filter_designer – An Interactive Tool for Designing Digital Filters

Danny Harvey
Boulder Real Time Technologies, Inc.
Antelope User Group Meeting
ARSO, Slovenian Environment Agency, Ljubljana
2018 May

All digital filtering in Antelope utilizes **time-domain** convolution and recursive methodologies.

Digital time-domain filtering offers significant advantages over FFT based frequency-domain filtering.

1. Can operate on infinite time series in a continuous fashion.
2. Minimal edge effects that can be confined wit finite time windows.
3. Much more computationally efficient.
4. Simplicity of implementation.

Implementation of Fourier transforms is done with th
Discrete Fourier Transform (DFT) using a clever digit
algorithm known as the Fast Fourier Transform (FFT)

All DFTs, regardless of how they are implemented, ar
necessarily computed over finite time windows, usual
no more than thousands of time samples, which cause
them to be subject to an artifact known as
"wraparound".

FFT computational efficiency of order 5 * N * log2(N)
brute force direct time-domain convolution computat
efficiency of order 2 * N * N.

However, most convolutions involve one function (th
filter impulse response) with a reduced and constant
value of N.

All filtering of time sampled waveforms in Antelope are done in the time domain and do not involve the computations of signal spectra using FFTs.

All Antelope digital time domain filters can be applied to arbitrary time series and can be applie to continuous time series of indefinite length.

There are no inherent time windowing parameter needed by the Antelope filters as there would be filtering were done in the frequency domain. No "wraparound" effects.

The Antelope time domain filters are very computationally efficient compared to frequency domain methods

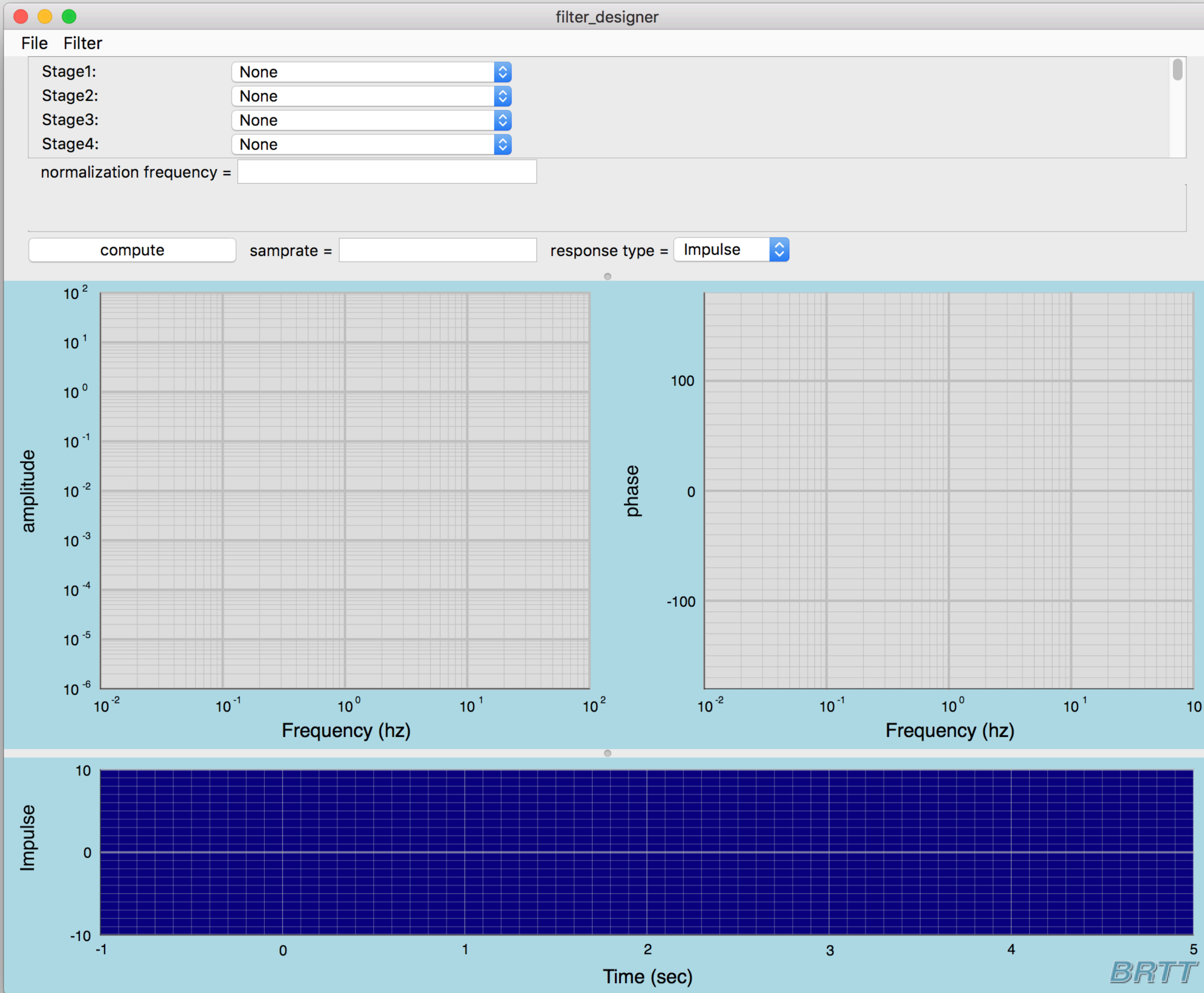
All Antelope time domain filters are implemente
with the **`wffil(3)`** library which provides gener
purpose interfaces to various time domain
waveform filter methods.

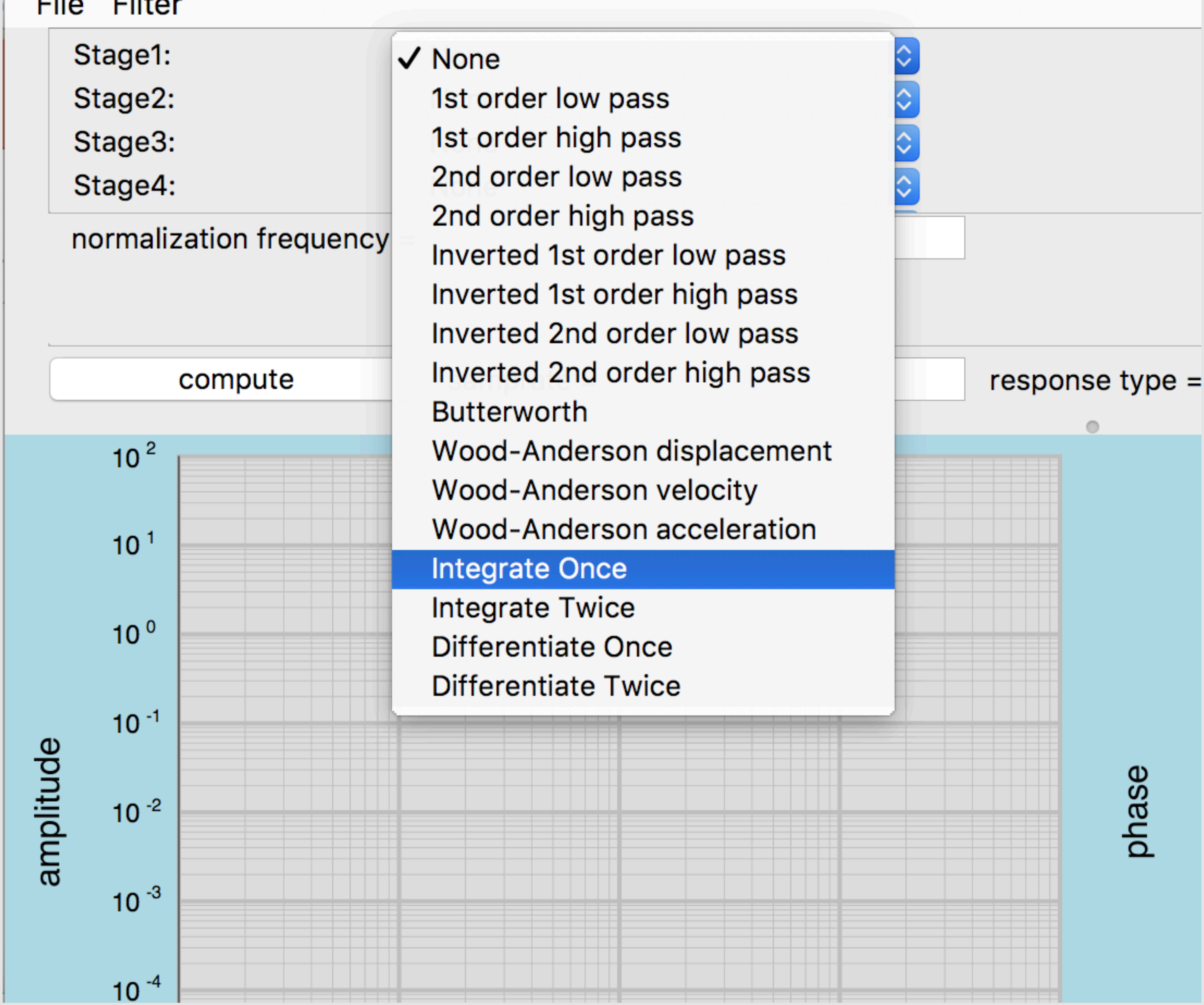
Specific filtering groups are defined in
`wffilbrtt(3)`, which includes Butterworth,
generalized S-domain polynomials, differentiato
integrator, Wood-Anderson instrument response
generalized FIR filters, and **`wffilave(3)`**, whic
provides a variety of averaging filters.

Most filtering in Antelope for the purpose of dat
processing, such as the filters used in **orbdetec**
for example, is done using recursive digital filter
also known as Infinite Impulse Response, or IIR,
filters.

A new application, **filter_designer**, is avail
in the 5.8 release of Antelope. This app provides
the design and visualization of Antelope IIR filte

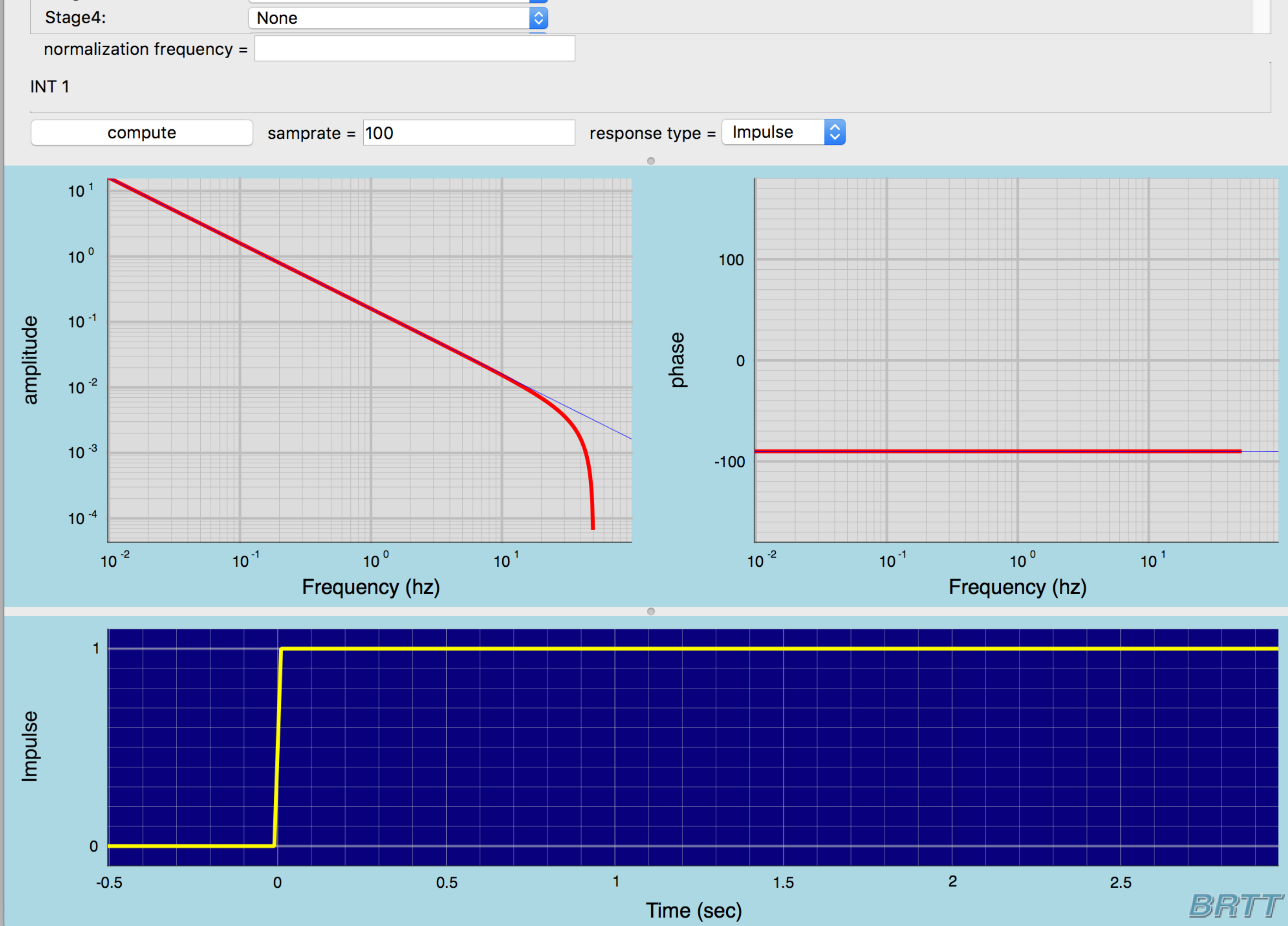
Python script using the new Antelope **pythonbqplot(3Y** python graphics libraries



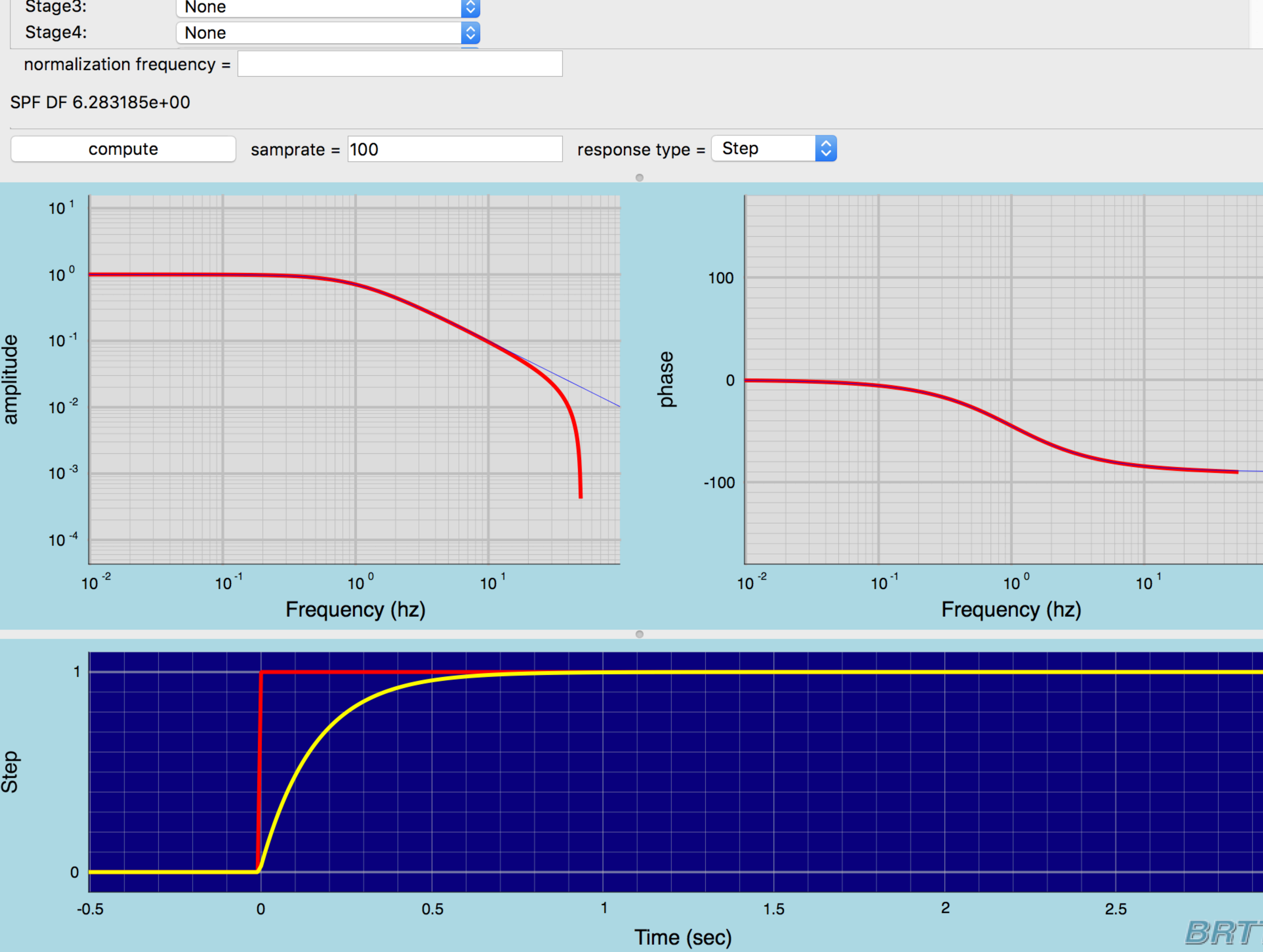

File
Filter
Stage1:
Stage2:
Stage3:
Stage4:
normalization frequency
None
1st order low pass
1st order high pass
2nd order low pass
2nd order high pass
Inverted 1st order low pass
Inverted 1st order high pass
Inverted 2nd order low pass
Inverted 2nd order high pass
Butterworth
Wood-Anderson displacement
Wood-Anderson velocity
Wood-Anderson acceleration
Integrate Once
Integrate Twice
Differentiate Once
Differentiate Twice
compute
response type =
amplitude
phase
10 2
10 1
10 0
10 -1
10 -2
10 -3
10 -4

pe	S-domain transfer function	Antelope filter string	Description
order low pass		DF C	first order denominator polynom suitable as a zero frequency normalized first order low-pass filter
order high pass		DFDIF1 C	first order denominator polynom with a single differentiation suitable as an infinite frequency normalized first order high-pass filter
d order low pass		DS B C	second order denominator polynomial suitable as a zero frequency normalized second order low-pass filter
d order high pass		DSDIF2 B C	second order denominator polynomial with a double differentiation suitable as an infinite frequency normalized second order high-pass filter
.			

Filter stages are defined in **wffilbrtt(3)**



The red line is the digital Z-domain response. The thin blu line is the analog response. Note the effects of the frequenc warping.


Stage3: None
Stage4: None
normalization frequency =
SPF DF 6.283185e+00
compute
samprate = 100
response type = Step
amplitude
10^{1}
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-2} 10^{-1} 10^{0} 10^{1}
Frequency (hz)
phase
100
0
-100
10^{-2} 10^{-1} 10^{0} 10^{1}
Frequency (hz)
Step
1
0
-0.5 0 0.5 1 1.5 2 2.5
Time (sec)

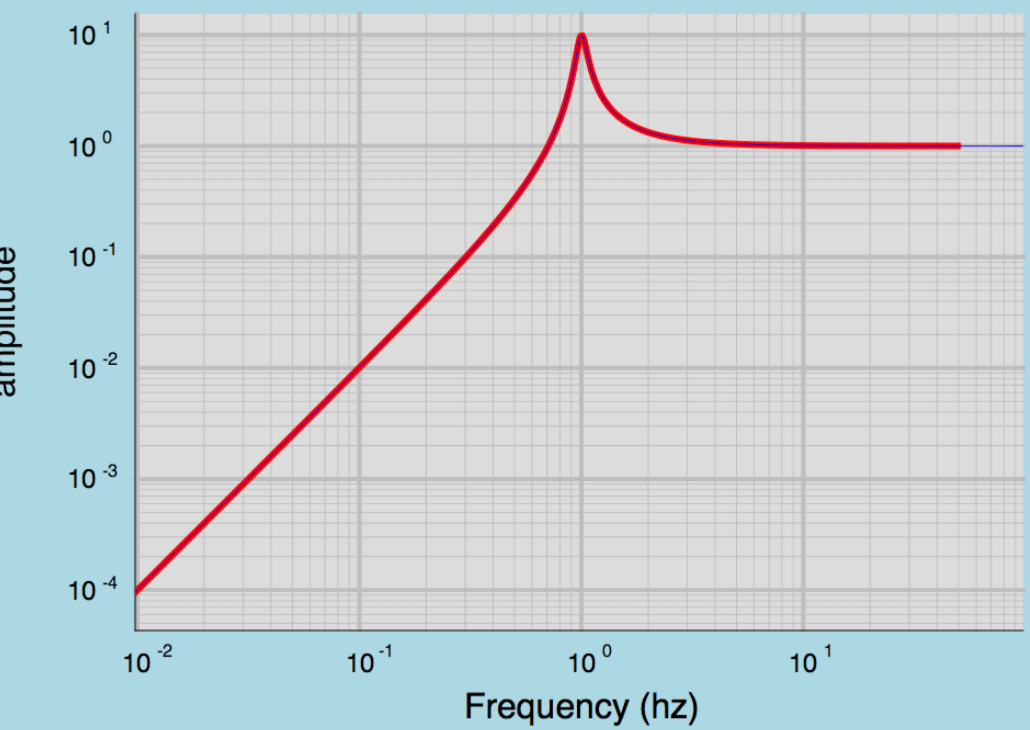
Stage3: None

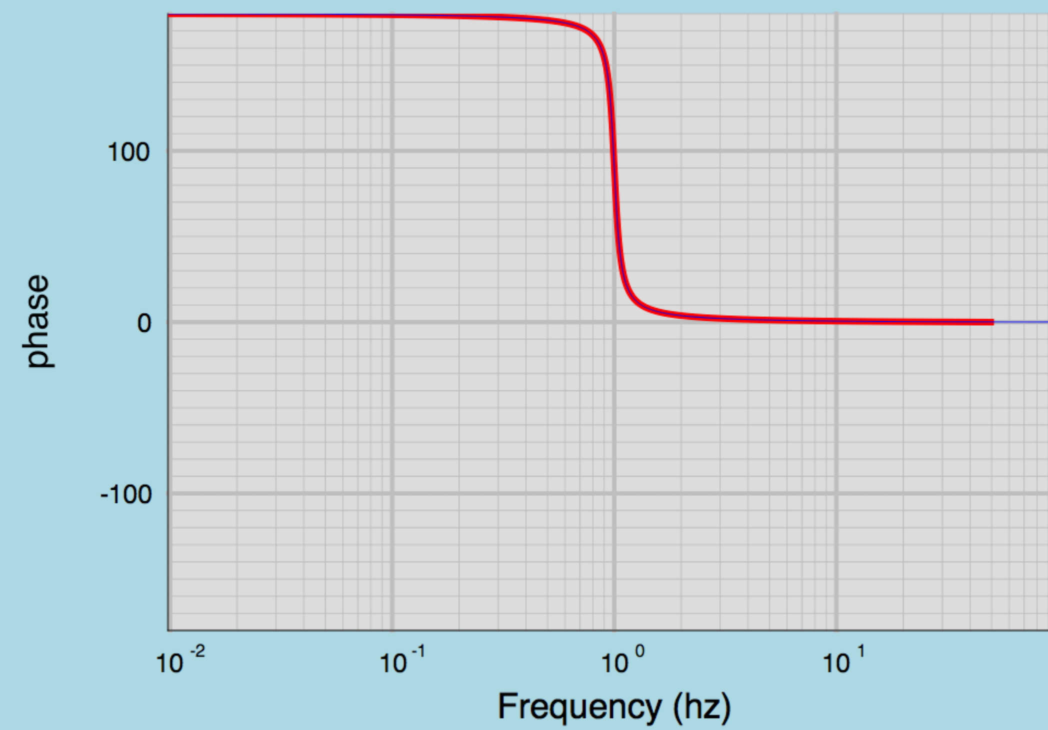
Stage4: None

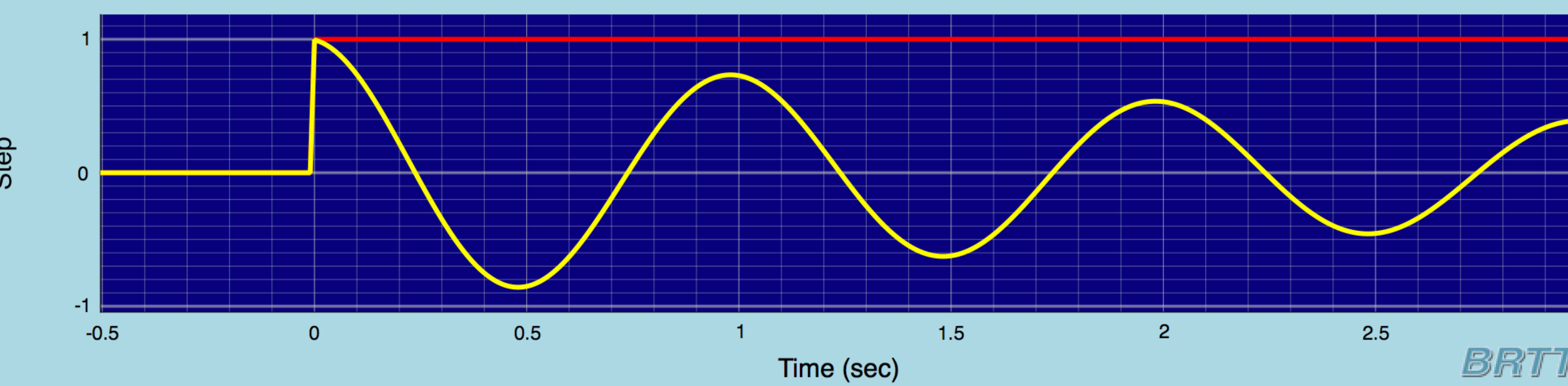
normalization frequency =

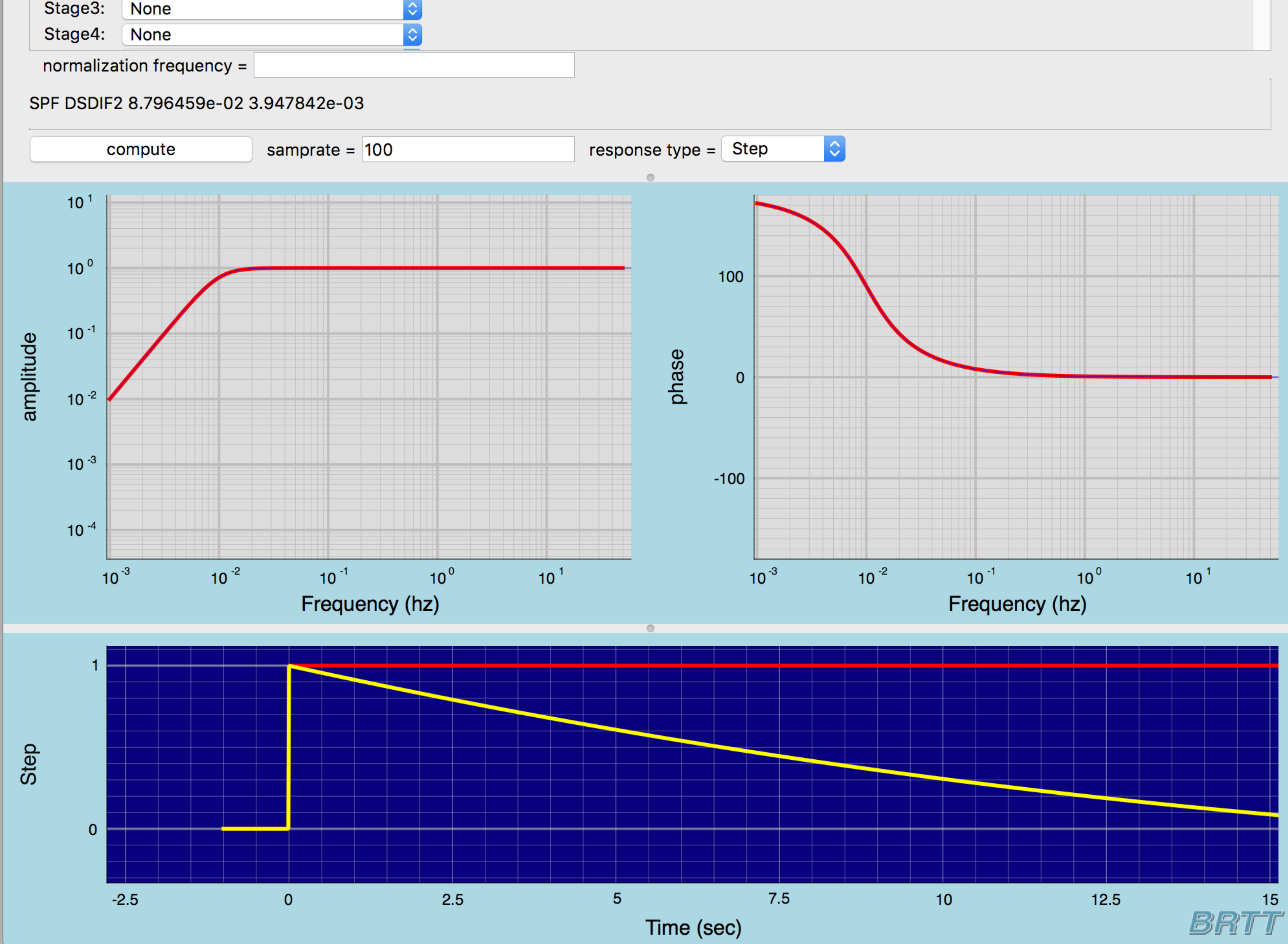
SPF DSDIF2 6.283185e-01 3.947842e+01

compute   samprate = 100   response type = Step

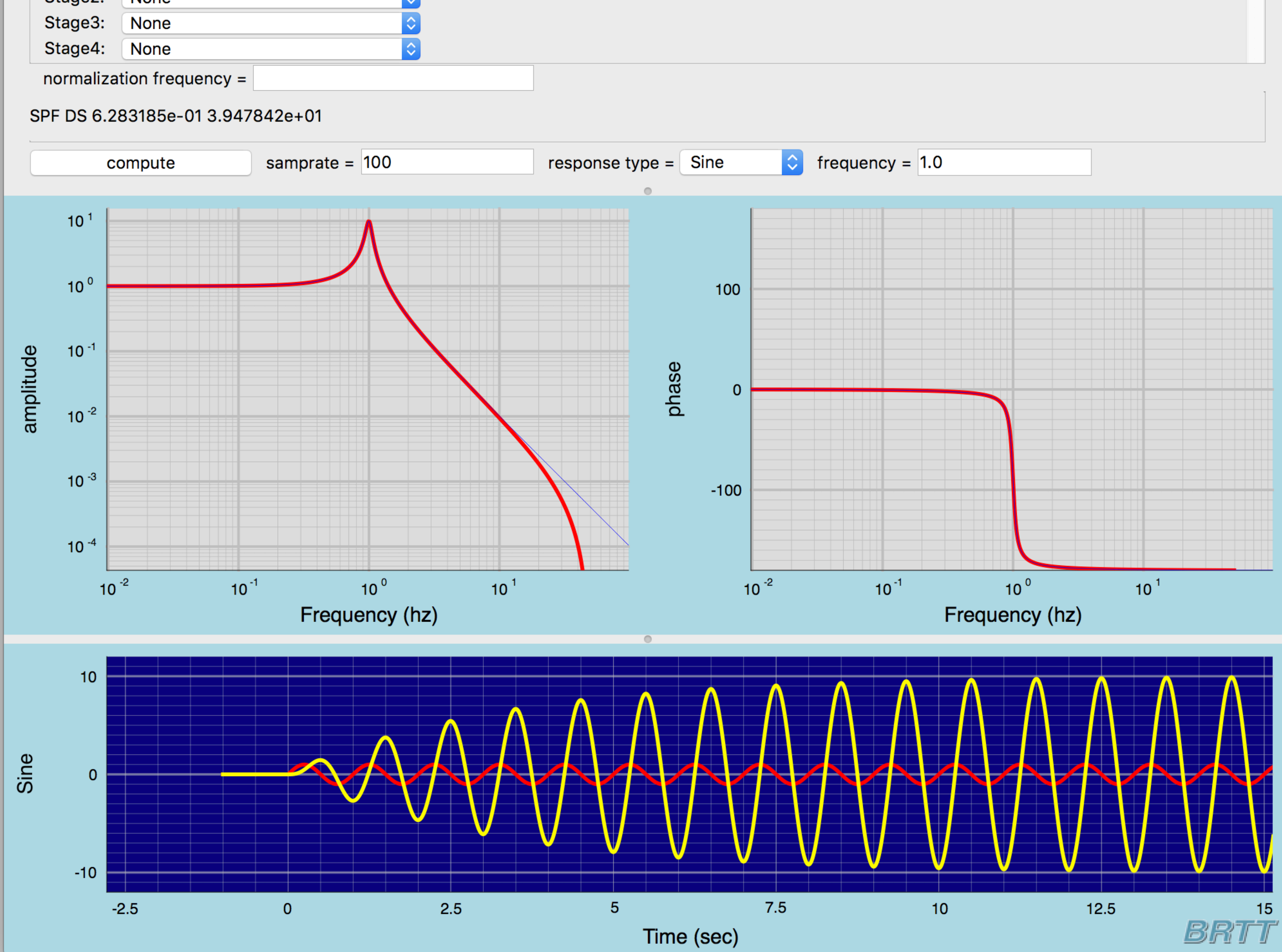






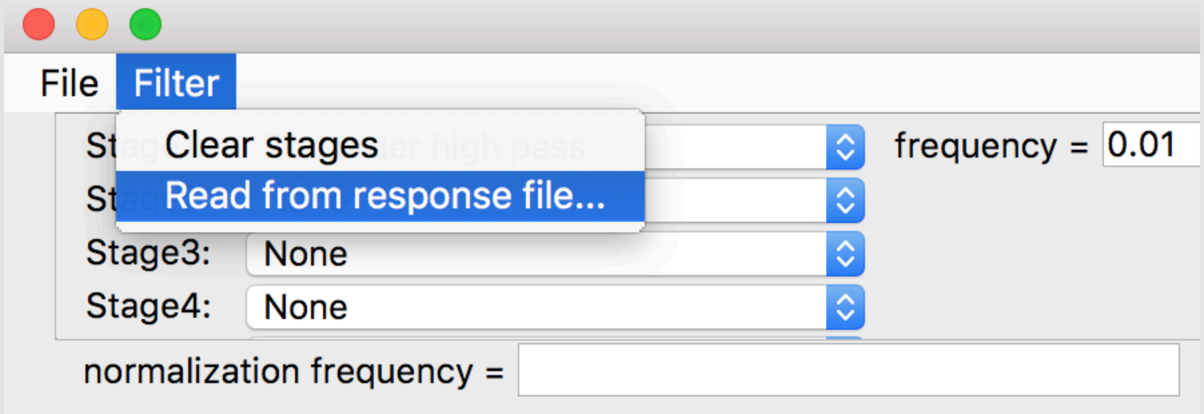


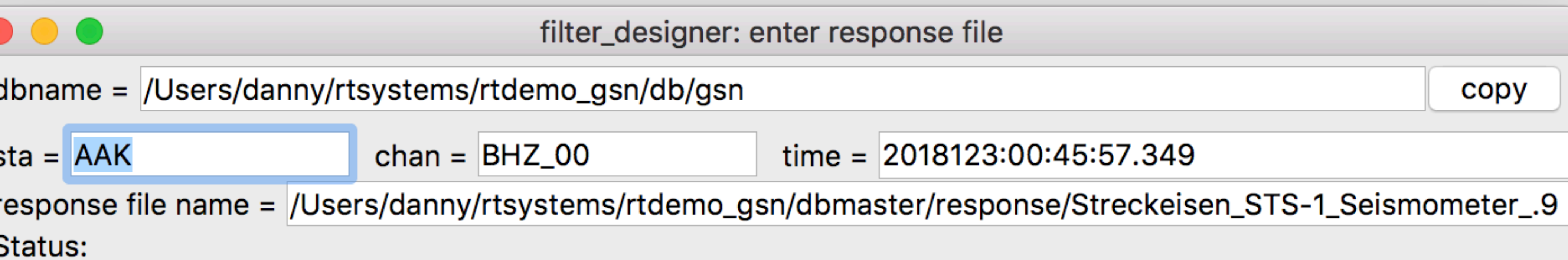
Basic seismometer response (note the filter string



Strong motion response function.

Grab real instrument responses





3: Inverted 1st order low pass | frequency = 0.0242718000002

4: 2nd order low pass | frequency = 0.0243926703301 | damping = 0.996000014423

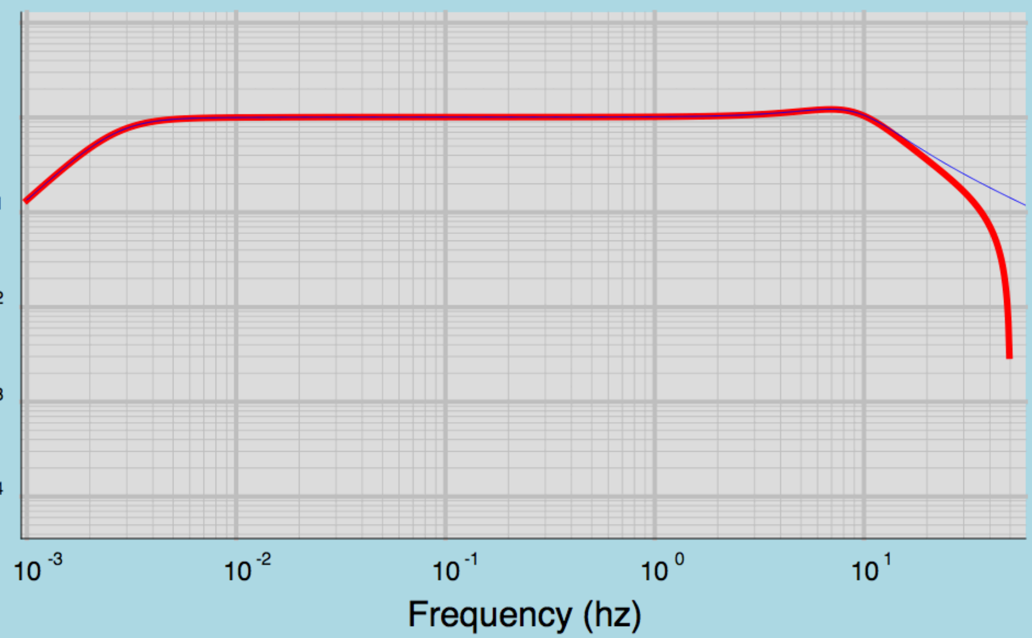
5: 2nd order low pass | frequency = 9.17499977475 | damping = 0.55499946866

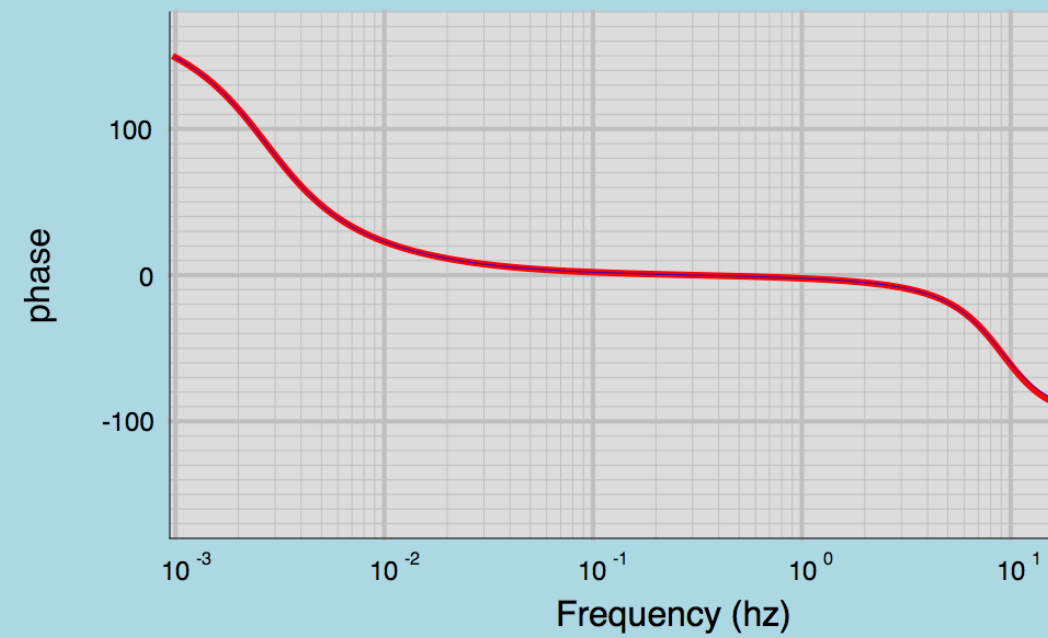
6: Inverted 1st order low pass | frequency = 12.5

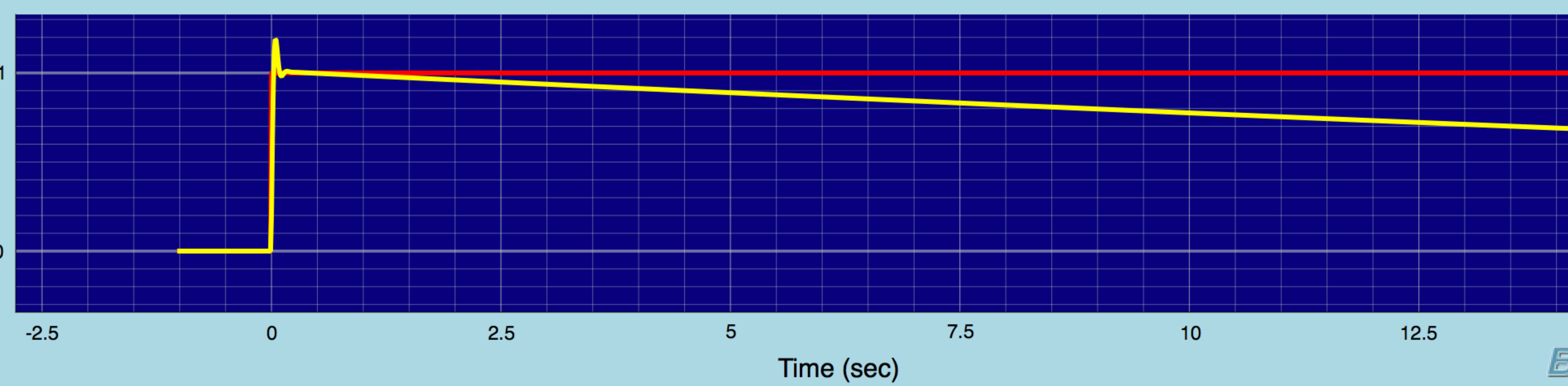
lization frequency =

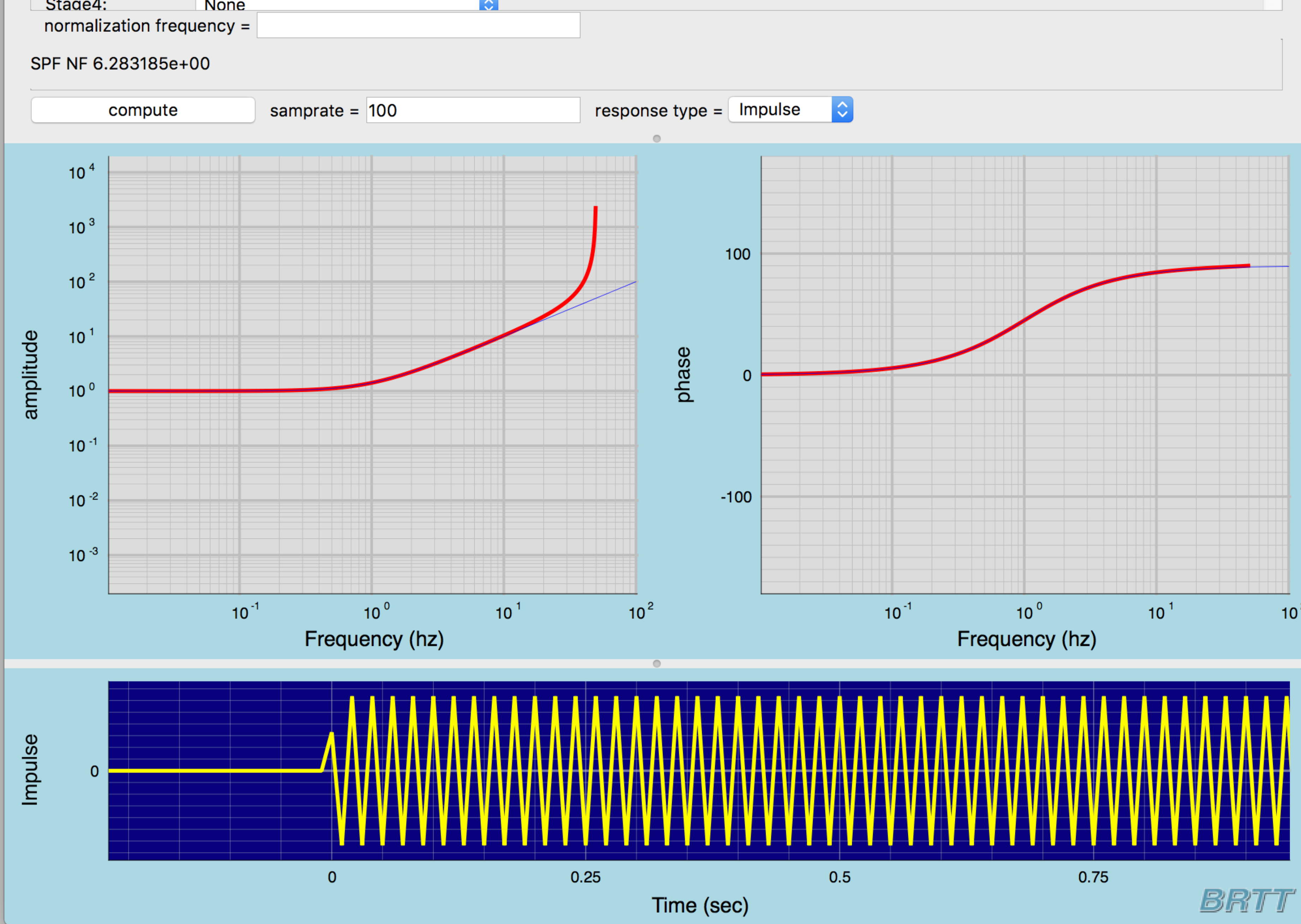
DIF2 2.423538e-02 2.904693e-04 , NF 1.525042e-01 , NF 1.525042e-01 , DS 3.053012e-01 2.348975e-02 , DS 6.398947e+01 3.323318e+03 , NF 7.853982e

compute | samprate = 100 | response type = Step

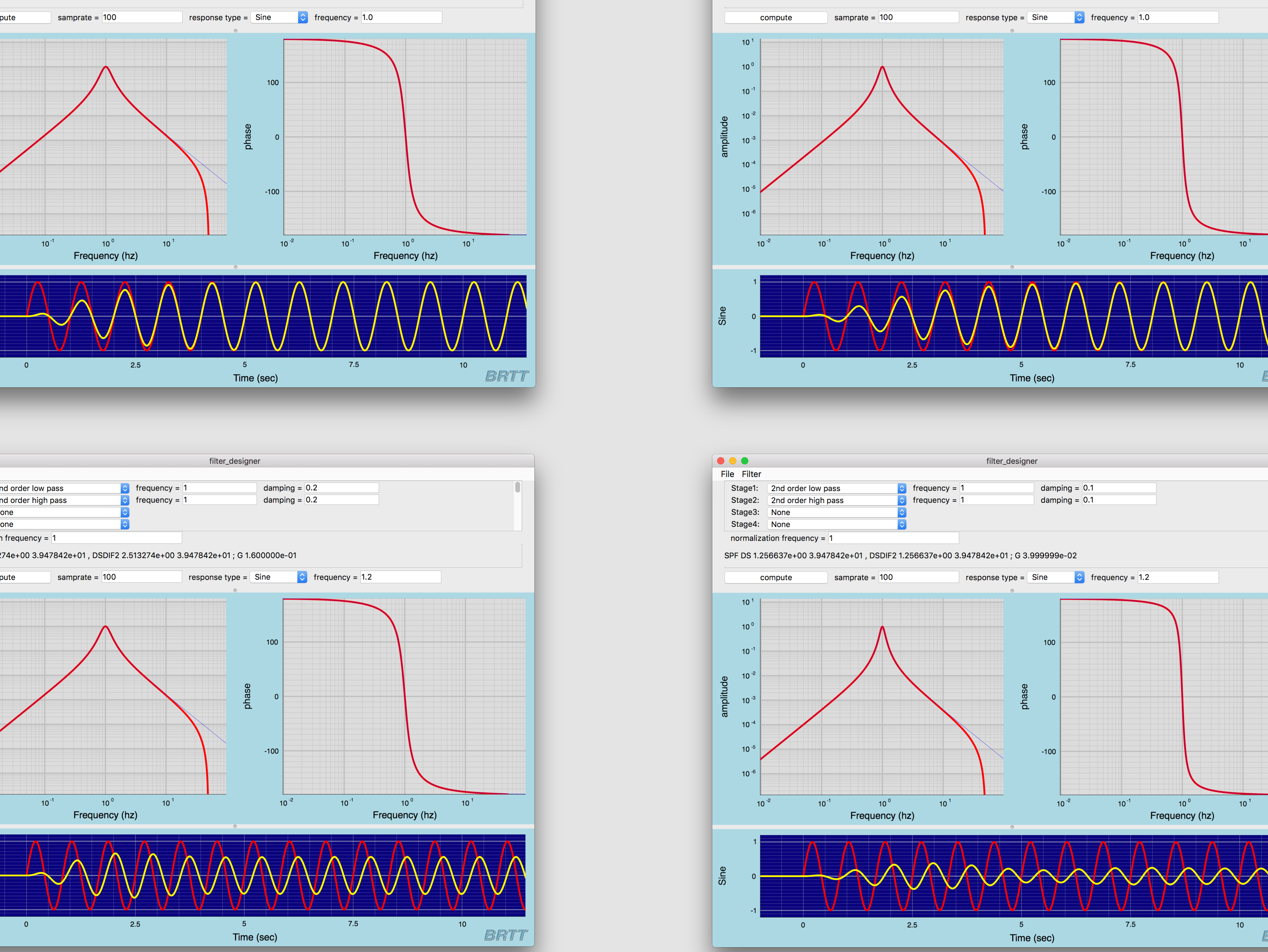








erted filter stages are inherently unstable. They should
be used in combination with non-inverted filter stage



filter_designer
File Filter
Stage1: 2nd order low pass frequency = 1 damping = 0.1
Stage2: 2nd order high pass frequency = 1 damping = 0.1
Stage3: None
Stage4: None
normalization frequency = 1
SPF DS 1.256637e+00 3.947842e+01 , DSDIF2 1.256637e+00 3.947842e+01 ; G 3.999999e-02
274e+00 3.947842e+01 , DSDIF2 2.513274e+00 3.947842e+01 ; G 1.600000e-01
damping = 0.2
compute
samprate = 100
response type = Sine
frequency = 1.0
frequency = 1.2
amplitude
phase
Frequency (hz)
Sine
Time (sec)
BRTT