LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Stefanie Donner

Geophysical Data Analysis

L08 – Filtering II

Classification

linear ⇔ ~~*non-linear*~~

~~*(output contains frequencies not present in input)*~~

analog ⇔ *digital*

(elect. elements: circuits, resistors, conductors)

(logical components and signal processors)

continuous ⇔ *discrete*

IIR ⇔ *FIR*

(Infinite Impulse Response)

(Finite Impulse Response)

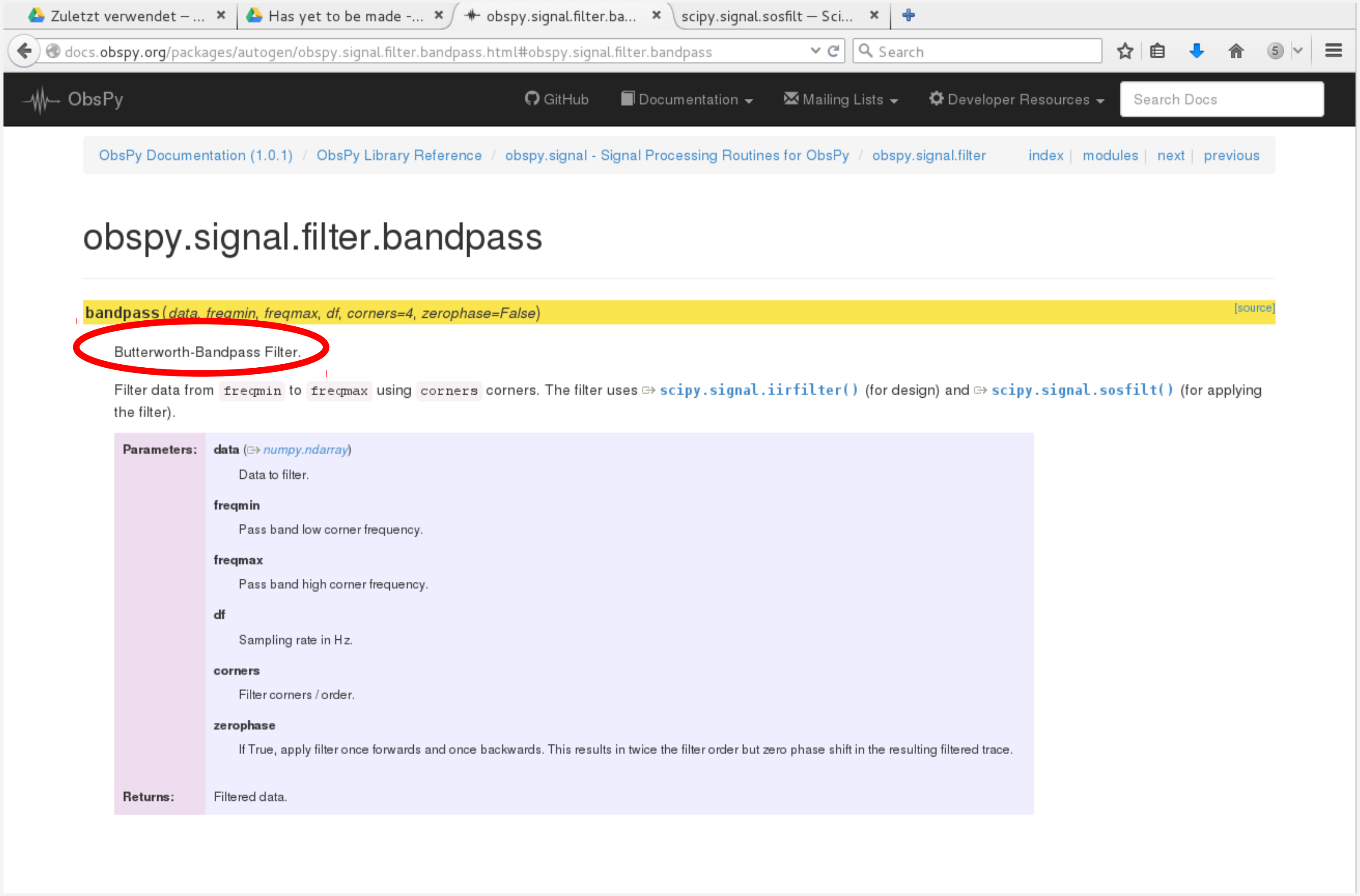
recursive ⇔ *non-recursive*

Examples of filters:

(analog or digital)

- *Butterworth*
- *Chebyshev*
- *Bessel*
- *Wiener (deconvolution)*

The Butterworth filter



Zuletzt verwendet – ... | Has yet to be made -... | obspy.signal.filter.ba... | scipy.signal.sosfilt — Sci...

docs.obspy.org/packages/autogen/obspy.signal.filter.bandpass.html#obspy.signal.filter.bandpass

ObsPy | GitHub | Documentation | Mailing Lists | Developer Resources | Search Docs

ObsPy Documentation (1.0.1) / ObsPy Library Reference / obspy.signal - Signal Processing Routines for ObsPy / obspy.signal.filter | index | modules | next | previous

obspy.signal.filter.bandpass

bandpass(*data, freqmin, freqmax, df, corners=4, zerophase=False*) [source]

Butterworth-Bandpass Filter.

Filter data from `freqmin` to `freqmax` using `corners` corners. The filter uses `scipy.signal.iirfilter()` (for design) and `scipy.signal.sosfilt()` (for applying the filter).

Parameters:

- **data** (*numpy.ndarray*) – Data to filter.
- **freqmin** – Pass band low corner frequency.
- **freqmax** – Pass band high corner frequency.
- **df** – Sampling rate in Hz.
- **corners** – Filter corners / order.
- **zerophase** – If True, apply filter once forwards and once backwards. This results in twice the filter order but zero phase shift in the resulting filtered trace.

Returns: Filtered data.

The Butterworth filter

… designed to have as flat a frequency response as possible within the passband.

e.g. gain of a ***low- pass filter*** *with a cut-off frequency at 1 Hz:*

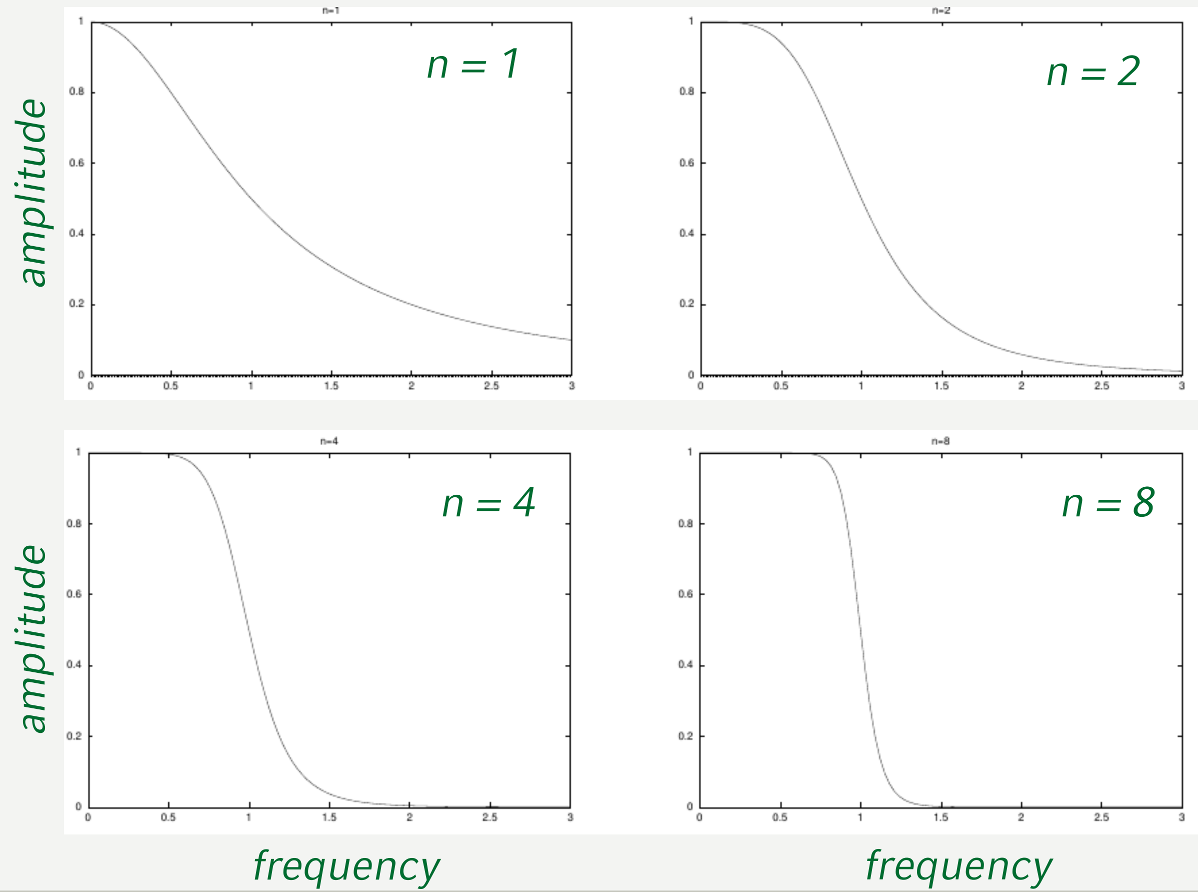
with n the number of poles (i.e. order of filter)

$$G(\omega) = |T(j\omega)| = \sqrt{\frac{1}{1+\omega^{2n}}}$$

with angular frequency $\omega = 1$ *Hz*

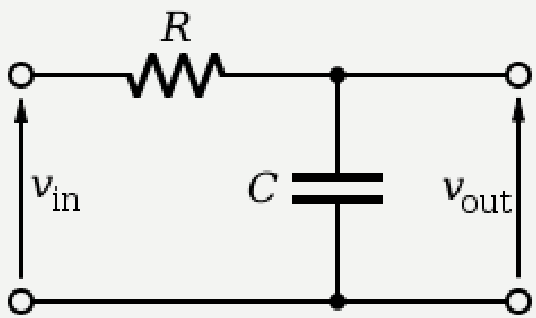
$$G(\omega) = 1/\sqrt{2} = 0.707$$

- *standard FFT filter*
- *very smooth response*
- *no ripples in pass-band*



The Butterworth filter

Remeber the RC filter?



$$RC\dot{y}(t) + y(t) - x(t) = 0$$

$$\frac{A_0}{A_i} = \frac{1}{RCj\omega+1} = T(j\omega)$$

$\Rightarrow$ *Analog Butterworth filter of order n = 1 !*

The Butterworth filter

Other than low-pass filters can be constructed from this one:

Lowpass:

$$G(\omega) = |T(j\omega)| = \sqrt{\frac{1}{1+\omega^{2n}}}$$

$$G^2(\omega) = |T(j\omega)|^2 = T(j\omega)T^*(j\omega) = \frac{1}{1+\omega^{2n}} = \frac{1}{1+(\frac{\omega}{\omega_c})^{2n}}$$

Highpass:

$$T_h(j\omega) = 1 - T_l(j\omega) \Longrightarrow T(j\omega) = \frac{RCj\omega}{\sqrt{1+(RCj\omega)^2}}$$

(for RC filter, n=1)

Bandpass: shifting transfer function along frequency axis to centre it around ω_b

$$|T_b(j\omega)|^2 = \frac{1}{1+[(\omega-\omega_b)/\omega_c]^{2n}}$$

Poles & zeros (very roughly)

Laplace and z-transform are two mathematical tools to break the impulse response into sinusoids and decaying amplitude

$$L\left[f(t)\right] = \int_{-\infty}^{\infty} f(t)e^{-st}dt = F(s)$$

$$s = \sigma + j\omega$$

$$Z\{x[n]\} = \sum_{n=-\infty}^{\infty} x[n]z^{-n} = X(z)$$

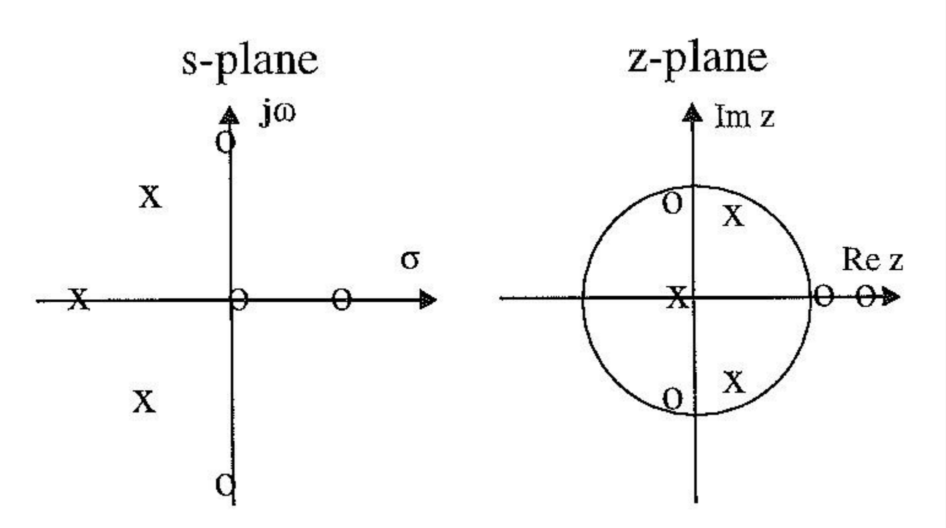
z = continuous complex variable

i.e. expressing the system's characteristics as one complex polynomial divided by another complex polynomial

$$T(j\omega) = \frac{a_0+a_1(j\omega)+a_2(j\omega)^2+\ldots}{b_0+b_1(j\omega)+b_2(j\omega)^2+\ldots}$$

***Zeros**: roots of numerator*

***Poles**: roots of denumerator*



The Butterworth filter

Zuletzt verwendet – ... × | Has yet to be made -... × | obspy.signal.filter.ba... × | scipy.signal.sosfilt — Sci... ×

docs.obspy.org/packages/autogen/obspy.signal.filter.bandpass.html#obspy.signal.filter.bandpass

ObsPy | GitHub | Documentation | Mailing Lists | Developer Resources | Search Docs

ObsPy Documentation (1.0.1) / ObsPy Library Reference / obspy.signal - Signal Processing Routines for ObsPy / obspy.signal.filter | index | modules | next | previous

obspy.signal.filter.bandpass

bandpass(*data, freqmin, freqmax, df, corners=4, zerophase=False*) [source]

Butterworth-Bandpass Filter.

Filter data from `freqmin` to `freqmax` using `corners` corners. The filter uses `scipy.signal.iirfilter()` (for design) and `scipy.signal.sosfilt()` (for applying the filter).

Parameters:	**data** (*numpy.ndarray*) Data to filter. **freqmin** Pass band low corner frequency. **freqmax** Pass band high corner frequency. **df** Sampling rate in Hz. **corners** Filter corners / order. **zerophase** If True, apply filter once forwards and once backwards. This results in twice the filter order but zero phase shift in the resulting filtered trace.
Returns:	Filtered data.

Recursive filters

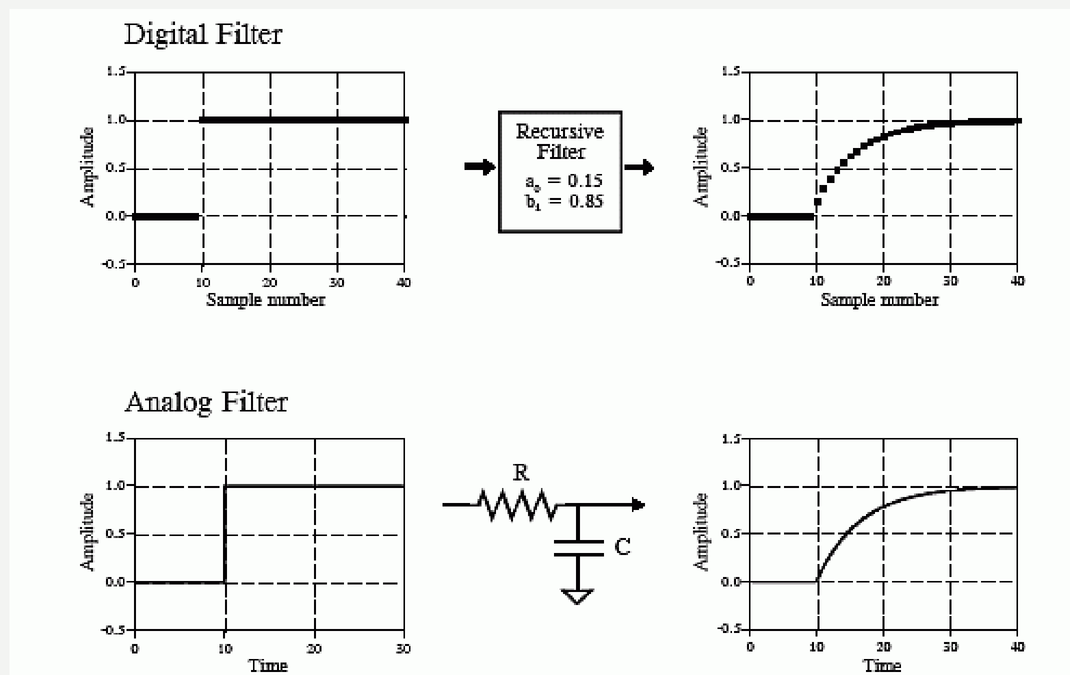
… filter, which re-uses one or more of its outputs as input (feedback system)

usually results in an **unending impulse response** (but not always!!!)

Recursive filters convolve the input signal with a very long fitler kernel. They *bypass* a longer convolution.

$$
\begin{aligned}
y[n] = {} & a_0 x[n] + a_1 x[n-1] + a_2 x[n-2] + a_3 x[n-3] + \cdots \\
& + b_1 y[n-1] + b_2 y[n-2] + b_3 y[n-3] + \cdots
\end{aligned}
$$

… they can mimik analog filters …
(1^{st} order lowpass filter)



IIR – Infinite Impulse Response

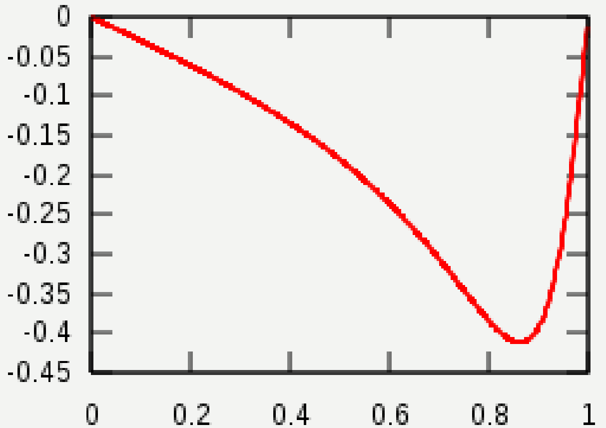
With recursive filters we can create filters with infinitely long impulse responses.
Almost all analog electronic filters are IIR filters.

difference equation (for derivation of transfer function):

$$y[n] = \frac{1}{a_0}\left(\sum_{i=0}^{P} b_i\, x[n-i] - \sum_{j=1}^{Q} a_j\, y[n-j]\right)$$

- *difficult to implement but easy construction*
- *require less computing effort than FIR filters (because it needs lower orders)*
- *problems with instability at higher orders (due to recursive character - “feedback”)*
- *non-linear phase*
- *time-invariant and causal*

non-linear phase



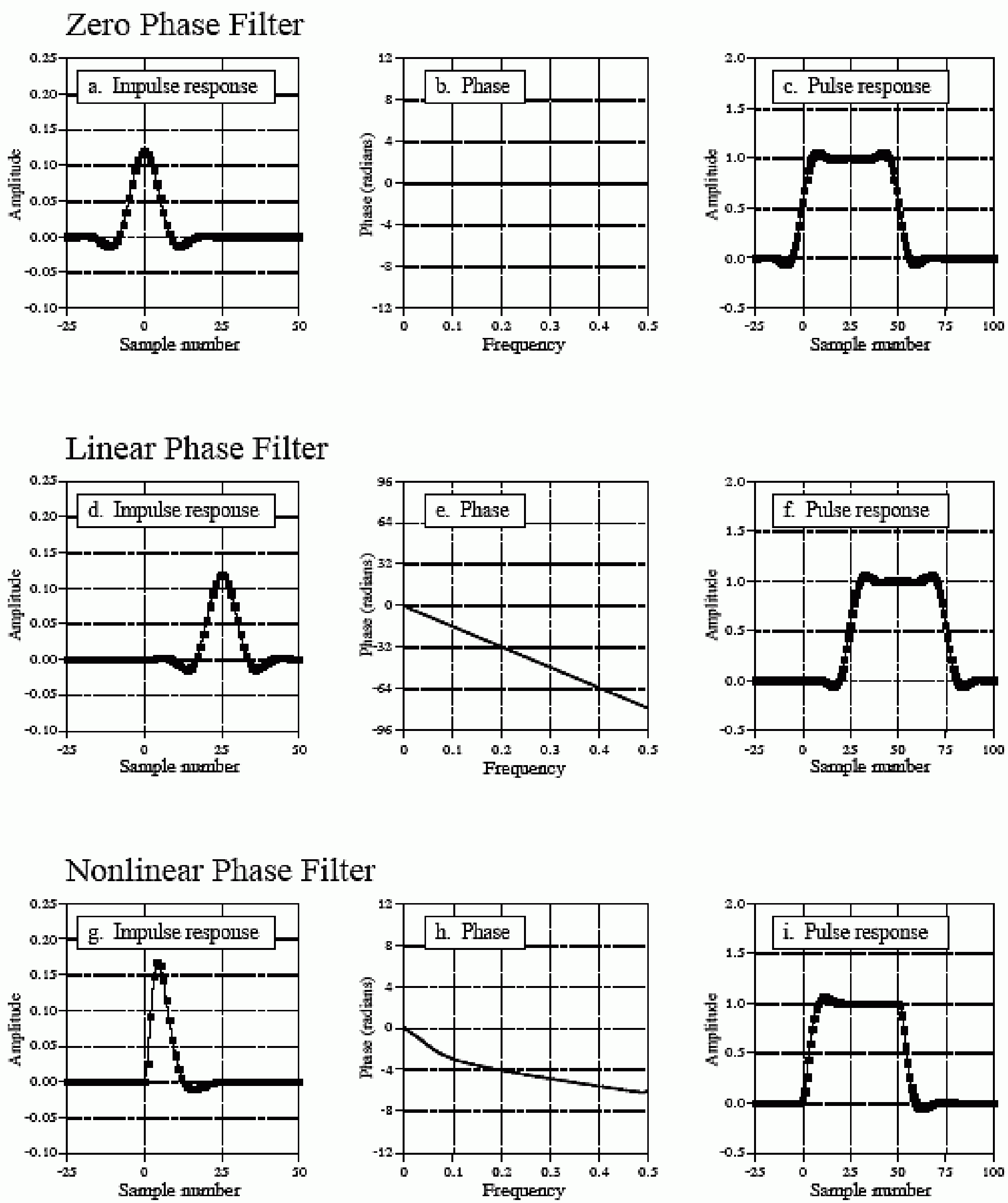
IIR filter - phase

What happens to the signal, when the filter has a non-linear phase?

Pulse response =

pos. step response + neg. step response visualises what happens to rising and falling edge in signal

How to solve?



FIR – Finite Impulse Response

FIR filters are non-recursive filters, i.e. the output only depends on the most recent input values

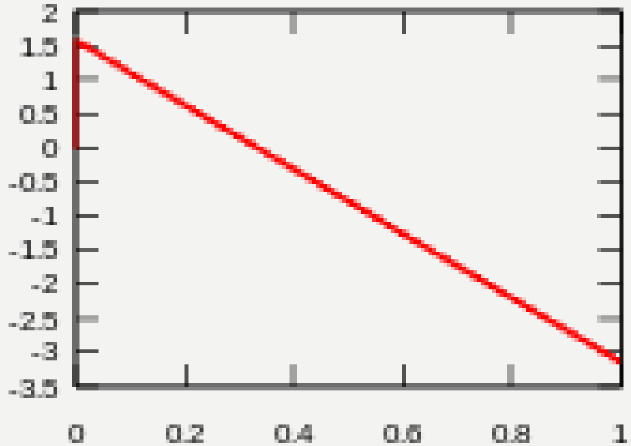
difference equation (for derivation of transfer function):

$$y[n] = b_0\, x[n] + b_1\, x[n-1] + \ldots + b_N\, x[n-N] = \sum_{i=0}^{N} b_i\, x[n-i]$$

- *easy to implement*
- *always stable*
- *easily designed to be linear- or zero-phase*
- *causal or acausal*
- *many coefficients needed for steep filter*

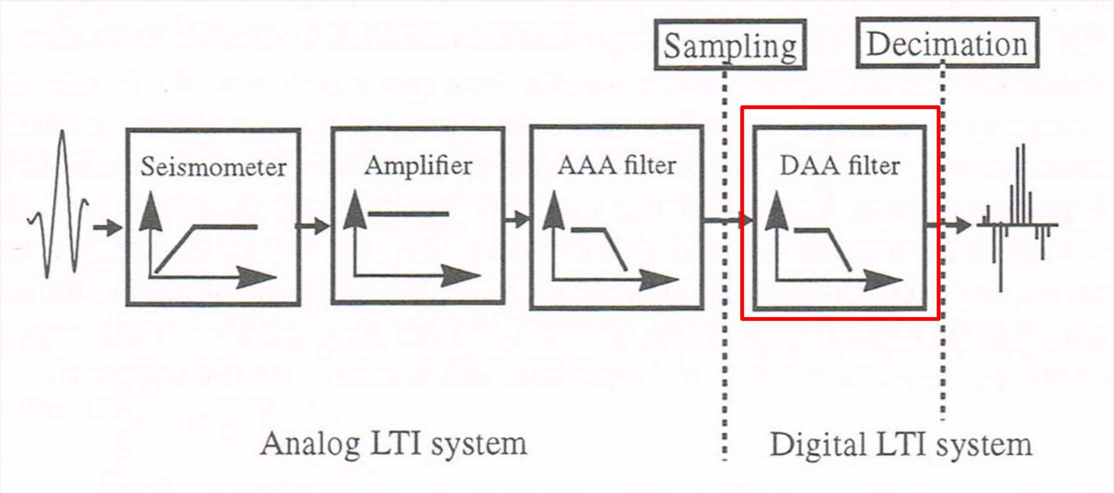
⇨ *not appropriate for FFT filtering,*
too slow & memory intensive,
no real-time processing possible

linear phase shift



FIR – Finite Impulse Response

Application: as digital anti-alias filter in modern seismic acquisition systems



Because:

easy to match given design specifications with great accuracy
very steep and stable filters possible
linear-phase property can be implemented exactly

Chebyshev filter

https://docs.obspy.org/packages/autogen/obspy.signal.filter.html

ObsPy GitHub Documentation Mailing Lists Developer Resources Search Docs

ObsPy Documentation (1.0.1) / ObsPy Library Reference / obspy.signal - Signal Processing Routines for ObsPy   index | modules | next | previous

obspy.signal.filter

Various Seismogram Filtering Functions

copyright:	The ObsPy Development Team (devs@obspy.org)
license:	GNU Lesser General Public License, Version 3 (https://www.gnu.org/copyleft/lesser.html)

Public Functions

bandpass	Butterworth-Bandpass Filter.
bandstop	Butterworth-Bandstop Filter.
envelope	Envelope of a function.
highpass	Butterworth-Highpass Filter.
integer_decimation	Downsampling by applying a simple integer decimation.
lowpass	Butterworth-Lowpass Filter.
lowpass_cheby_2	Cheby2-Lowpass Filter
lowpass_fir	FIR-Lowpass Filter. (experimental)
remez_fir	The minimax optimal bandpass using Remez algorithm. (experimental)

Chebyshev filter

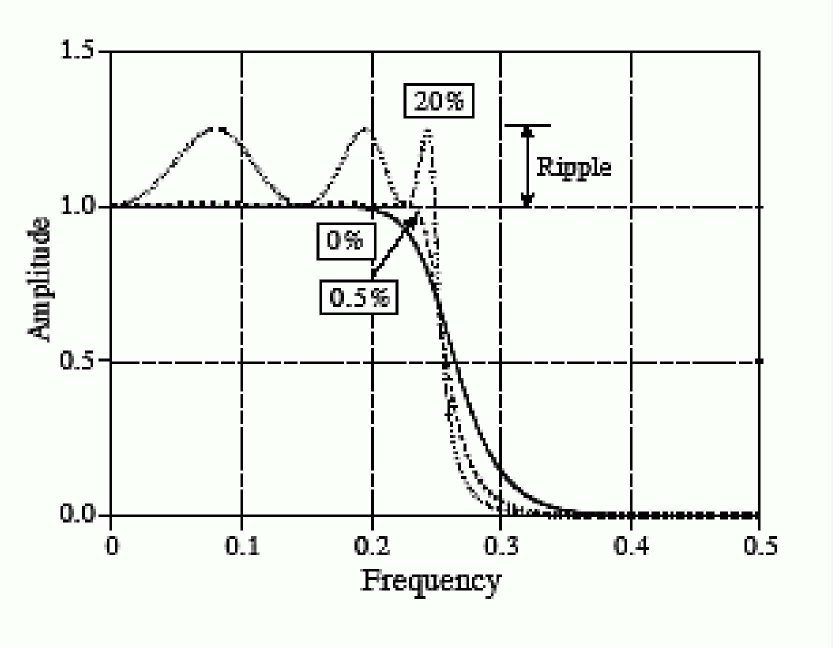
... steeper roll-off and more ripples than Butterworth

- *Very fast (recursion rather than convolution)*
- *Mathematical characteristic derived from Chebyshev polynomials*
- *Ripples either in pass- (Chebyshev I) or stop-band (Chebyshev II)*

$$G_n(\omega) = |T_n(j\omega)| = \frac{1}{\sqrt{1 + \epsilon^2 \cdot P_n^2(\frac{\omega}{\omega_0})}}$$

with ϵ as the ripple factor (good choice: 0.5%)

Used where the frequency content of the signal is more important than having a constant amplitude.



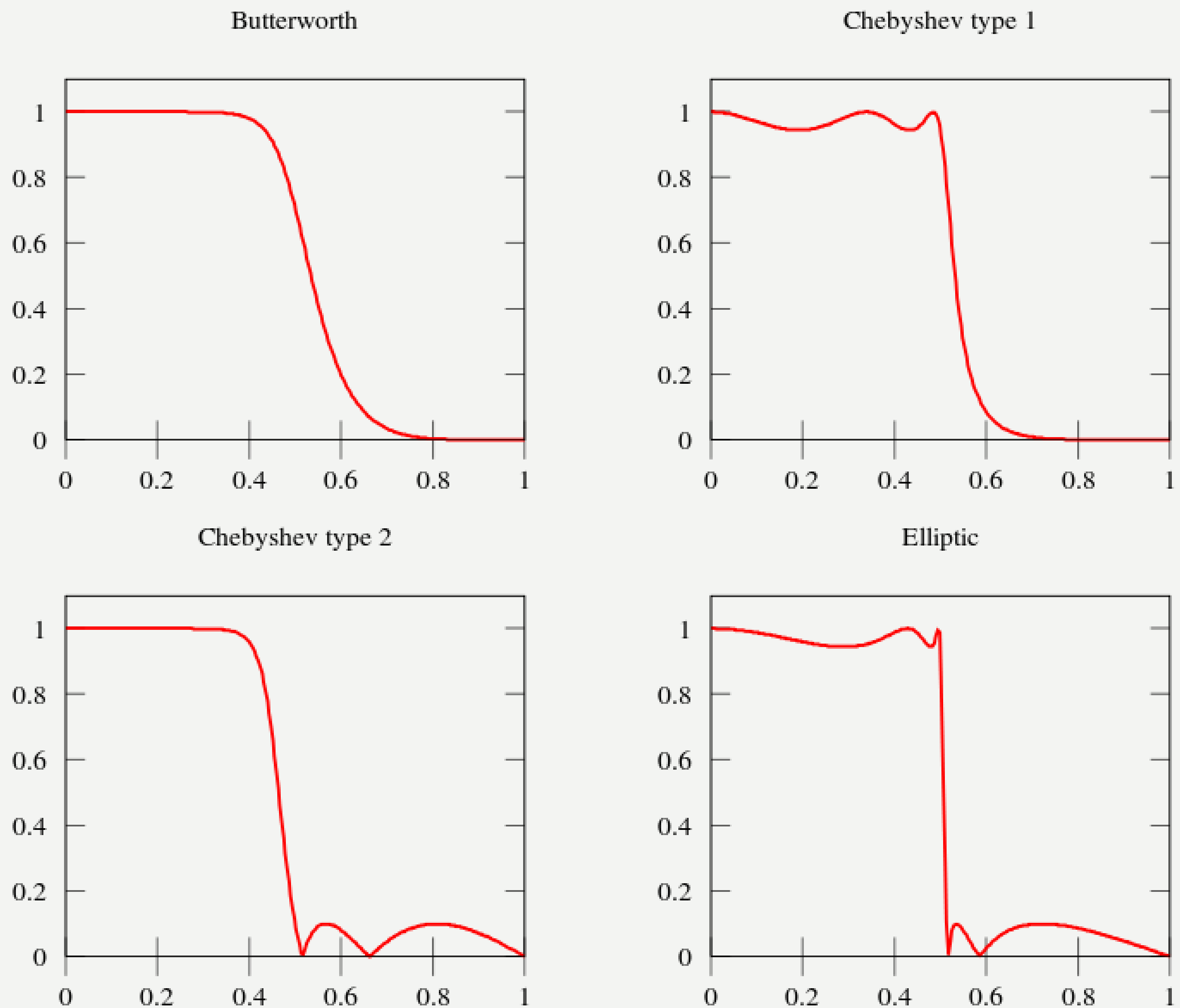
more ripples (bad)
→ steeper roll-off (good)

Comparison

- *Amplitude response functions*
- *All fifth-order filters*

Butterworth: poles on a "circle"

Chebyshev: poles on an "ellipse"



scypi signal processing

docs.scipy.org/doc/scipy/reference/signal.html

SciPy.org Sponsored By ENTHOUGHT

Scipy.org | Docs | SciPy v0.17.1 Reference Guide — index | modules | modules | next | previous

Signal processing (scipy.signal)

Convolution

convolve(in1, in2[, mode])	Convolve two N-dimensional arrays.
correlate(in1, in2[, mode])	Cross-correlate two N-dimensional arrays.
fftconvolve(in1, in2[, mode])	Convolve two N-dimensional arrays using FFT.
convolve2d(in1, in2[, mode, boundary, fillvalue])	Convolve two 2-dimensional arrays.
correlate2d(in1, in2[, mode, boundary, ...])	Cross-correlate two 2-dimensional arrays.
sepfir2d((input, hrow, hcol) -> output)	Description:

B-splines

bspline(x, n)	B-spline basis function of order n.
cubic(x)	A cubic B-spline.
quadratic(x)	A quadratic B-spline.
gauss_spline(x, n)	Gaussian approximation to B-spline basis function of order n.
cspline1d(signal[, lamb])	Compute cubic spline coefficients for rank-1 array.
qspline1d(signal[, lamb])	Compute quadratic spline coefficients for rank-1 array.
cspline2d((input {, lambda, precision}) -> ck)	Description:
qspline2d((input {, lambda, precision}) -> qk)	Description:
cspline1d_eval(cj, newx[, dx, x0])	Evaluate a spline at the new set of points.
qspline1d_eval(cj, newx[, dx, x0])	Evaluate a quadratic spline at the new set of points.
spline_filter(Iin[, lmbda])	Smoothing spline (cubic) filtering of a rank-2 array.

Filtering

order_filter(a, domain, rank)	Perform an order filter on an N-dimensional array.
medfilt(volume[, kernel_size])	Perform a median filter on an N-dimensional array.
medfilt2d(input[, kernel_size])	Median filter a 2-dimensional array.
wiener(im[, mysize, noise])	Perform a Wiener filter on an N-dimensional array.
symiirorder1((input, c0, z1 {, ...)	Implement a smoothing IIR filter with mirror-symmetric boundary conditions using a cascade of first-order sections.
symiirorder2((input, r, omega {, ...)	Implement a smoothing IIR filter with mirror-symmetric boundary conditions using a cascade of second-order sections.

Table Of Contents

Previous topic

scipy.optimize.LbfgsInvHessProduct

Next topic

scipy.signal.convolve

Wiener filter

Konzept: $y(t) = h(t) \cdot x(t) + n(t)$

noise

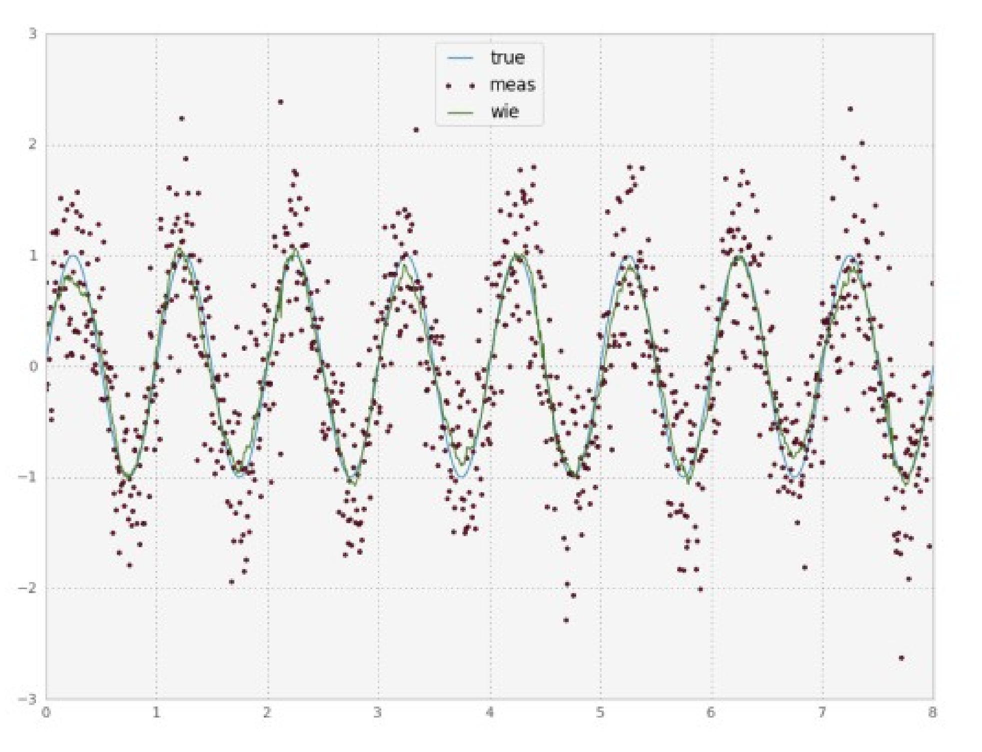
measured signal

impulse response

unknown original signal

Look for g(t), such that: $x(t) \approx \hat{x}(t) = g(t) \cdot y(t)$

$$\curvearrowright \quad G(\omega) = \frac{H^*(\omega)X(\omega)}{|H(\omega)|^2 X(\omega) + N(\omega)}$$



Wiener filter

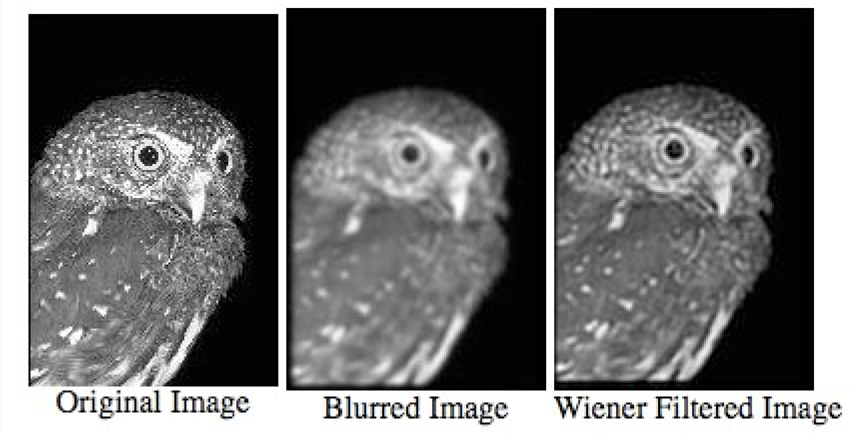
*… used for **deconvolution** …*

Rewriting the equation:

$$G(\omega) = \frac{1}{H(\omega)}\left[\frac{|H(\omega)|^2}{|H(\omega)|^2 + \frac{N(\omega)}{X(\omega)}}\right]$$

- *[ ] = 1 for noise-free data*
  → simple inverse filter

- *If SNR decreases → [ ] decreases*
  → frequencies are damped with dependance on SNR



However, we need to know the SNR (or at least have a good estimate).

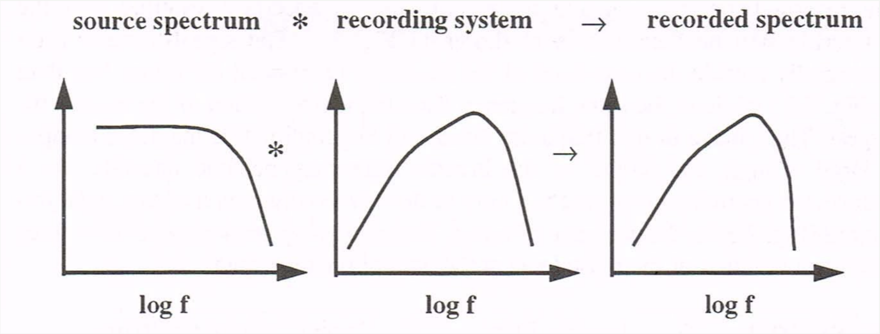
So far, questions to ....

or ...

Inverse filters

- *Recording systems act as a filter on the data*

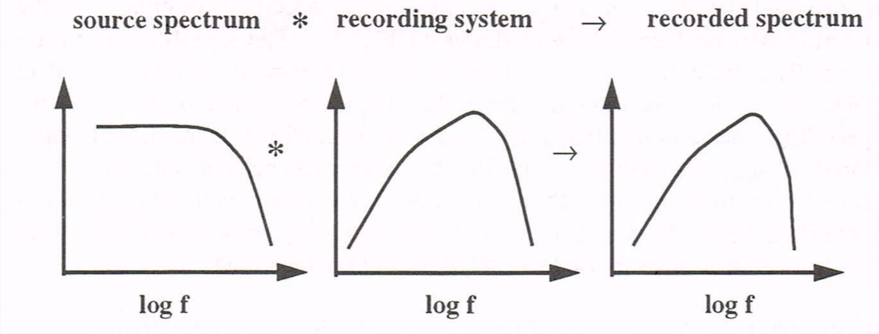
$$X(\omega) \cdot T(j\omega) = Y(\omega)$$



Inverse filters

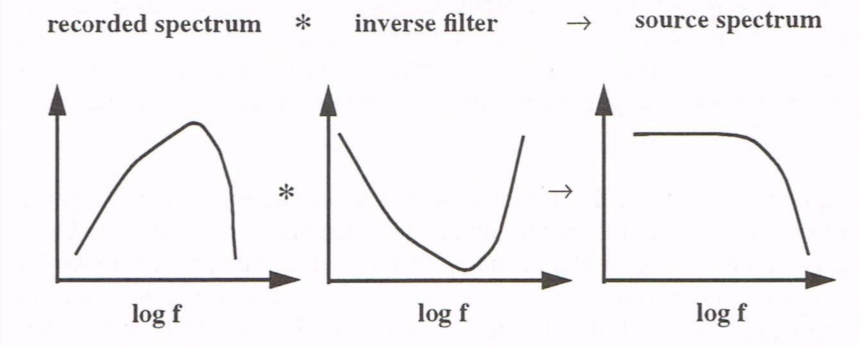
- *Recording systems act as a filter on the data*

$$X(\omega) \cdot T(j\omega) = Y(\omega)$$



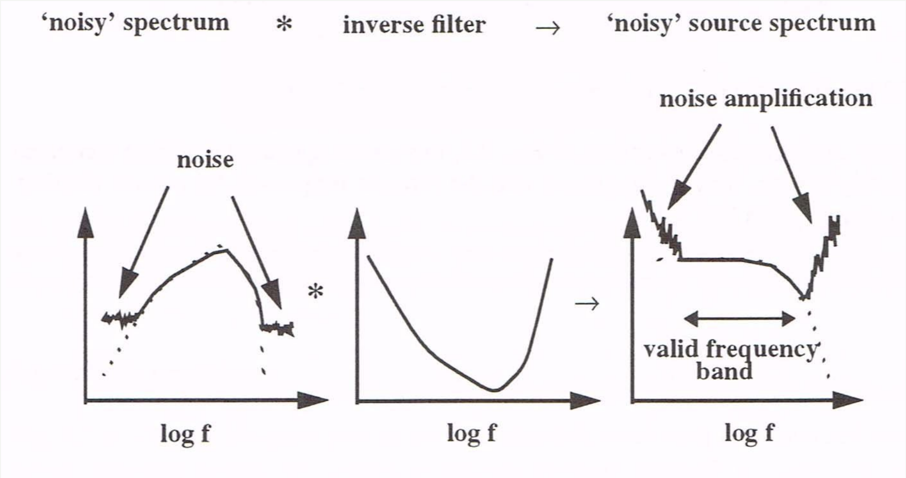
- *Before analysis this filter needs to be reversed*
  - restitution
  - deconvolution
  - removing the instrument response or instrument characteristic

$$Y(\omega) \cdot \frac{1}{T(j\omega)} = X(\omega)$$



Inverse filter problems

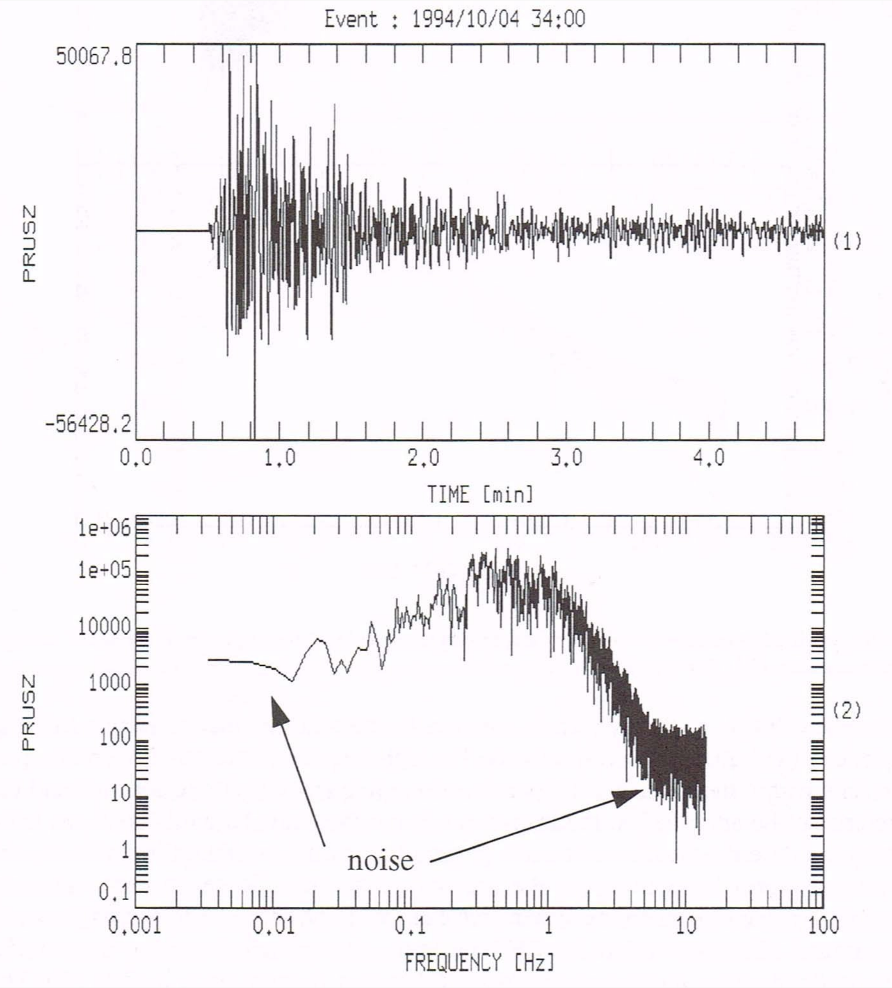
Noise in the data can get strongly amplified during the restitution.



Solution:

→ water-level method

→ apply bandpass afterwards



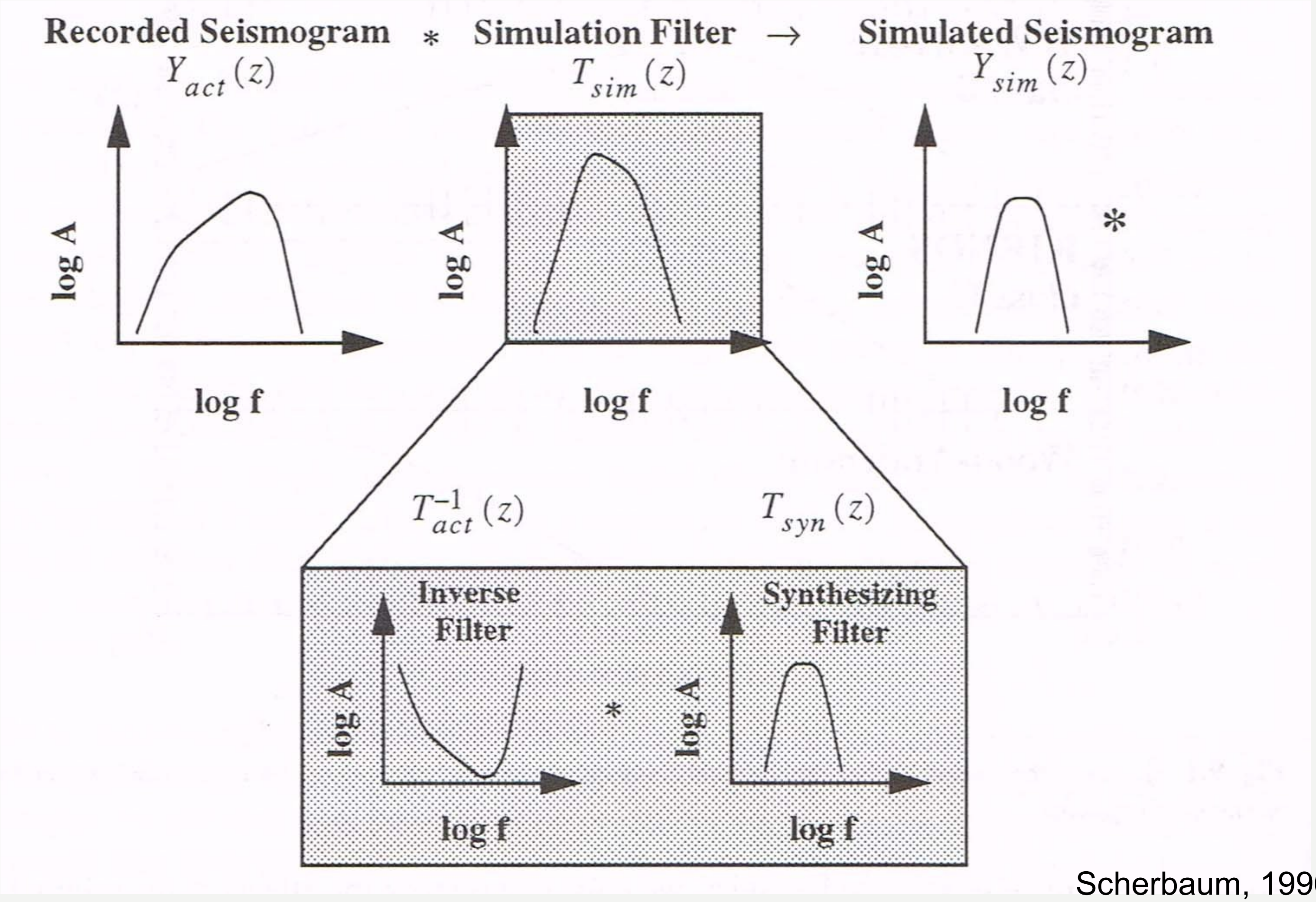
Simulation filter

- *Simulation filters are used to modify a signal as if recorded under certain conditions*
  - *Seismogram recorded with a special instrument*
  - *A song played in a special surrounding (e.g. concert hall with special echoing)*
  - *Simulate site effects on the signal, e.g. influence of wind, overlapping second source, ...*
- *Simulation filters can be ANY filter (depending on what effect is wanted)*

No single solution for set-up

Definition

- *Simulation is always a two step process, closely connected to inverse filters*



Scherbaum, 1996

Definition

- *Simulation is always a two step process, closely connected to inverse filters*

$$Y_{sim}(z) = \frac{T_{syn}(z)}{T_{act}(z)} \cdot Y_{act}(z)$$

$$= T_{sim}(z) \cdot Y_{act}(z)$$

- *Tact , Tsyn – transfer function of actual and synthesized recording system, respectively*
- *Yact, Ysim – z-transforms of recorded and simulated seismogram, respectively*

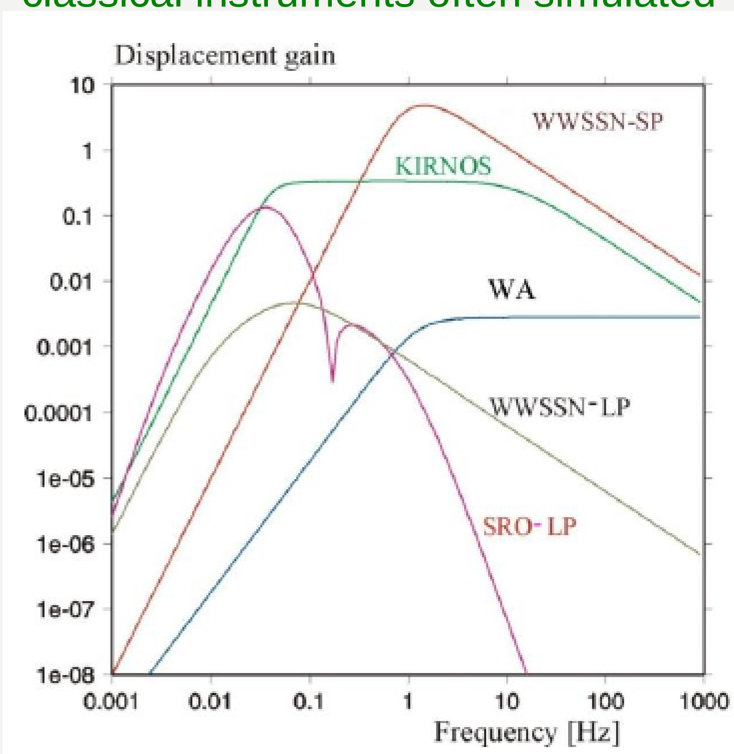
Scherbaum, 1996

Necessities

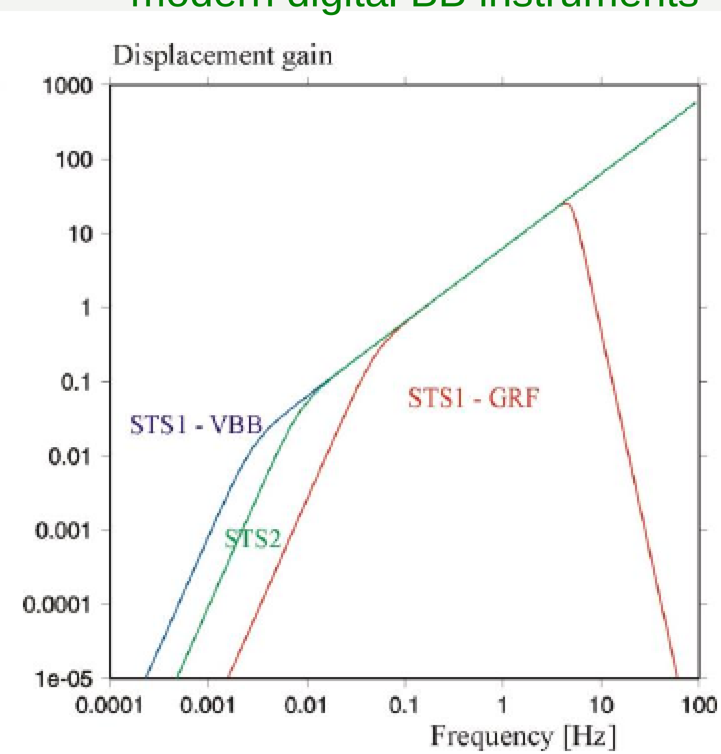
To be able to obtain a high accuracy and stable simulation, the actual instrument used for the recording need to fulfil the following necessities:

- *Large bandwidth*
- *Large dynamic range*
- *High resolution*
- *Low instrumental self-noise*
- *Low noise induced by variations of air pressure+ temperature*
- *Analytically exactly known transfer function*

classical instruments often simulated



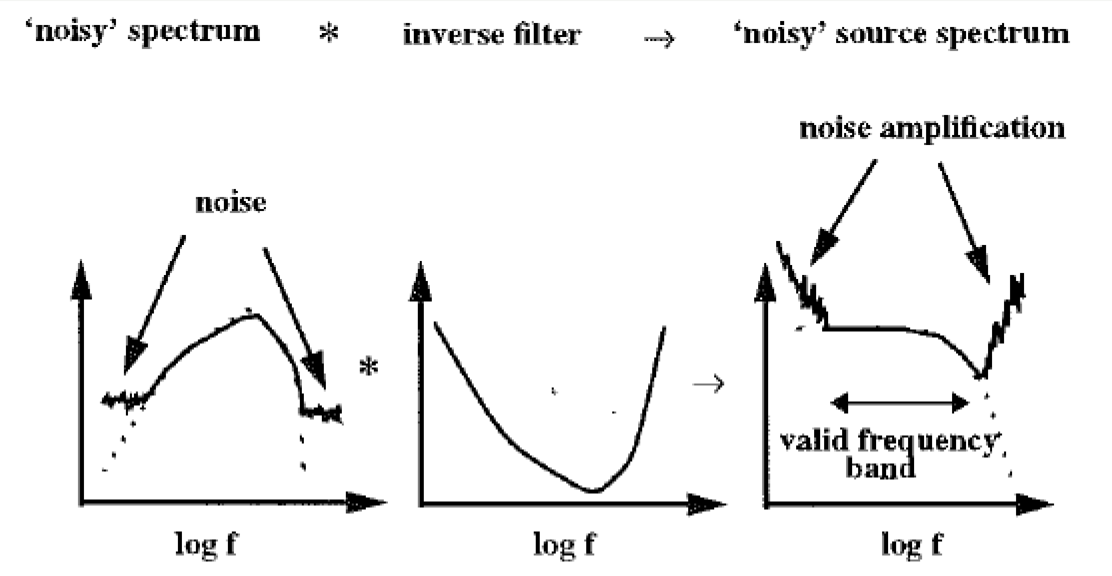
modern digital BB instruments



NMSOP 2013, chap. 11

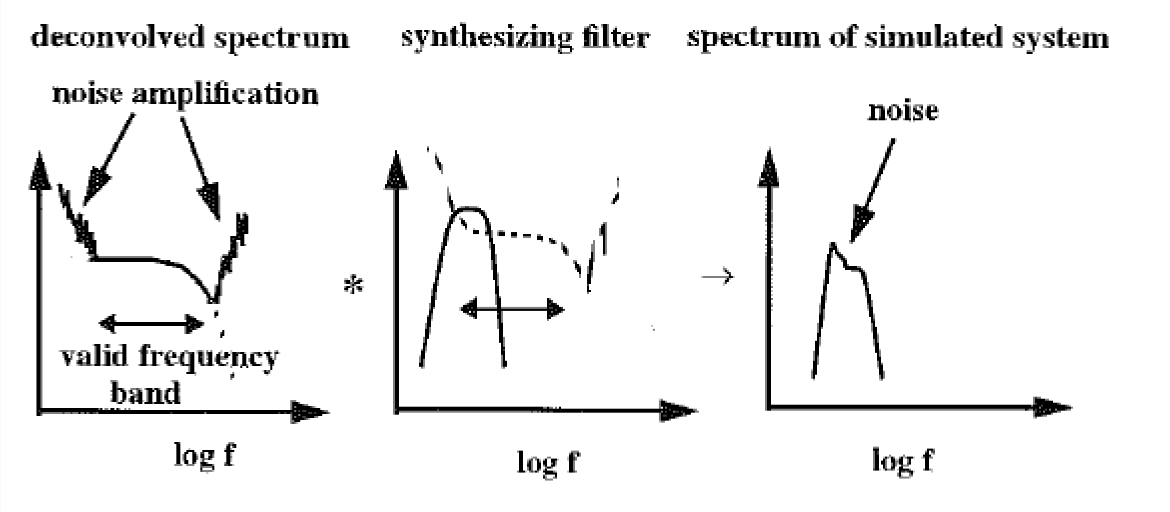
Stability: 1st problem

- *Noise ….*



Possible solutions:

- *bandpass filter final trace*
- *water-level during deconvolution*



Scherbaum, 1996

Stability: 2nd problem

- *Mathematical singularities in the transfer function at s = 0*

$$T^{-1}(s) = \frac{s^2+2h\omega_0 s+w_0^2}{s^3}$$

Possible solution: regularization by subsequent highpass filtering

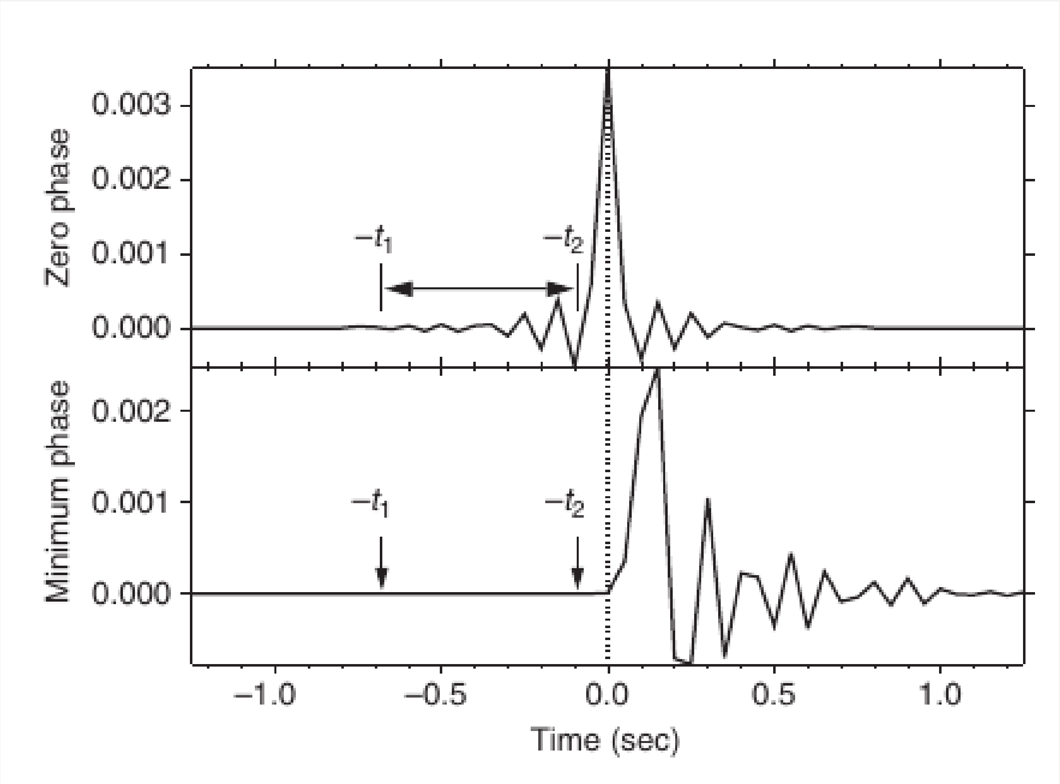
$$T^{-1}(s) \cdot T_{HP}(s) = \frac{s^2+2h\omega_0 s+w_0^2}{s^3} \cdot \frac{s^3}{(s+\epsilon)^3} = \frac{s^2+2h\omega_0 s+w_0^2}{(s+\epsilon)^3}$$

(Remember properties of linear systems ...)

Scherbaum, 1996

Stability: 3rd problem

- *Causality*
  (It basically has something to do with the relationship between the pole and zero position of the transfer function; see Scherbaum, 1996)

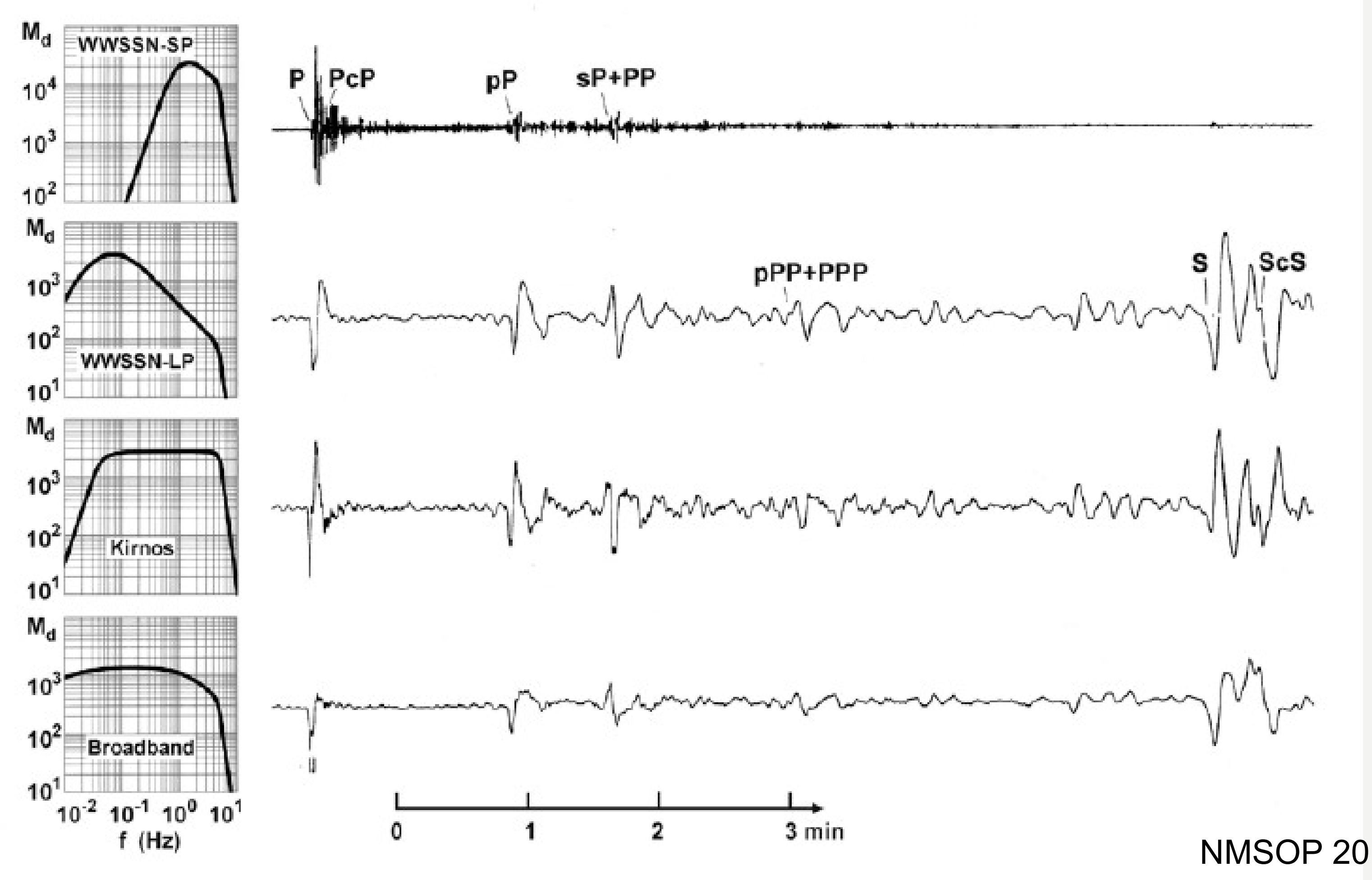


acausal

causal

International Handbook of Earthquake & Engineering Seismology, Part 1, 2002

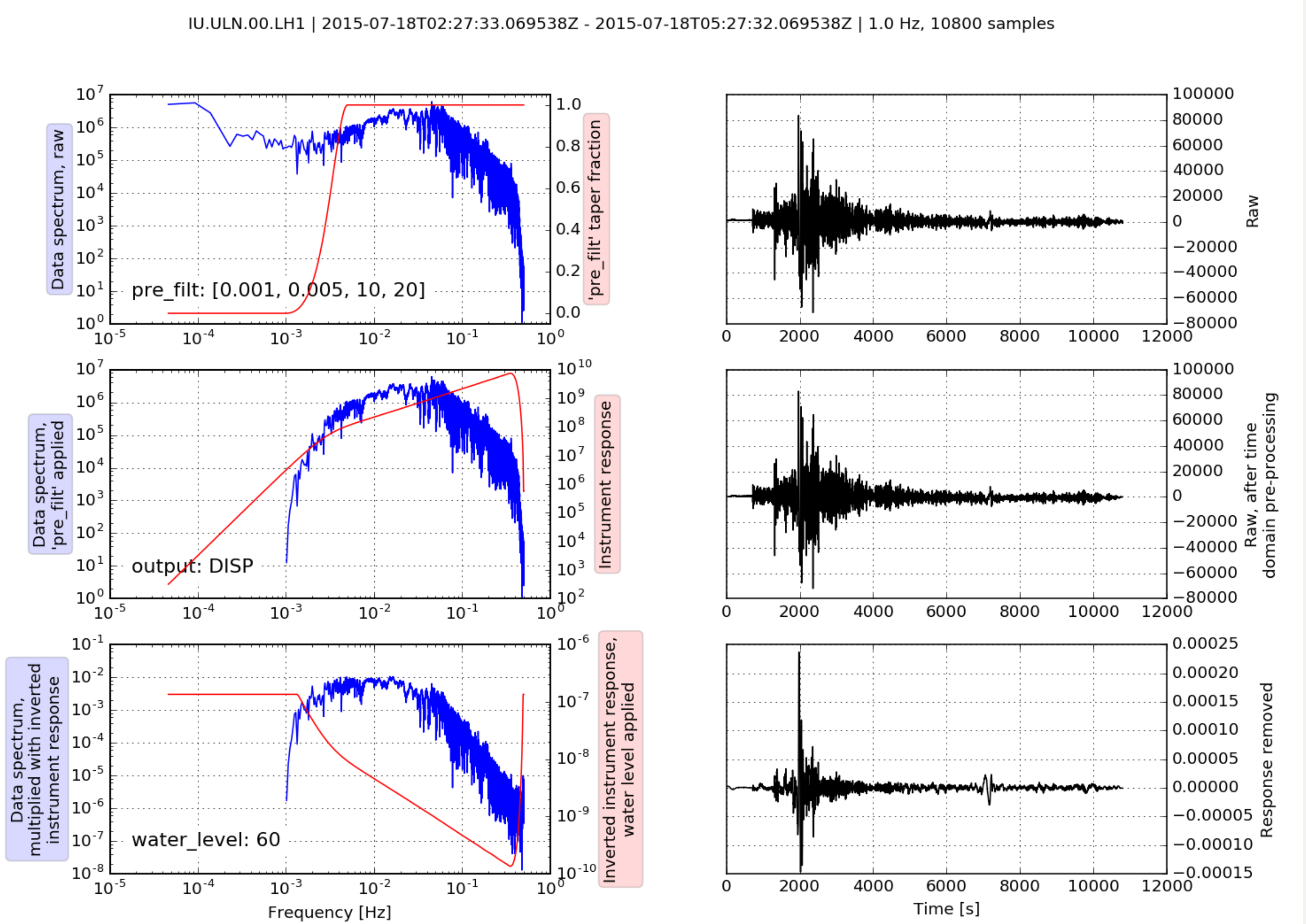
Example 1



NMSOP 2013, chap. 4

- *Deep earthquake (h=570km; d=75°) at Gräfenberg Observatory*
- *Filtered according to response curves of some traditional standard characteristics*
- *Bottom trace: deconvolved seismogram (without instrument characteristic)*

Example 2



ObsPy tutorial, no. 14