Data Conditioning for Gravitational Wave Burst Analysis

Shourov K. Chatterji

shourov@ligo.mit.edu

LIGO Scientific Collaboration

LIGO

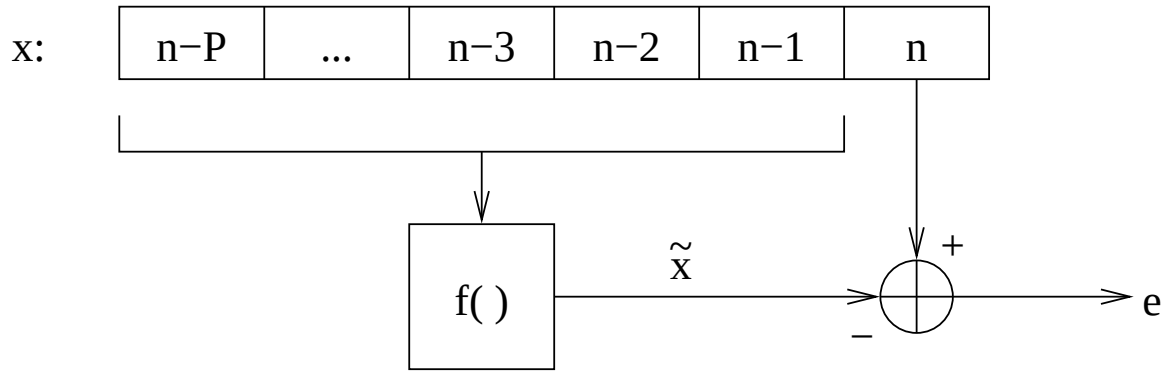
Linear Predictors

Predict future values of a time series from previous values

- Remove predictable signal content
  - Whitening and Line Removal
    - Simplified burst search
  - Unanticipated bursts detection
    - Glitch investigations
  - Necessary for Cross-correlation
    - Untriggered burst search (Cadonati)
    - Externally triggered burst search (Marka)
- Remove inter-dependent signal content
  - Continuous time domain veto

Single Channel Predictors

General Predictor:



$$x_n = f\left(\{x_{n-m}\}\right) \quad 1 \le m \le P$$

How do we choose $f()$?

LIGO

Single Channel Linear Predictors

Choose $f()$ to minimize the mean squared error.

$$\sigma_e^2 = \frac{1}{N}\sum_{n=1}^{N}(x_n - \tilde{x}_n)^2$$

This is an optimal filtering problem.
Restrict the problem to the class of FIR linear filters.

$$\tilde{x}_n^0 = \sum_{m=P_0}^{Q_0} c_m^{00} x_{n-m}^0$$

Yule-Walker Equations

$$\sum_{m=P_0}^{Q_0} c_m^{00} r_{m-k}^{00} = r_k^{00} \quad \text{for} \quad P_0 \le k \le Q_0.$$

$$\begin{bmatrix} r_0^{00} & r_1^{00} & r_2^{00} & \cdots & r_{Q_0-P_0}^{00} \\ r_1^{00} & r_0^{00} & r_1^{00} & \cdots & r_{Q_0-P_0-1}^{00} \\ r_2^{00} & r_1^{00} & r_0^{00} & \cdots & r_{Q_0-P_0-2}^{00} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ r_{Q_0-P_0}^{00} & r_{Q_0-P_0-1}^{00} & r_{Q_0-P_0-2}^{00} & \cdots & r_0^{00} \end{bmatrix} \begin{bmatrix} c_{P_0}^{00} \\ c_{P_0+1}^{00} \\ c_{P_0+2}^{00} \\ \vdots \\ c_{Q_0}^{00} \end{bmatrix} = \begin{bmatrix} r_{P_0}^{00} \\ r_{P_0+1}^{00} \\ r_{P_0+2}^{00} \\ \vdots \\ r_{Q_0}^{00} \end{bmatrix}$$

Linear Prediction Error Filter

$$e_n^0 = x_n^0 - \tilde{x}_n^0 = \sum_{m=0}^{Q_0} h_m^{00} x_{n-m}^0$$

$$h_m^{00} = \begin{cases} 1 & m = 0 \\ -c_m^{00} & P_0 \le m \le Q_0 \\ 0 & \text{otherwise} \end{cases}$$

White Noise

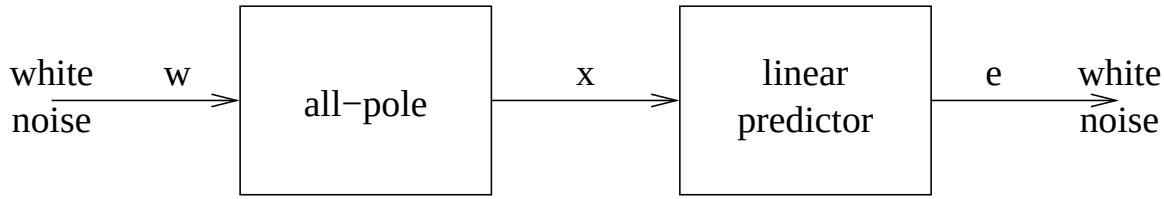
$$r_n^{00} = \begin{cases} 1 & n = 0 \\ 0 & \text{otherwise} \end{cases}$$

$$\begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix} \begin{bmatrix} c_{P_0}^{00} \\ c_{P_0+1}^{00} \\ c_{P_0+2}^{00} \\ \vdots \\ c_{Q_0}^{00} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Therefore all coefficients are zero.

LPEF Order

- LPEF can exactly compensate for all-pole LTI model



- IIR filter compensates for arbitary linear model
- FIR filter is rectangularly windowed IIR filter
- Ideal response convolved with spectrum of window
- LPEF compensates for features with

$$\text{bandwidth} \geq \frac{\text{sample rate}}{\text{filter order}} \quad \text{Hz}$$

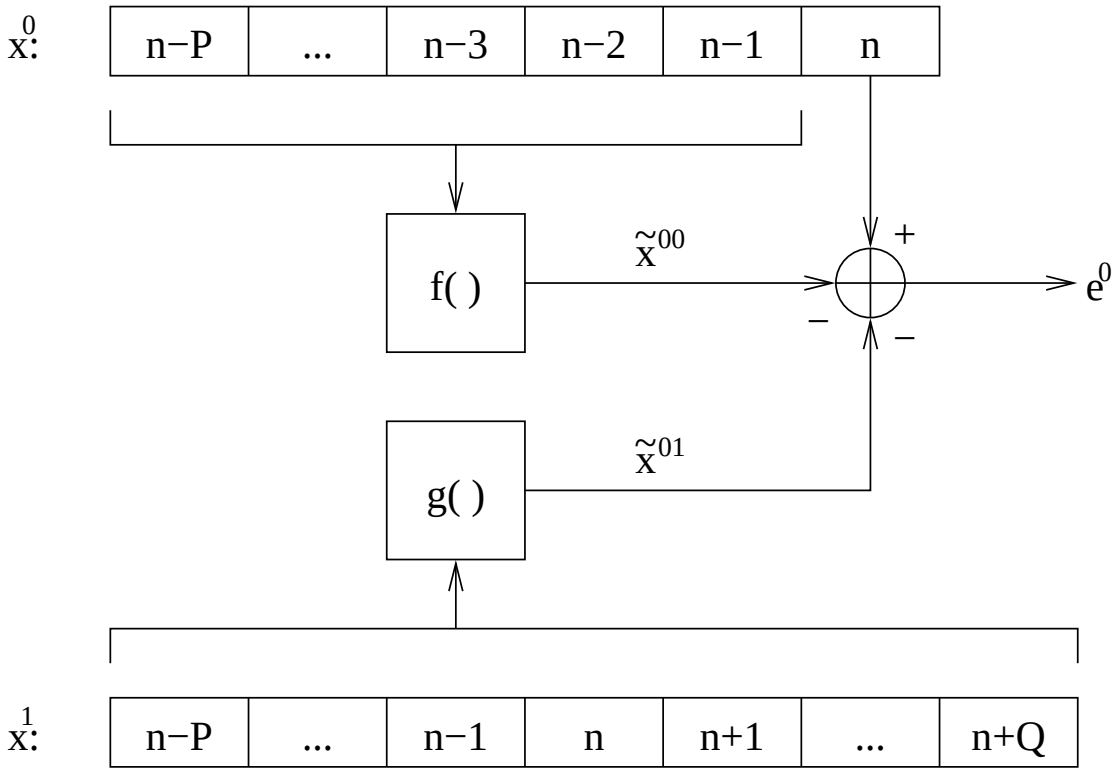
LPEF Training Time

- LPEF learns about features with

$$\text{bandwidth} > \text{training time}^{-1} \quad \text{Hz}$$

Multi-Channel Predictors

General Multi-Channel Predictor:



$$x_n^0 = f\left(\{x_{n-i}^0\}\right) + g\left(\{x_{n-j}^1\}\right) \quad P^0 \le i \le Q^0 \quad P^1 \le j \le Q^1$$

Multi-Channel Linear Predictor: Definition

Restrict the problem to the class of FIR linear filters.

$$\tilde{x}_n^0 = \sum_{i=0}^{\mathcal{I}} \sum_{m=P_i}^{Q_i} c_m^{0i} x_{n-m}^i$$

Multi-Channel Yule-Walker

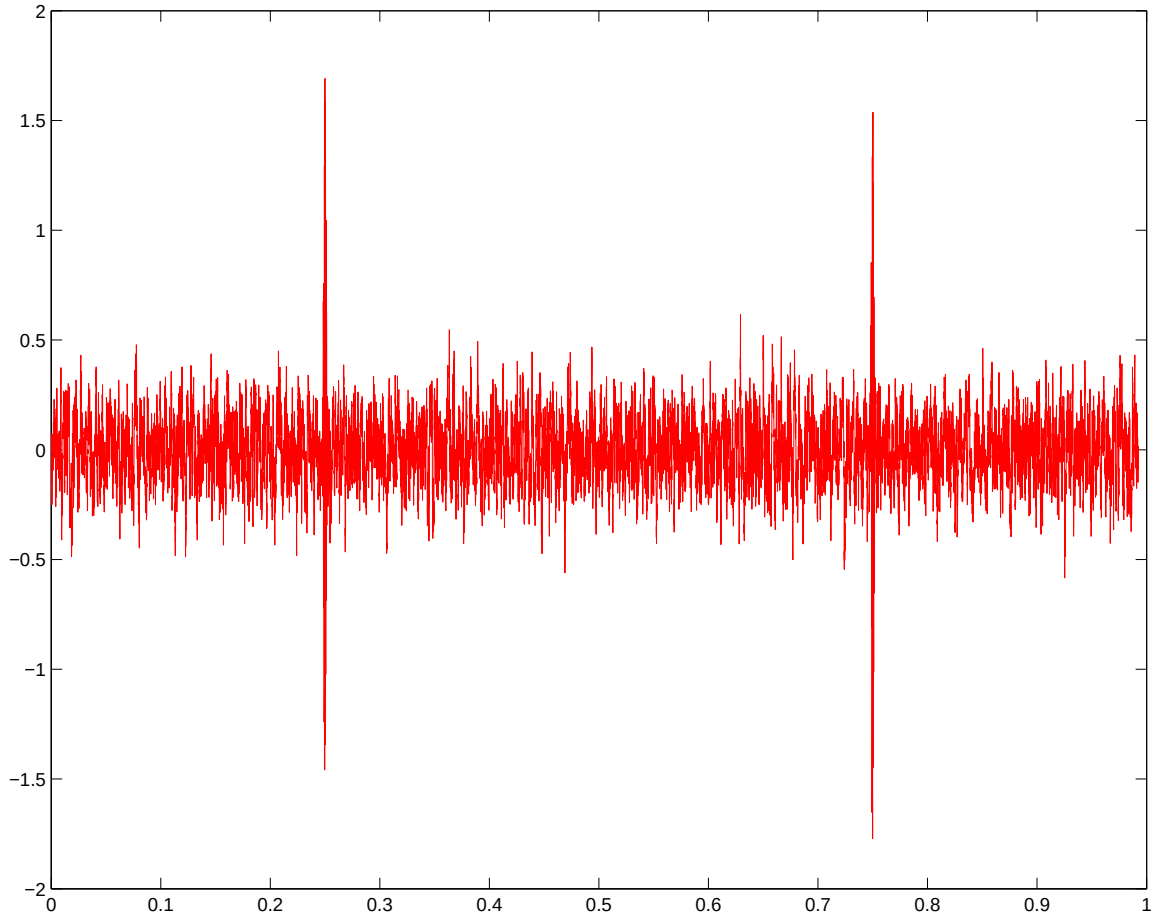
- Once again, this produces a symmetric matrix equation for the unknown filter coefficients.
- However, it is not toeplitz and the Levinson-Durbin recursion algorithm cannot be applied.
- It is, however, composed of smaller symmetric toeplitz sub-matrices of the auto and cross-correlation sequences between channels.

Veto Example

In the following example, a two channel predictor was used to remove dependent signal content between a simulated gravity channel and simulated auxilliary channel. The following three pulses were injected.

- $t = 0.25$ gravity channel only
- $t = 0.50$ both gravity channel and auxilliary channel
- $t = 0.75$ auxilliary channel only

Veto Example Error Signal



Adaptive Predictors

What if the signal content is not stationary?

- Discrete
  - periodic re-training
  - threshold criteria
- Continuous
  - update autocorrelation matrix
  - recursive least squares

This is no longer truly linear!

Cross Correlation I

$$x_1(t) = b_1(t) * (n_1(t) + r_1(t) * h_1(t))$$
$$x_2(t) = b_1(t) * (n_2(t) + r_2(t) * h_2(t))$$

$$X_1(f) = B_1(f)\,(N_1(f) + R_1(f)H_1(f))$$
$$X_2(f) = B_2(f)\,(N_2(f) + R_2(f)H_2(f))$$

$h_1(t)$, $h_2(t)$ gravitational-wave strains
$r_1(t)$, $r_2(t)$ interferometer responses
$n_1(t)$, $n_2(t)$ intrinsic and extrinsic noise
$b_1(t)$, $b_2(t)$ arbitrary linear filters
$x_1(t)$, $x_2(t)$ measured responses

Cross Correlation II

$$
\begin{aligned}
X_{1,2}(f) &= B_{1,2}(f)N_{1,2}(f) + B_{1,2}(f)R_{1,2}(f)H_{1,2}(f) + \\
&\quad B_{1,2}(f)N_1(f)R_2^*(f)H_2^*(f) + \\
&\quad B_{1,2}(f)N_2^*(f)R_1(f)H_1(f)
\end{aligned}
$$

$$
\begin{aligned}
x_{1,2}(t) &= b_{1,2}(t) * \{n_{1,2}(t) + r_{1,2}(t) * h_{1,2}(t) + \\
&\quad \text{crosscorr}\,(n_1(t), r_2(t) * h_2(t)) + \\
&\quad \text{crosscorr}\,(r_1(t) * h_1(t), n_2(t))\}
\end{aligned}
$$

Cross Correlation III

$$x_{1,2}(t) = b_{1,2}(t) * \{n_{1,2}(t) + r_{1,2}(t) * h_{1,2}(t)\}$$

$h_{1,2}(t)$ this is the desired result
$n_{1,2}(t)$ correlated background noise
$r_{1,2}(t)$ disperses desired result
$b_{1,2}(t)$ cancels stationary component of $n_{1,2}(t)$ and disperses desired result

DMT LPEFilter Class

class LPEFilter : public FIRFilter

- LPEFilter(int order, int period, int length)
- apply(TSeries &in, TSeries &out)
- train(TSeries &data)
- setFilterOrder(int order)
- setTrainPeriod(int period)
- setTrainLength(int length)
- setHistory(const TSeries &in)
- reset(void)

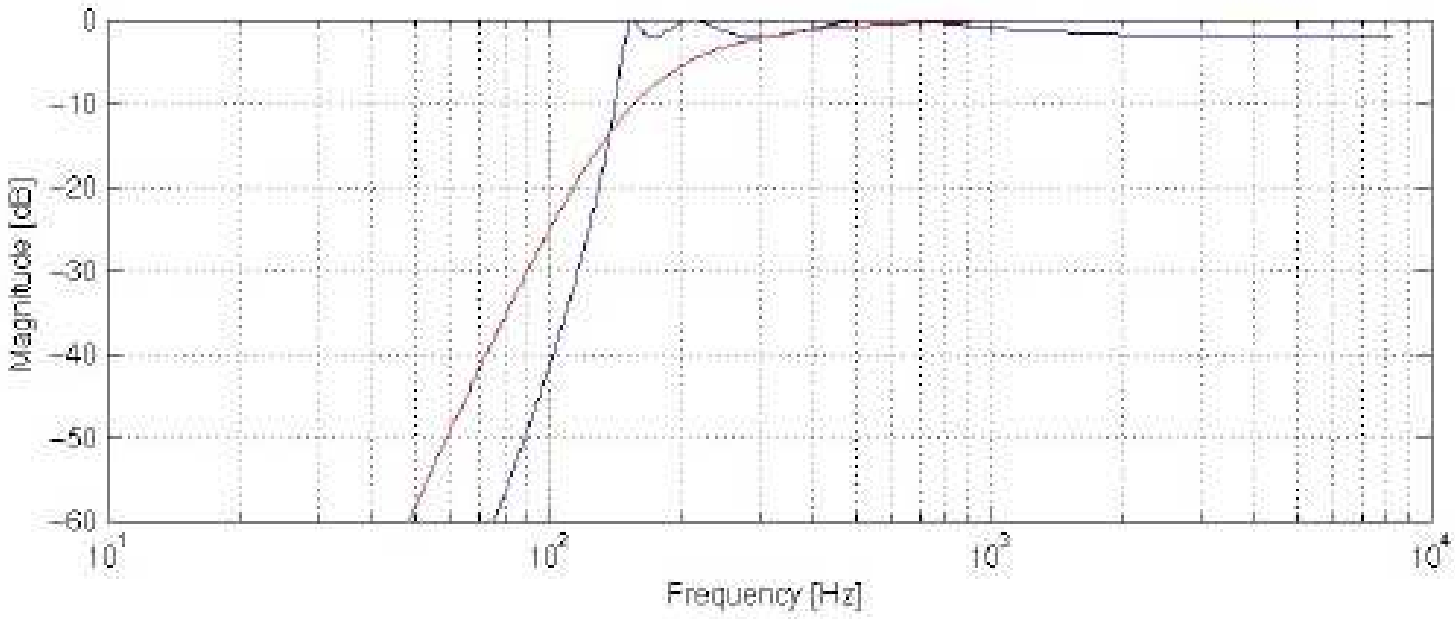
http::/ligo.mit.edu/~shourov/lpef/dmt

Matlab Algorithm Library

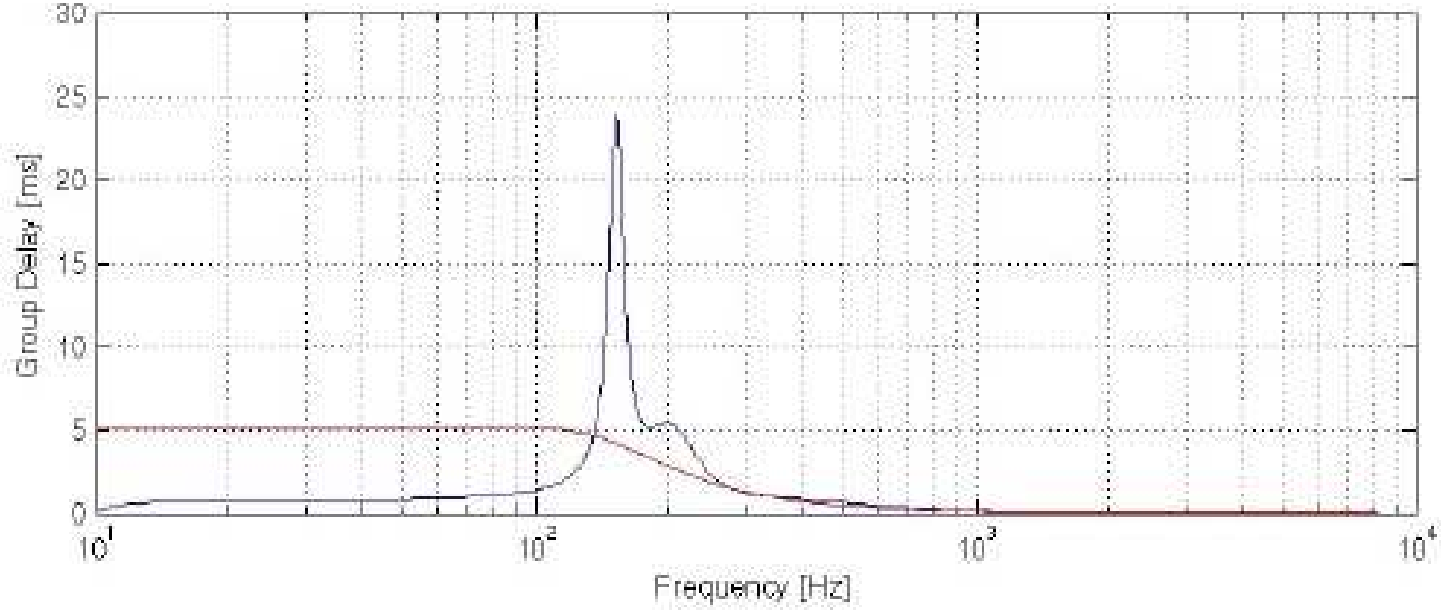
http://emvogil-3.mit.edu/~shourov/matlab/
Contributions welcome

GETFRAMEDATA	Extract a single channel of data from frame files
LPF	Modified butterworth low pass filter coefficients
HPF	Modified butterworth high pass filter coefficient
BPF	Modified butterworth band pass filter coefficient
NOTCH	Second order notch filter coefficients
NOTCHES	Aggregate notch filter for multiple line sources
LPEFC	Linear predictive error filter coefficients
FIRFILTER	Piecewise FIR block filtering via FFT
SQRTFIR	FIR filter with square root magnitude response
FINDLINES	Identify line frequencies and bandwidths
PSDBASE	Smooth fit to power spectrum baseline noise
SOSFILTFILT	Zero-phase filtering with second-order sections
PREPROCESS	Data conditioning for cross-correlation based analysis
FFTFFT	Simultaneous fast fourier transform of two real sequences
ILWDWRITE	Write ilwd trigger file

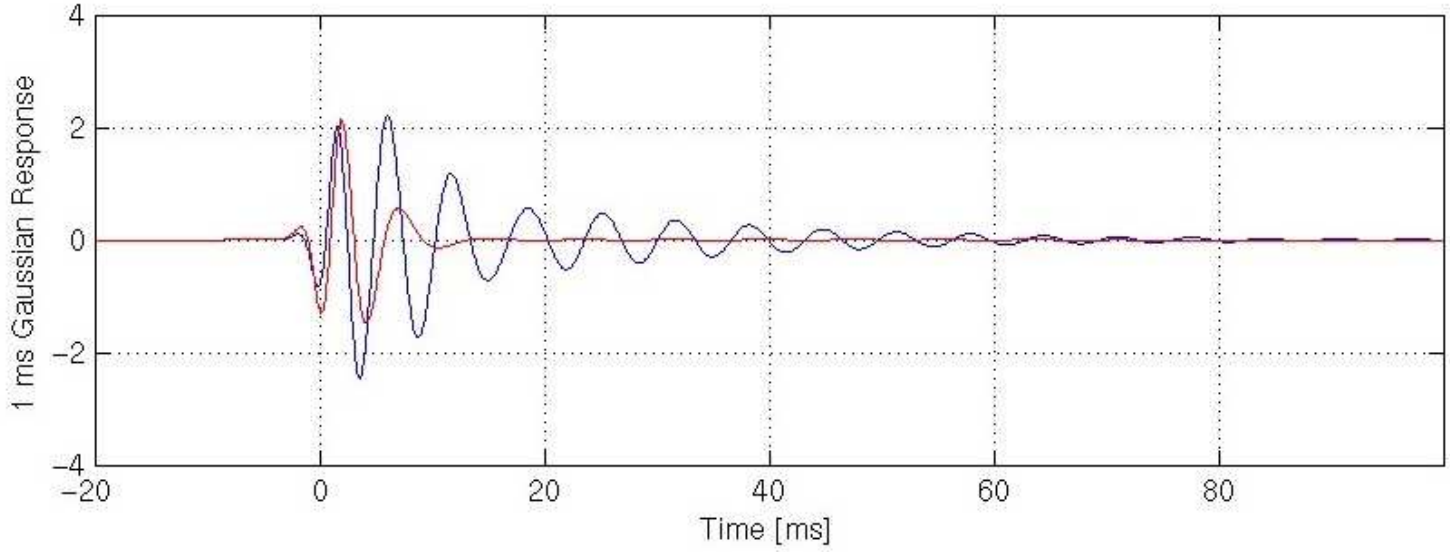
Reduced Filter Dispersion I



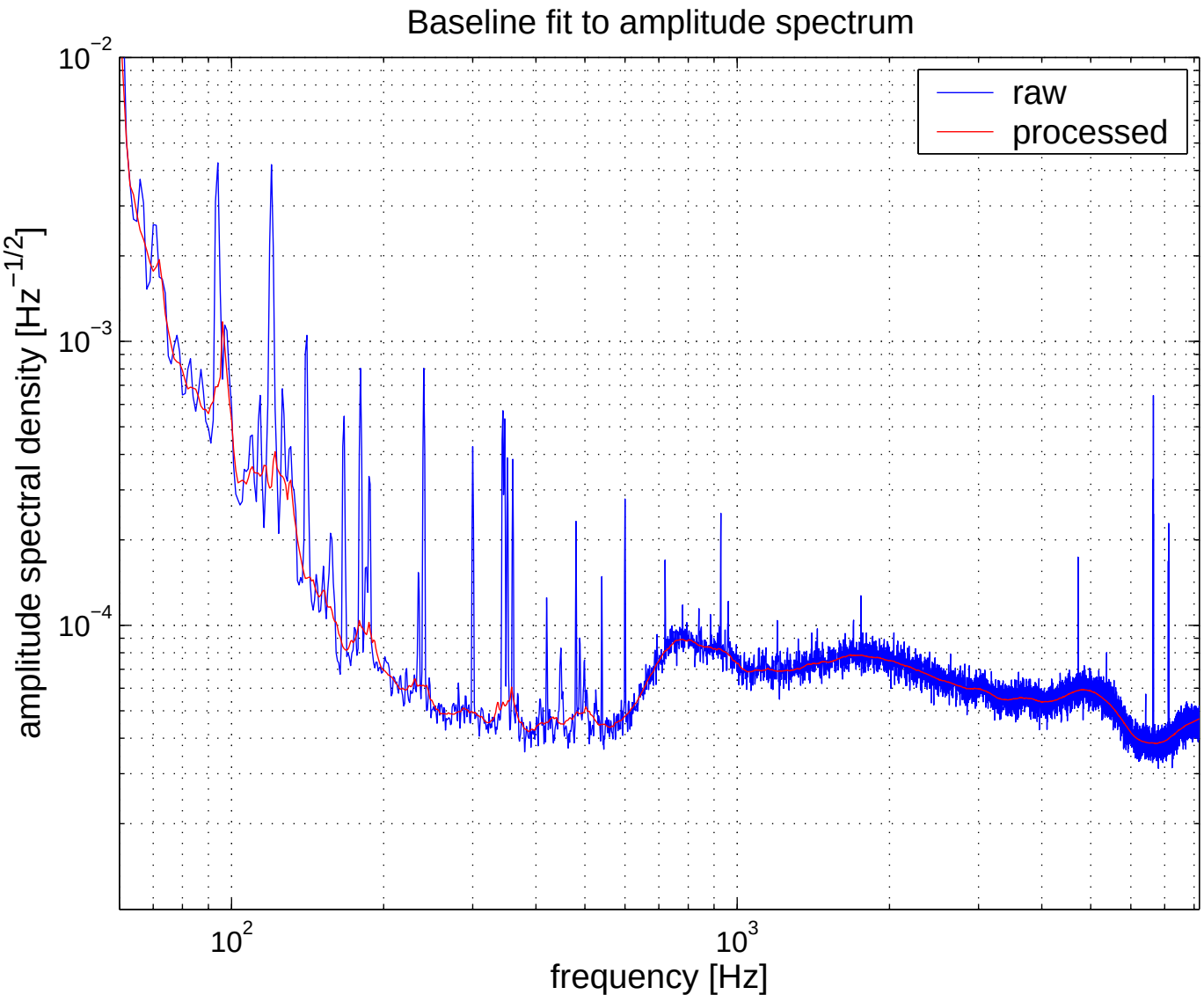
Reduced Filter Dispersion II



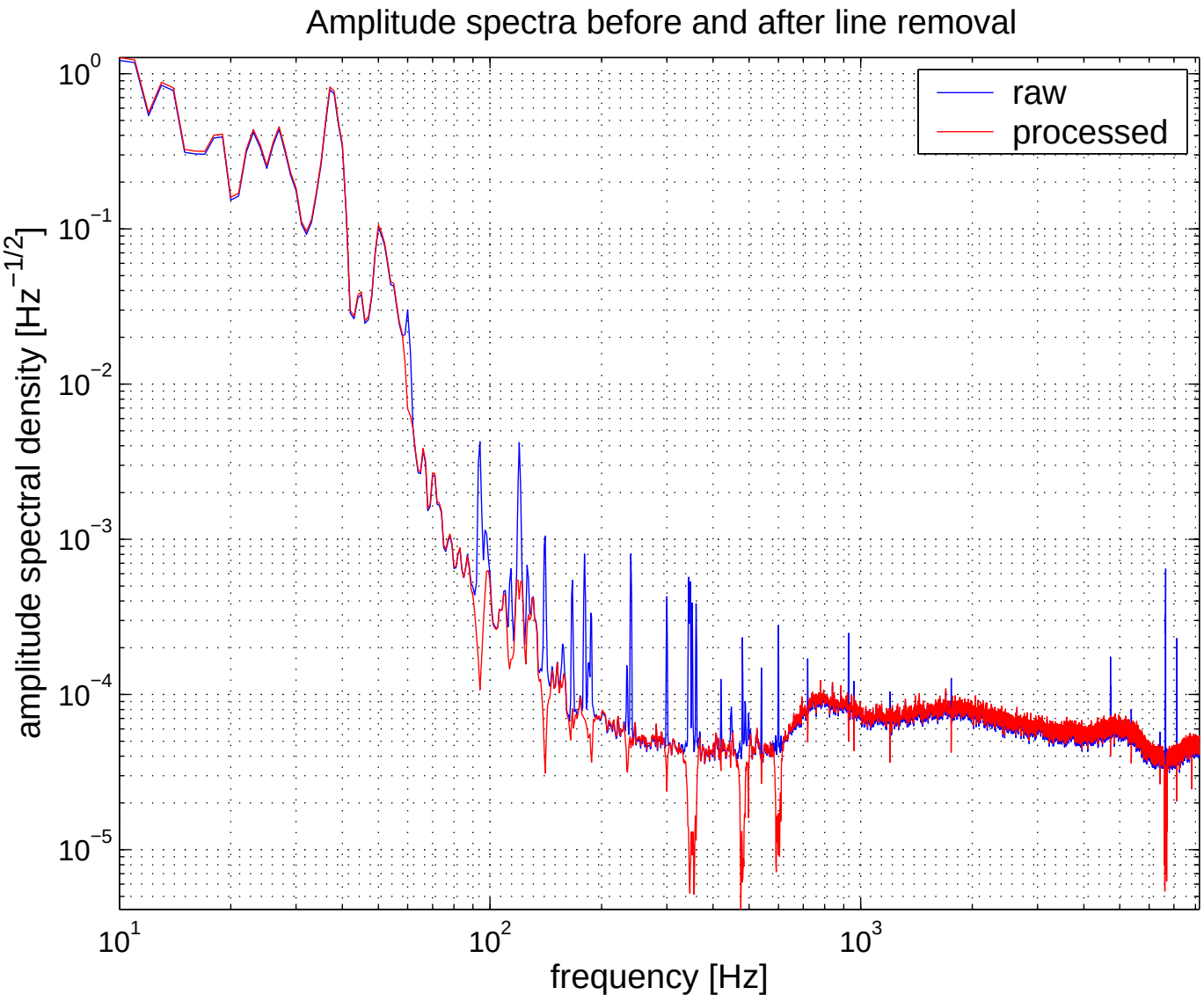
Reduced Filter Dispersion III



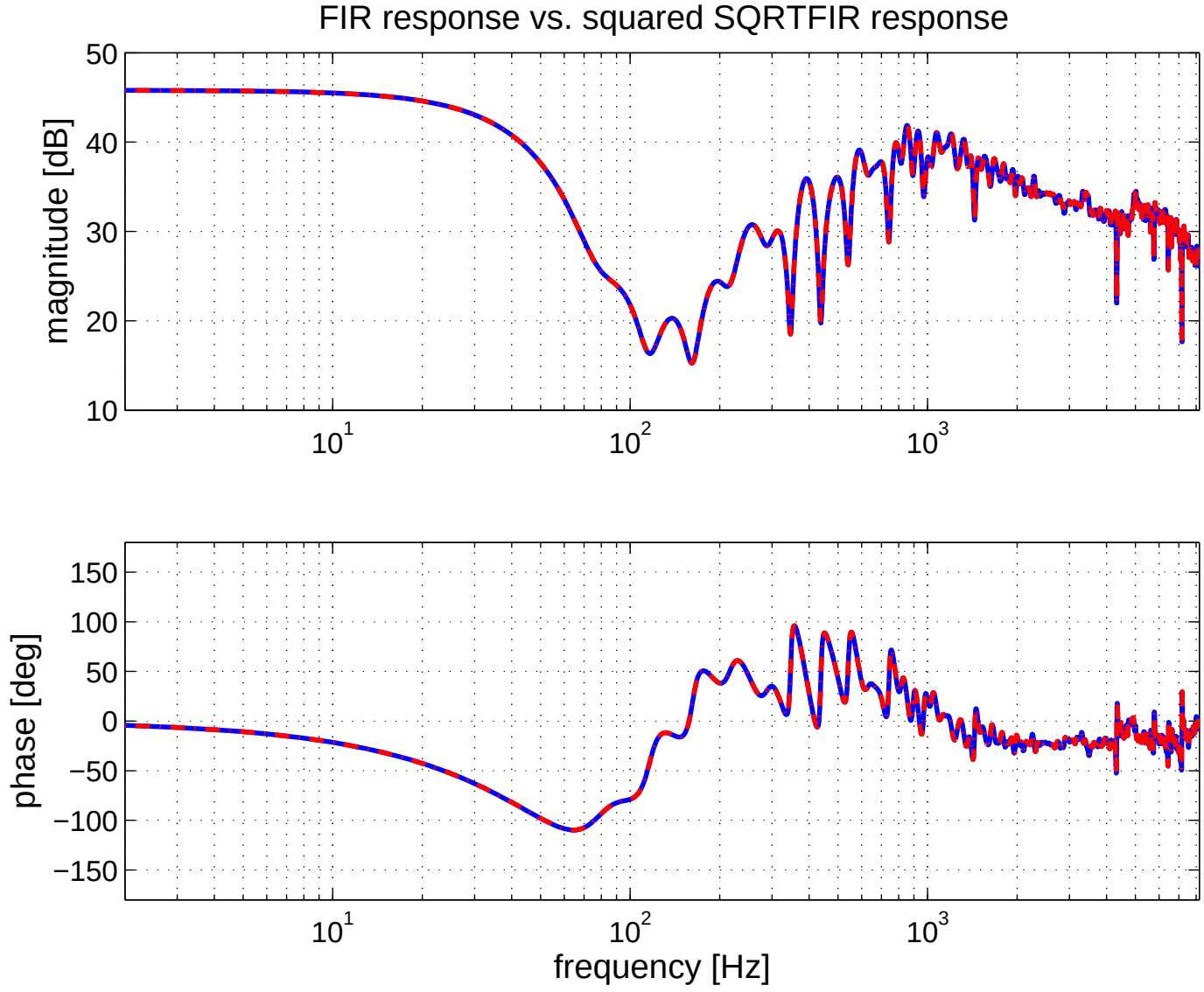
Baseline Fitting



Line Finding and Notching



Square Root FIR Magnitude Response



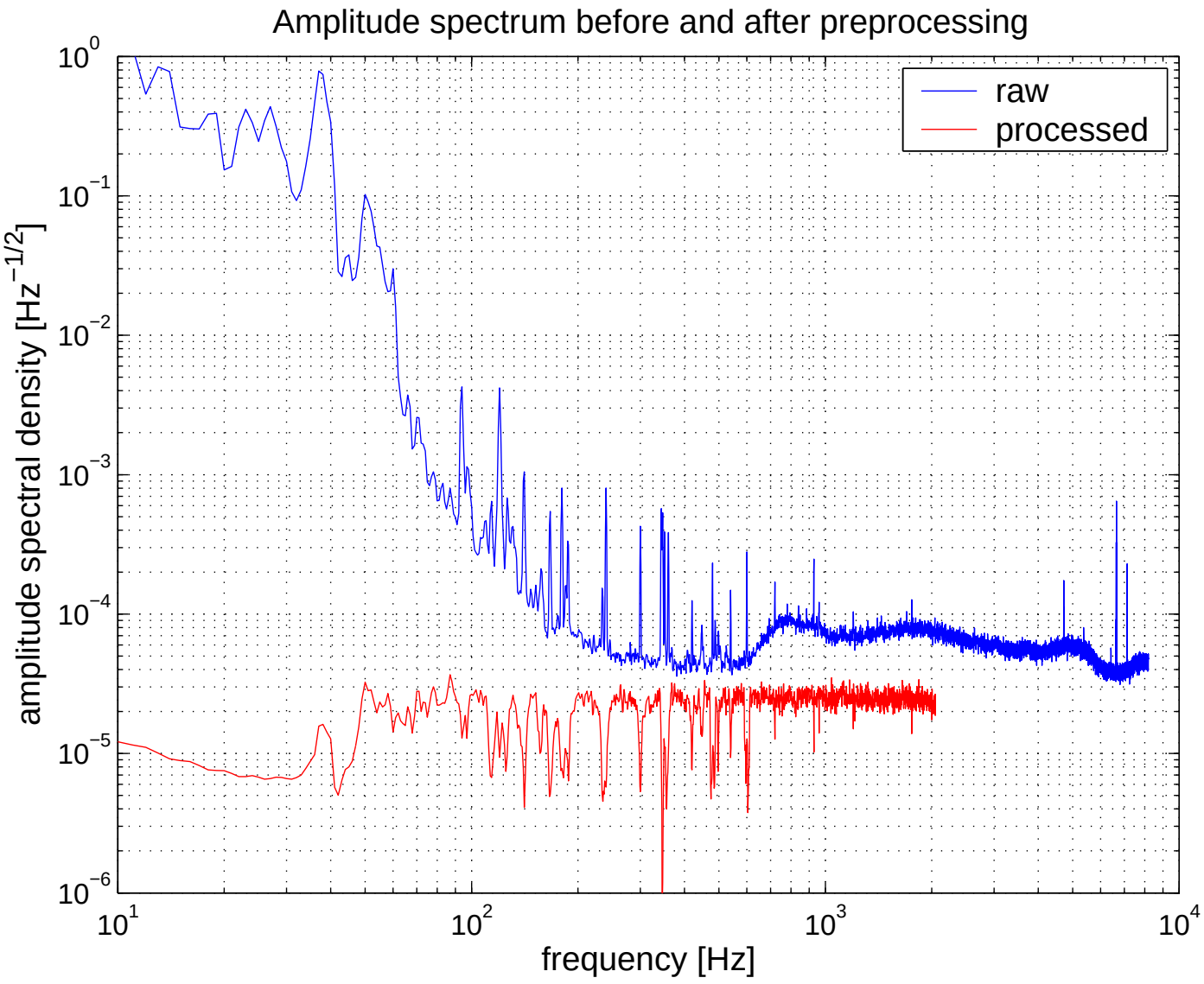
LIGO

Conditioning for Cross-Correlation

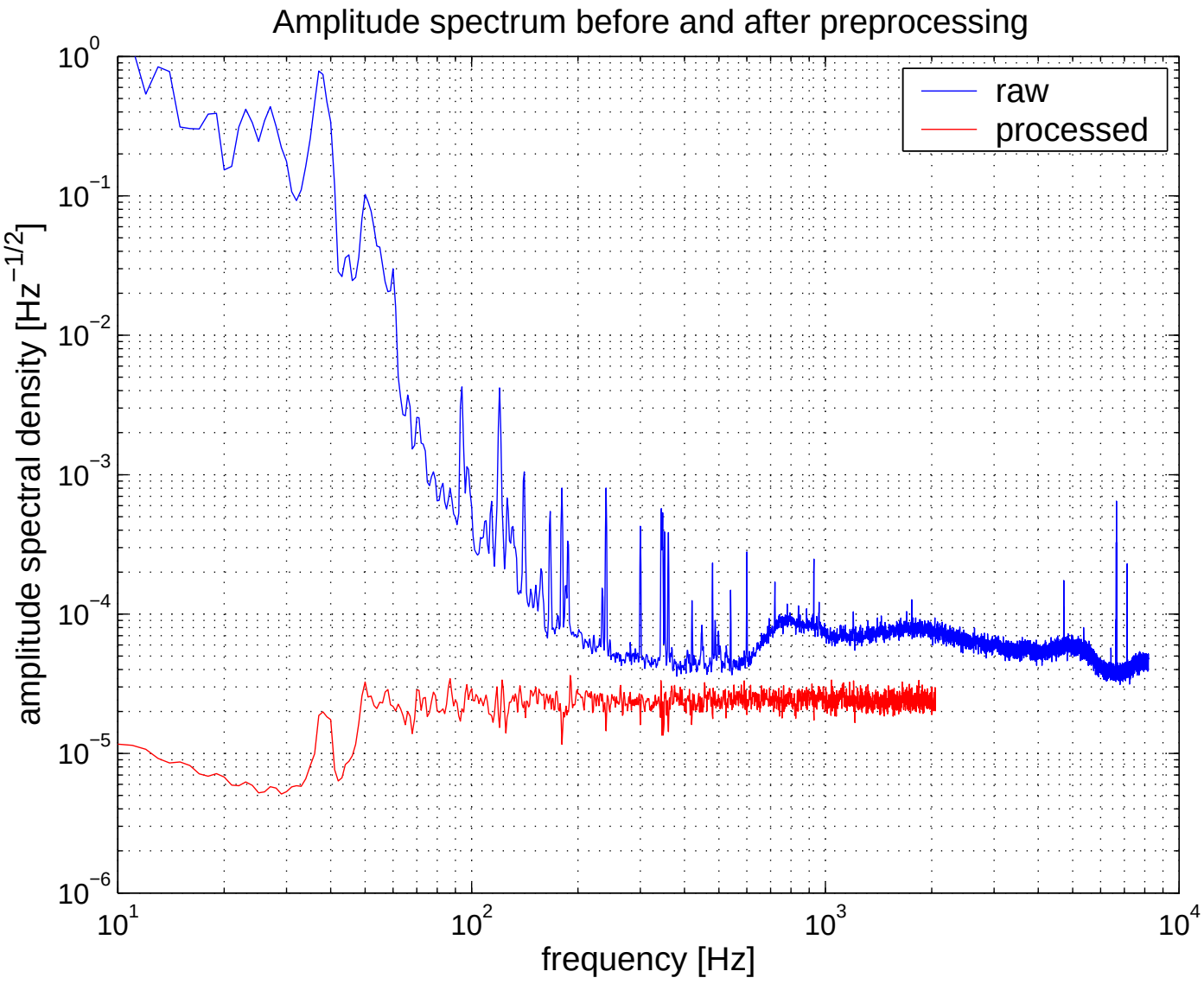
- find line sources
- high pass filter
- notch out line sources
- low pass filter and downsample
- train and apply linear predictor error filter
- high pass filter
- remove transients

All filters are applied as second-order sections model using zero-phase filtering.

Conditioning Example I



Conditioning Example II



LDAS Filter Library

- Filter coefficients in LDAS ILWD format
- Mirrored at all LDAS sites *and burst CVS repository*
- Filters: NOTCH, HPF, and LPEF
- Parameters: filter order, cutoff frequency, bandwidth, lock segment, etc.
- Poor man's adaptive filter: LIGOTOOls *segID*
- Contributions welcome: LDAS *stage data* command
- http://ligo.mit.edu/~shourov/ilwd/filters/S2/

Filter Library Example

From *burstprod.tcl*:

```
...
set segID [ exec segID ${SR} ${ifo} ${stime} ]
...
set filterfiles "# pre-whitening filters for ${SR}:
  $filroot/${SR}/${SR}_HPF_6_100_a.ilwd,push,a1
  $filroot/${SR}/${SR}_HPF_6_100_b.ilwd,push,b1
  $filroot/${SR}/${SR}_NOTCH_927.7_100_a.ilwd,push,a3
  $filroot/${SR}/${SR}_NOTCH_927.7_100_b.ilwd,push,b3
  $filroot/${SR}/${SR}_LPEF_${ifo}_${segID}_4_1_256.ilwd,push,b2"
...
set filterit "# high pass, notch, whiten, and detrend:
   gw = linfilt(b1, a1, gw);
   gw = linfilt(b2, gw);
   gw = linfilt(b3, a3, gw);
   gwf = meandetrend(gw); "
...
```

LIGO

Under Construction

- Recursive least squares (RLS) algorithm
- LPEFMon DMT glitch Monitor
- ETG tuning and pipeline testing
- Migrate Matlab code to DMT/LAL