58	3480.1102	0.0836	2.740
9	540.0171	1.663	58.76	34	2040.0646	0.667	5.39	59	3540.1121	0.0150	0.637
10	600.0190	1.033	0.86	35	2100.0665	0.210	0.85	60	3600.1140	0.0702	0.549
11	660.0209	0.989	13.61	36	2160.0684	0.264	1.54	61	3660.1159	0.0124	1.204
12	720.0228	0.619	5.87	37	2220.0703	0.230	0.69	62	3720.1178	0.0299	2.684
13	780.0247	0.159	2.22	38	2280.0722	0.123	0.50	63	3780.1197	0.0066	0.535
14	840.0266	1.045	9.55	39	2340.0741	0.069	0.60	64	3840.1216	0.0244	1.617
15	900.0285	0.472	12.66	40	2400.0760	0.253	2.69	65	3900.1235	0.0307	2.129
16	960.0304	0.673	9.64	41	2460.0779	0.087	0.53	66	3960.1254	0.0297	2.271
17	1020.032	0.472	5.79	42	2520.0798	0.366	1.92	67	4020.1273	0.0044	0.846
18	1080.034	1.343	11.26	43	2580.0817	0.037	0.45	68	4080.1292	0.0252	3.508
19	1140.036	0.591	4.33	44	2640.0836	0.049	0.52	69	4140.1311	0.0053	0.675
20	1200.038	0.427	2.58	45	2700.0855	0.081	0.60	70	4200.1330	0.0225	1.975
21	1260.039	0.111	1.36	46	2760.0874	0.163	0.29	71	4260.1349	0.0072	1.042
22	1320.041	1.667	16.98	47	2820.0893	0.469	0.49	72	4320.1368	0.0056	0.690
23	1380.043	0.322	1.80	48	2880.0912	0.215	3.06	73	4380.1387	0.0087	2.579
24	1440.045	0.381	3.95	49	2940.0931	0.034	0.17	74	4440.1406	0.0078	1.653
25	1500.047	0.445	4.08	50	3000.0950	0.123	1.45	75	4500.1425	0.0016	0.272

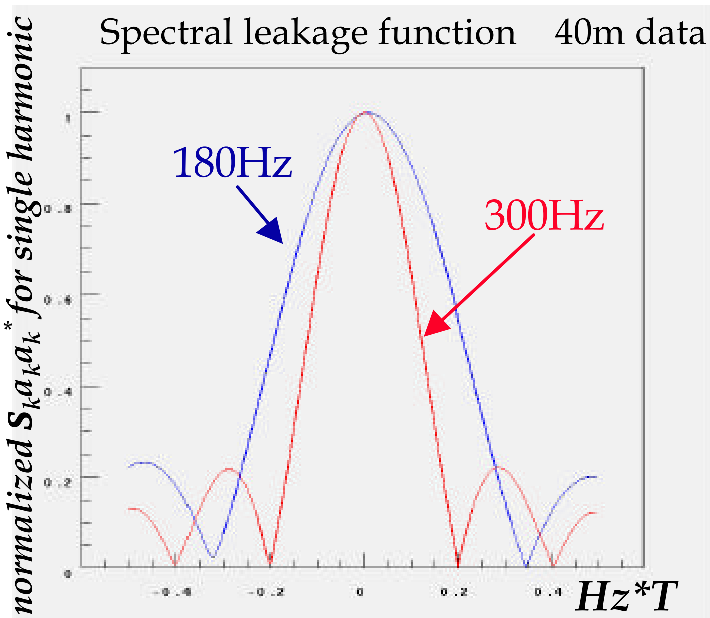
Signal Amplitude

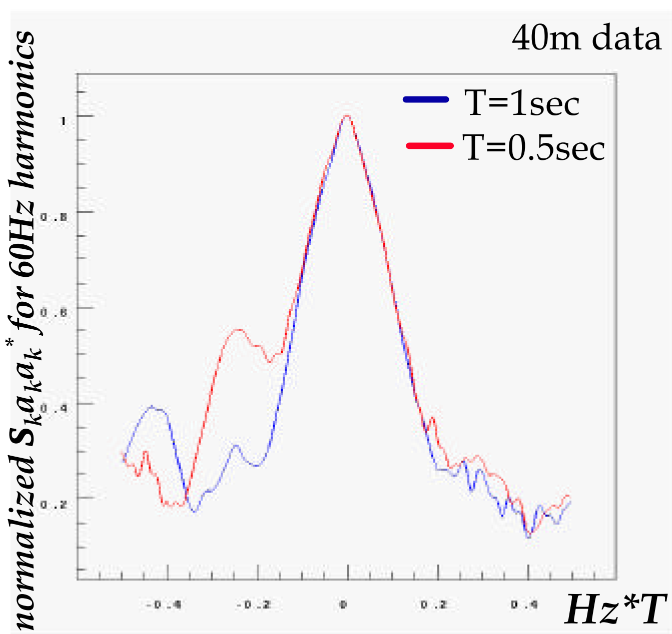
- blue - original data, red - 60Hz lines removed, green - 582.4Hz lines removed. Signal energy: $23.4^2 : 17.5^2 : 10.8^2$
- Energy balance: $dE = \langle s^2 \rangle - \langle I^2 \rangle - \langle (s-I)^2 \rangle$, $dE/\langle s^2 \rangle \sim 10^{-4}$



Fundamental Line Frequency

- To use line removal algorithm, an accurate prediction of line fundamental frequency f is required.
- f estimation:
  - DFT of data gives rough estimate of f: $\delta f \sim f_0/N = 1/T$
  - re-sample data for given f and find harmonic amplitudes a_k
  - tune f to maximize $S_k a_k a_k^*$ for all (or group of) harmonics

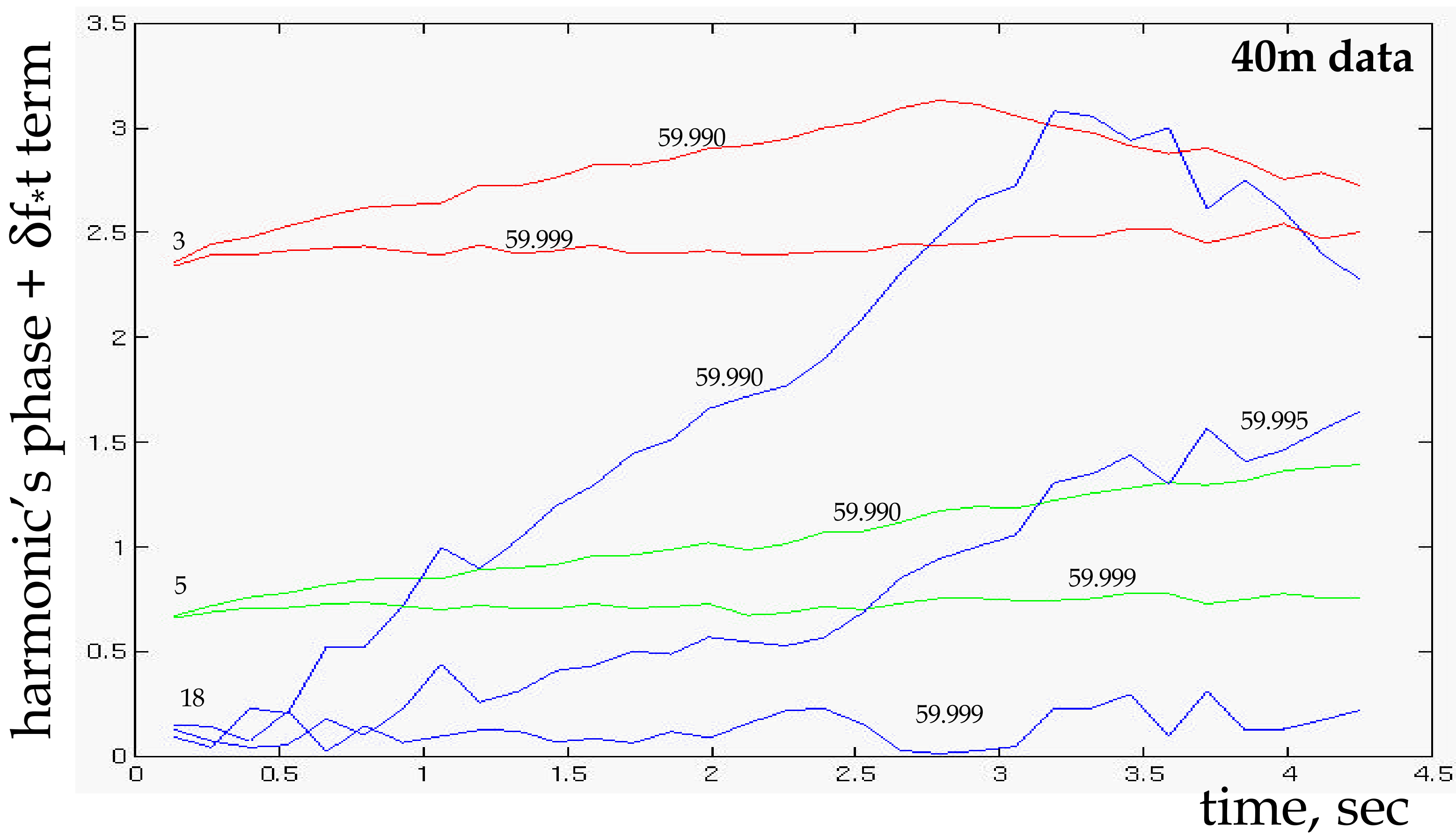




S.Klimenko

Phase of Harmonics

- 3,5,18 harmonic's phase (δf*t term) for 32 samples of data (0.132 sec each) for different fundamental frequencies.



S.Klimenko

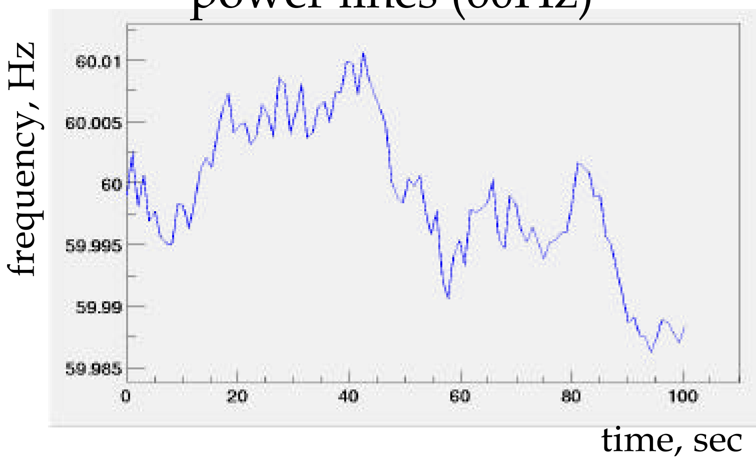
Line Removal Software

- s.resample(s', f_s)
  - re-sample data s' with sampling rate f_s .
  - uses polynomial interpolation
- s.extract(f, n, m, E)
  - extract n-m harmonics of frequency f from data s.
  - E - total energy of harmonics n-m
  - uses constant weight function
- s.tune(f, n, m)
  - tune fundamental frequency f maximizing $E=\Sigma_n a_k a_k^*$ for harmonics n-m and data set s
  - seed value of f can be taken from previous data set.
- LRS can be used for E, f monitoring & fast lines removal.
- LRS could be included into the Data Monitoring Tool (?)

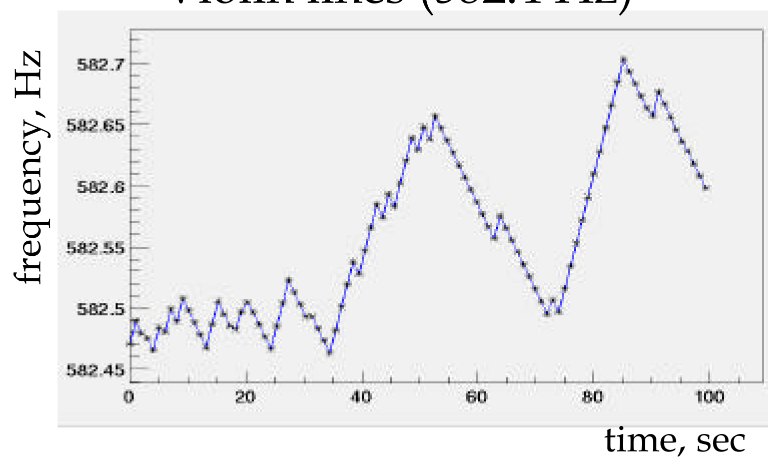
Monitoring of harmonic lines

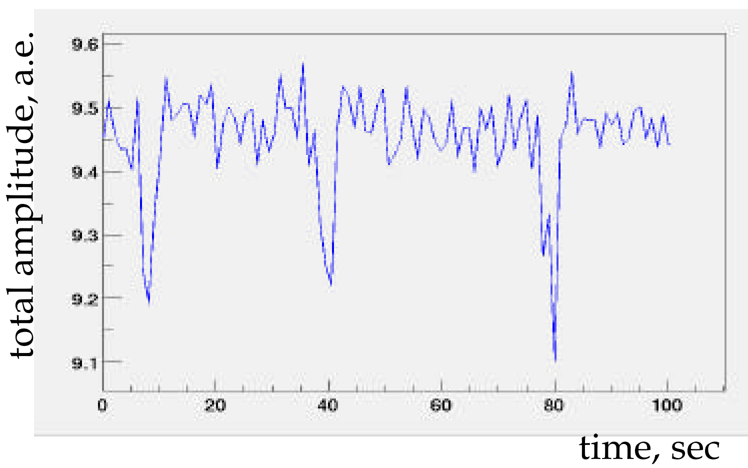
- 100 sec stretch of 40m data (20 frames)

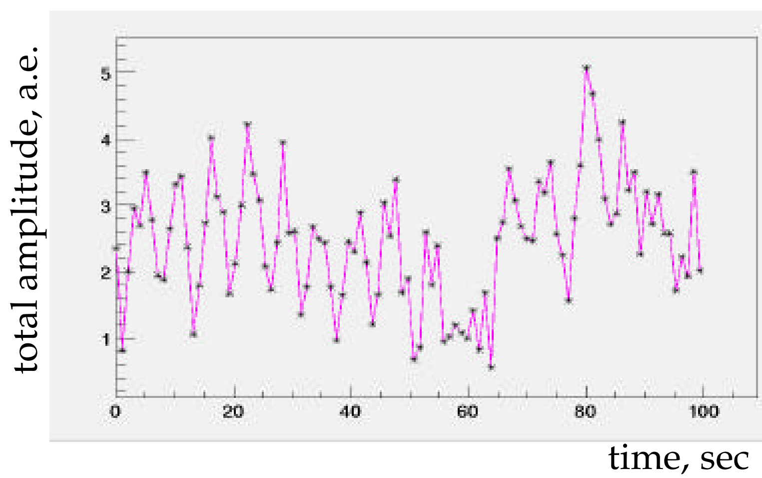
power lines (60Hz)



Violin lines (582.4 Hz)







S.Klimenko

Conclusion

- Fast and simple algorithm for coherent lines removal is presented
- Single harmonic, group of harmonics or all harmonics of fundamental frequency f can be removed in one shot.
- Relatively small amount of information need to be saved to recover original data.
- Coherent line removal software has been developed (C++ class-library). It can be used
  - ➢ to reduce non-Gaussian noise in data
  - ➢ to monitor harmonic lines.

Re-sampling artifacts

$$f(nD) \xrightarrow{f_0} f'(nD') \xrightarrow{f_s} f''(nD);$$