# Peak Monitoring Tools for Earthquakes

Brian Lantz

T1900149-v1, April 4, 2019

## 1 Summary

We describe a realtime tool to monitor the peak velocity of the ground in the 20 to 100 mHz band driven by surface waves from earthquakes. The goal of this tools is to rapidly and accurately track the peak velocity for the earthquake band. It seems plausible that the peak motion, rather than rms motion, is responsible for lock losses. For peaks, this tool is more accurate and has smaller delays than the BLRMS monitor. We describe the installation and the details of the tool, as well as some off-line tests of the performance.

## 2 Installation

### 2.1 New parts

Update the {userapps}/release/isi/common/ path to get 3 new files (-r 19184 and -r 19185):

```
A    models/Monitor_Library.mdl
A    medm/demo_peak.adl
A    src/PEAK_MON.c
```

Update the SeismicSVN/Common/MatlabTools/EQ_filters to get three more files (-r 9350, -r9355)

```
U    EQ_filters/EQfilts4foton.txt
U    EQ_filters/READ_ME_EQ_filters.txt
A    EQ_filters/make_EQ_filter12.m
```

The new tool is in the Monitor_Library. It's called 'PEAK', and has 1 input (see figure 1).

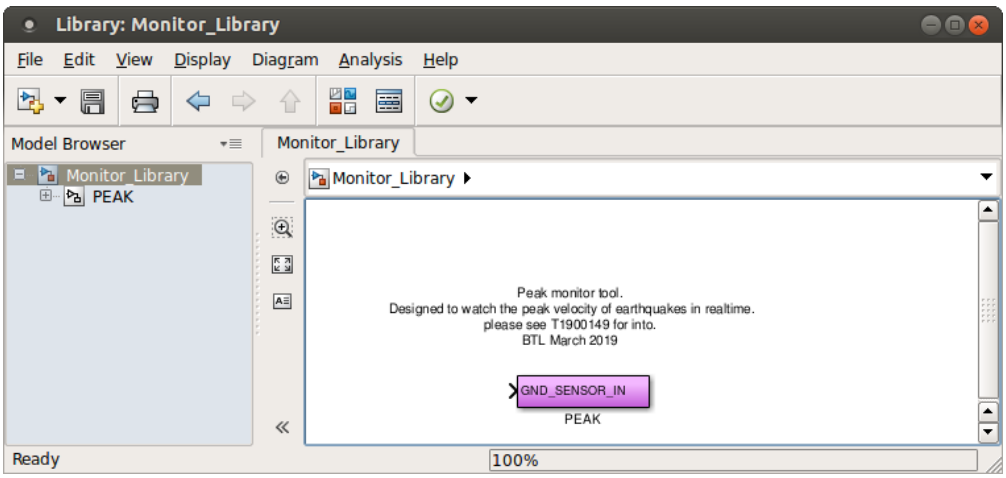


Figure 1: New Peak monitor tool in the Monitor Library.

To install this, I recommend adding this part to the SEIPROC model and attaching it to vertical channel of the ground seismometer in the corner station. Leave the 'PEAK' in the name. For naming consistency, I suggest 'GND_STS_CS_Z_EQ_PEAK'. See figure 2.

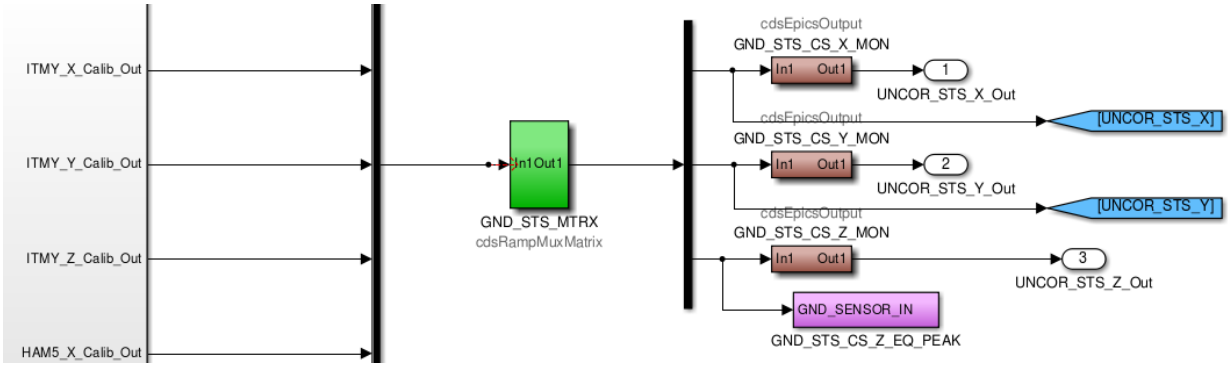


Figure 2: Install the peak monitor tool in SEIPROC to monitor the vertical motion of the corner station STS-2. Give it a good name which includes PEAK. I suggest 'GND_STS_CS_Z_EQ_PEAK'.

## 2.2 MEDM screens

You will need to make a new MEDM screen for the peak monitor. I didn't see an obvious place to put it in the current LHO SEI overview screens. I've made a simple prototype to give a sense of what might work, as shown in figure 3. It is called demo_peak.adl and is saved in the isi/common/medm/ directory. It uses hard-coded names from Stanford.

## 2.3 Setup

Once the part is installed and the MEDM screen is setup, you need to set the time constant. It will start at 1 second, which is the minimum value allowed by the Simulink diagram. (The max allowed value is 500 seconds.) The time constant should be **initially set to 50 seconds**. Feel free to adjust it.

The filter needs to be set up. For LHO, load the filter EQ_Filt12 into the first module and engage it. Leave everything else at the defaults. We'll probably need a different filter for Livingston because the microseism is typically bigger.

**To get EQ_Filt12:**
EQ_Filt12 is saved in a tiny foton file, and you can just copy and paste it. The source file is:
{SeismicSVN}/Common/MatlabTools/EQ_filters/EQfilts4foton.txt.
It is in the filter bank PASSBAND_FILT. You will get a big mess of warnings, but it should be OK.

## 2.4 Key Epics variables

There are 3 important Epics variables:
*blah-blah_PEAK* _OUTMON - current peak value.
*blah-blah_PEAK* _TIME_CONST - Decay time for the peak monitor, in seconds.
*blah-blah_PEAK* _RESET - momentary - the decay history is cleared when this is 1.

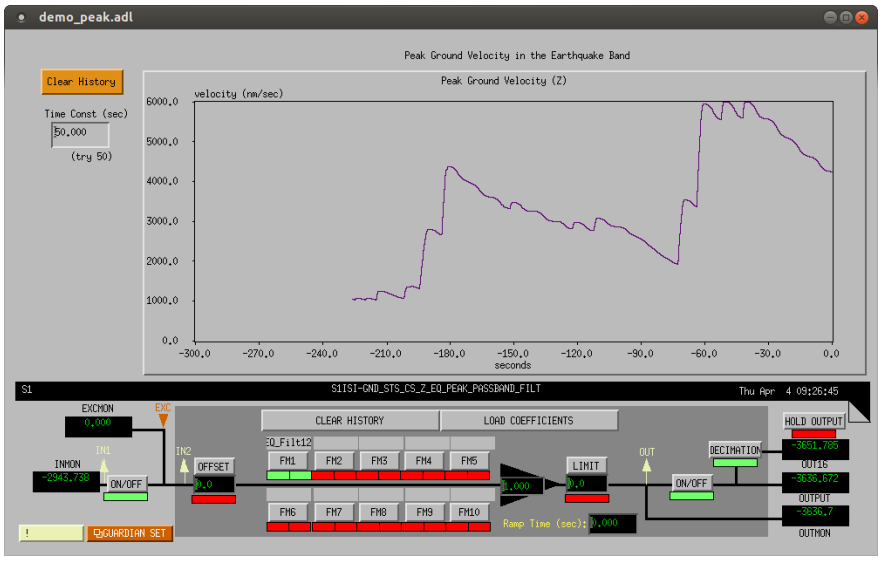


Figure 3: Prototype MEDM screen for the peak monitor. Peak velocities of more than 500 nm/sec are likely to cause trouble, so some marker there might be good. I'm not sure where to set the max value of the history. I set the graph by putting values into the filter bank with the oscillator tool set to 50 mHz w/ variable amplitude. The correct setting for the time constant is 50 seconds, and the correct filter for the first filter module is EQ_Filt12.

There is a realtime testpoint for the peak value as well, which can be used for diagnostics, but is probably not worth writing to the frames. The library part does not write it to the frames.
*blah-blah_PEAK* _OUT - testpoint for current peak value

The CDS filter bank is named
*blah-blah_PEAK* _PASSBAND_FILT - filter bank to select the earthquake band, user defined. Start with EQ_Filt12 for LHO.

## 3 Filter design information

(Adapted from SEI log entry 1438 ). I tried a variety of things to monitor the peak value for the surface waves, based on 3 sets of data I had on hand for LHO. I've used vertical data, since it is less corrupted by tilt. It should also be insensitive to the direction of the quake. This is OK if the Reighley waves are the problem. Not so good if the motion is dominated by Love waves (transverse). However, it seems that the BLRMS monitors for EQs usually show the vertical and horizontal motion moving together, although the ratio is different for LLO and LHO.

The BLRMS filters are terrible for showing the peak velocity. I tried several others, and the trick is to suppress the microseism enough that it doesn't matter, while not distorting the shape of the surface motion. For LHO, I got good results with a modified elliptic stop-band filter from 0.102 to 0.6 Hz, 1 dB ripple, and 20 dB attenuation in the stop band. I also added AC coupling and 1 pole to roll off the high frequency. The gain is set to be 1 at 50 mHz.

I then follow this with a 'jumpy low-pass'. This starts by taking the absolute value of the signal. This is followed by a 1-pole low pass filter which, at each time step, computes the low-pass of the

history, compares that to the current input, and keeps the larger of the two. Thus, it jumps up immediately on big inputs, then falls off with a 50 sec. time constant. One can see in the time series that the motion in the Earthquake band actually varies more rapidly than this, but a shorter time constant makes the peak velocity look really jumpy.

### 3.1 Bandpass Filter Performance

In the following plots I show the performance of the bandpass filter. The data are all taken from the vertical motion seen by the corner station STS-2 at LHO. I show data from 2 earthquakes. The first is a M6.9 on 4/24/2017 near Valparaiso, Chile. The second is from a M5.4 earthquake on Jan 17, 2017 near Guisa, Cuba. See LHO log 33356. The Guisa earthquake is interesting because the motion is smaller and the microseism is at a low freq: 0.115 Hz. This makes it difficult to see what is going on in the raw time series - hence a good test of the filter. This event seriously interfered with locking. These events were used because I've looked at them for other earthquake studies.

In figures 4 and 5 we see the Valpariso quake. EQ_Filt12 does a good job of showing the surface waves. In figure 6 one can compare the performance of this filter with the filter used by the BLRMS tool.

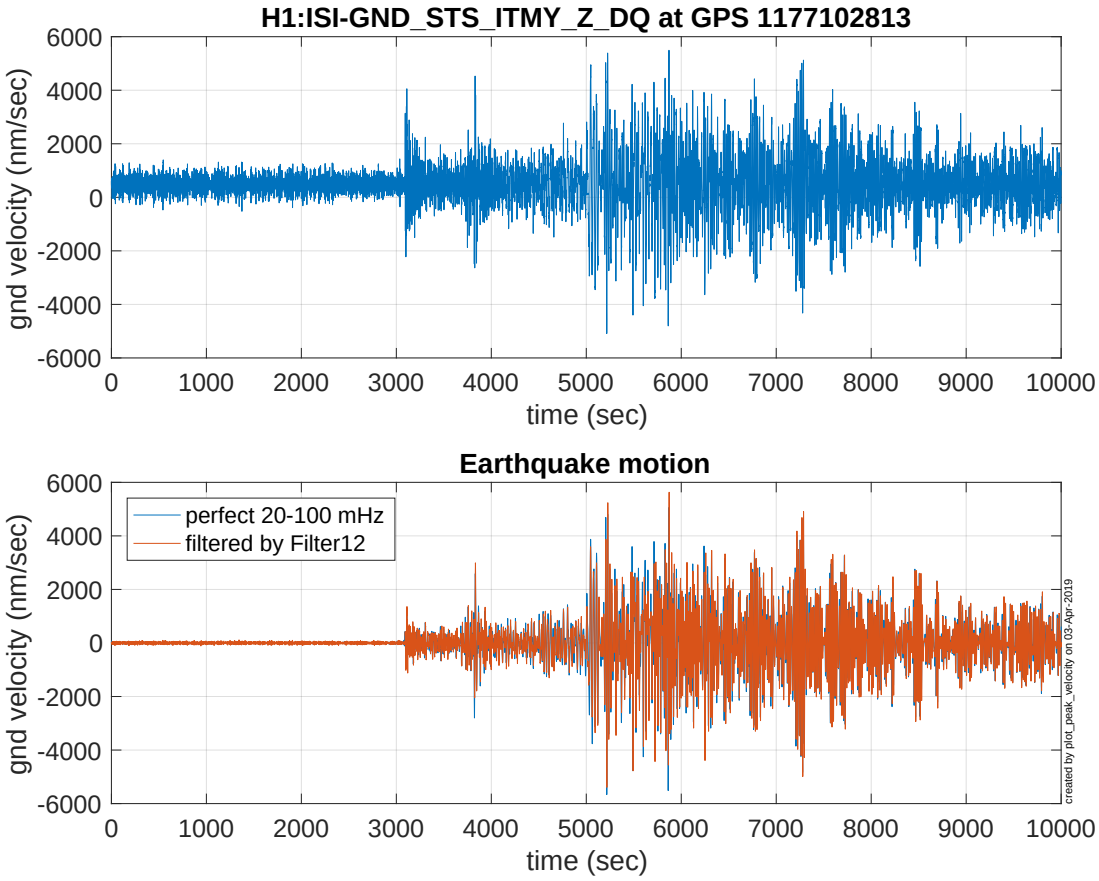


Figure 4: Time series of Valpariso EQ. Top series is the velocity measured by the vertical STS-2 in the corner station. the lower series shows the components of this which are from the earthquake. The blue trace shows the 'best' data for the earthquake data, which comes from a FFT bandpass from 20 - 100 mHz. The red is the data as filtered by Filter 12.

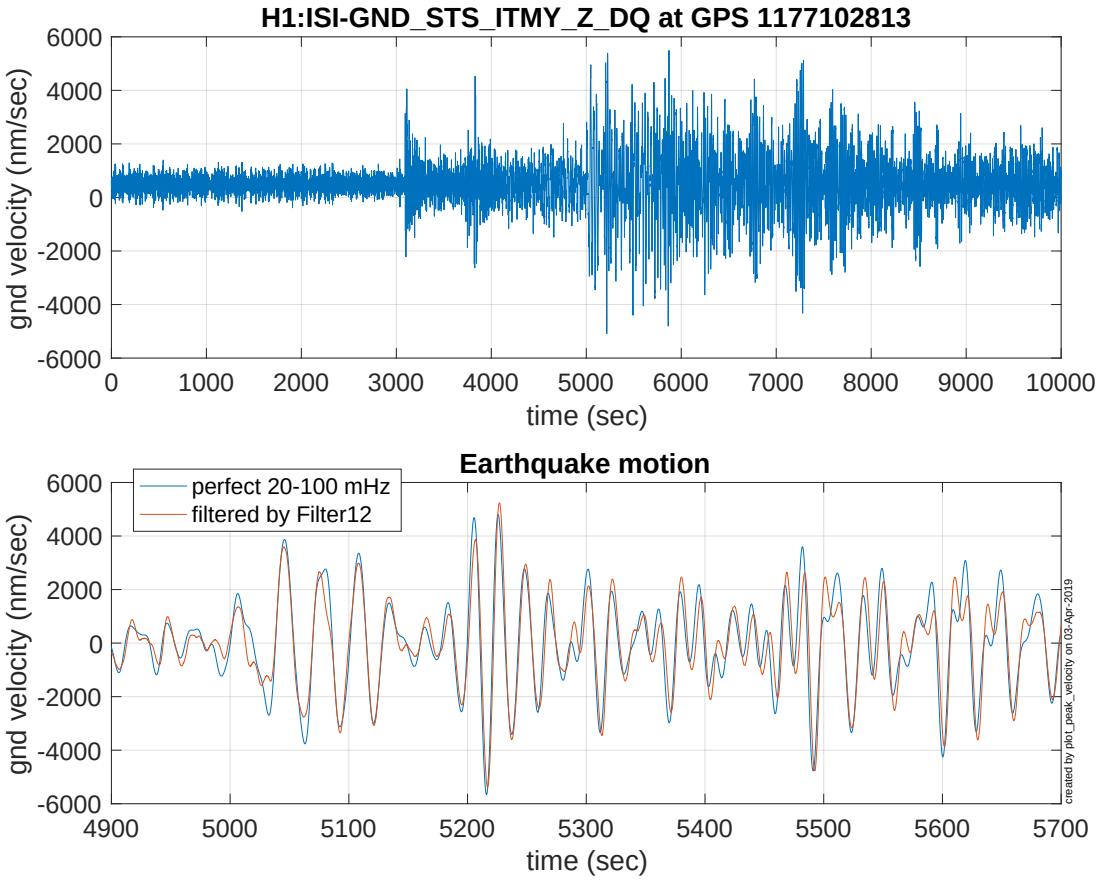


Figure 5: Detail for the time series of the Valparaiso EQ. The difference between the ideal filter and filter 12 is small, but apparent.

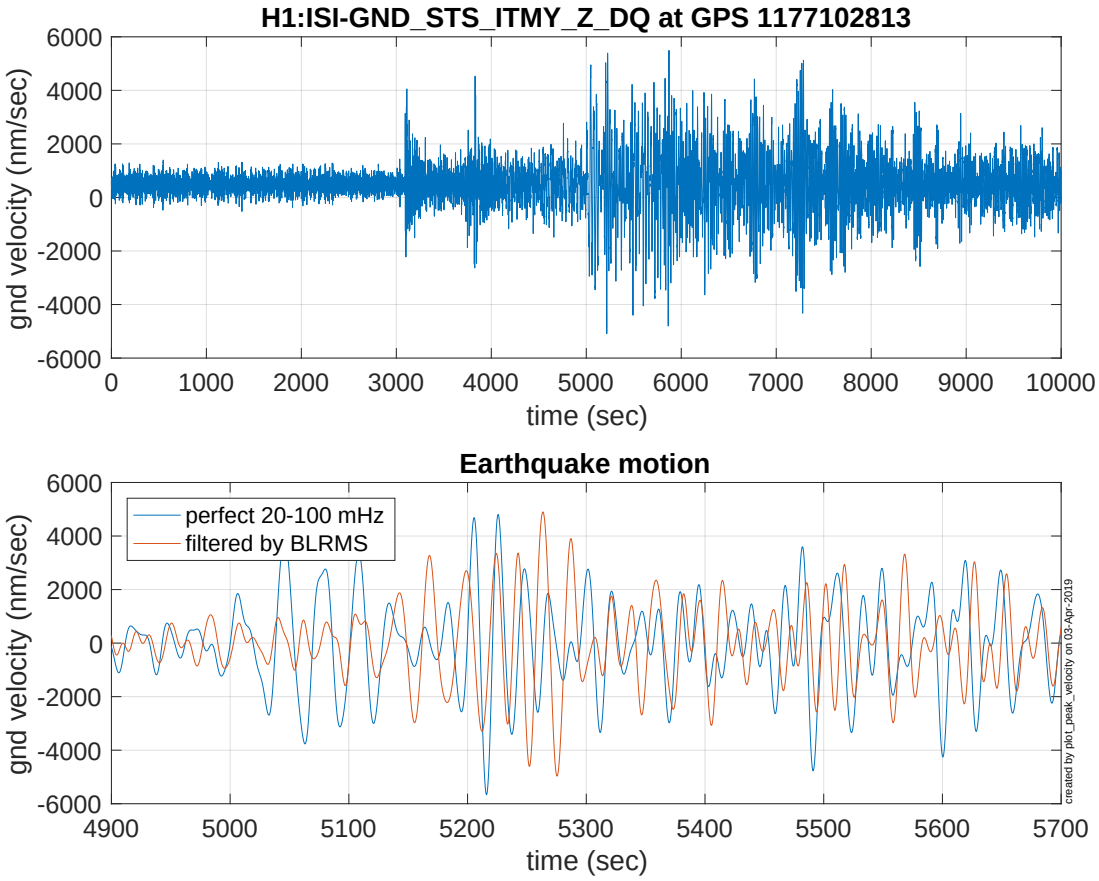


Figure 6: Detail for the time series of the Valparaiso EQ using the BLRMS filter. The RMS is OK, but the amplitude and timing of the peaks is distorted by the phase of the filter.

The in figures 7 and 8, we show the same filter effect for the earthquake near Guisa. Here, we see that the microseism masks the peak velocity of the earthquake. The small relative amplitude and the relatively low frequency of the microseism here set the the limits for Filter 12.

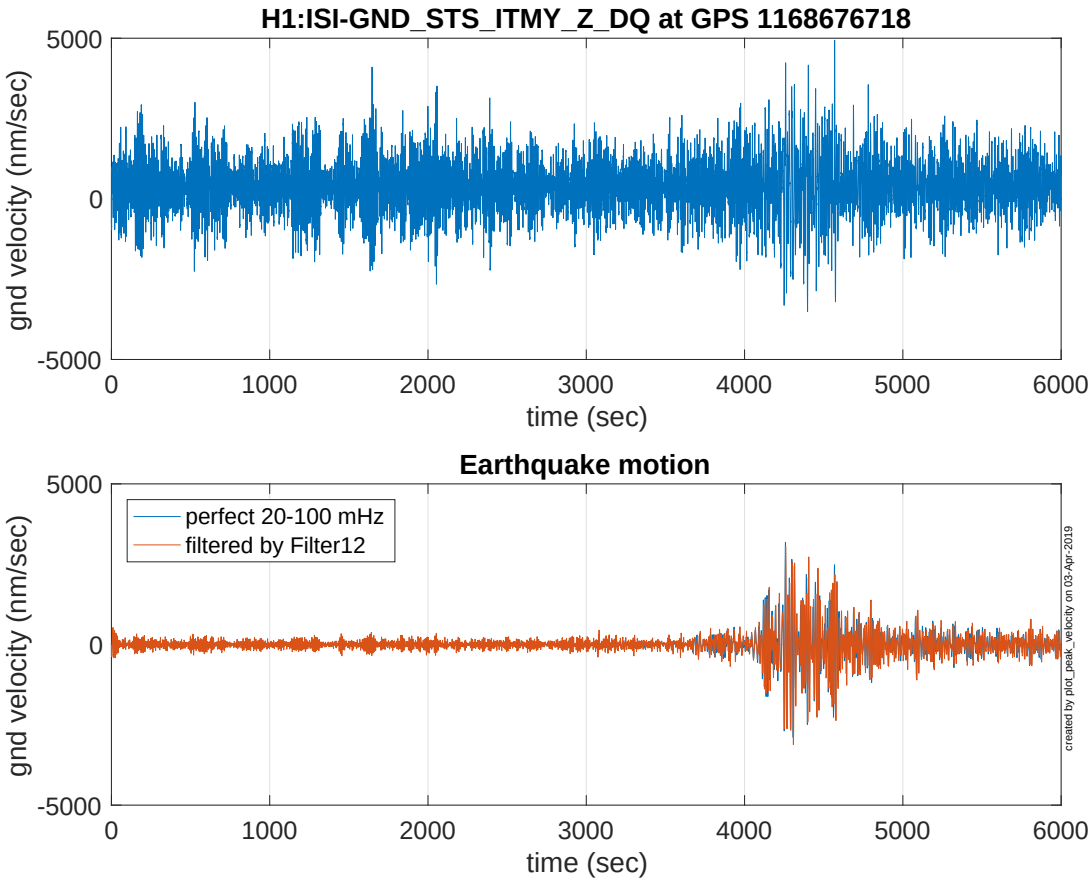


Figure 7: Time series for the Earthquake near Guisa, Cuba on Jan. 27, 2017. The peak velocity of the microseism masks the size of the earthquake, as shown by the difference between the upper and lower panels. The signal in red in the lower panel shows before 3500 sec shows that there is some leakage from the microseism.

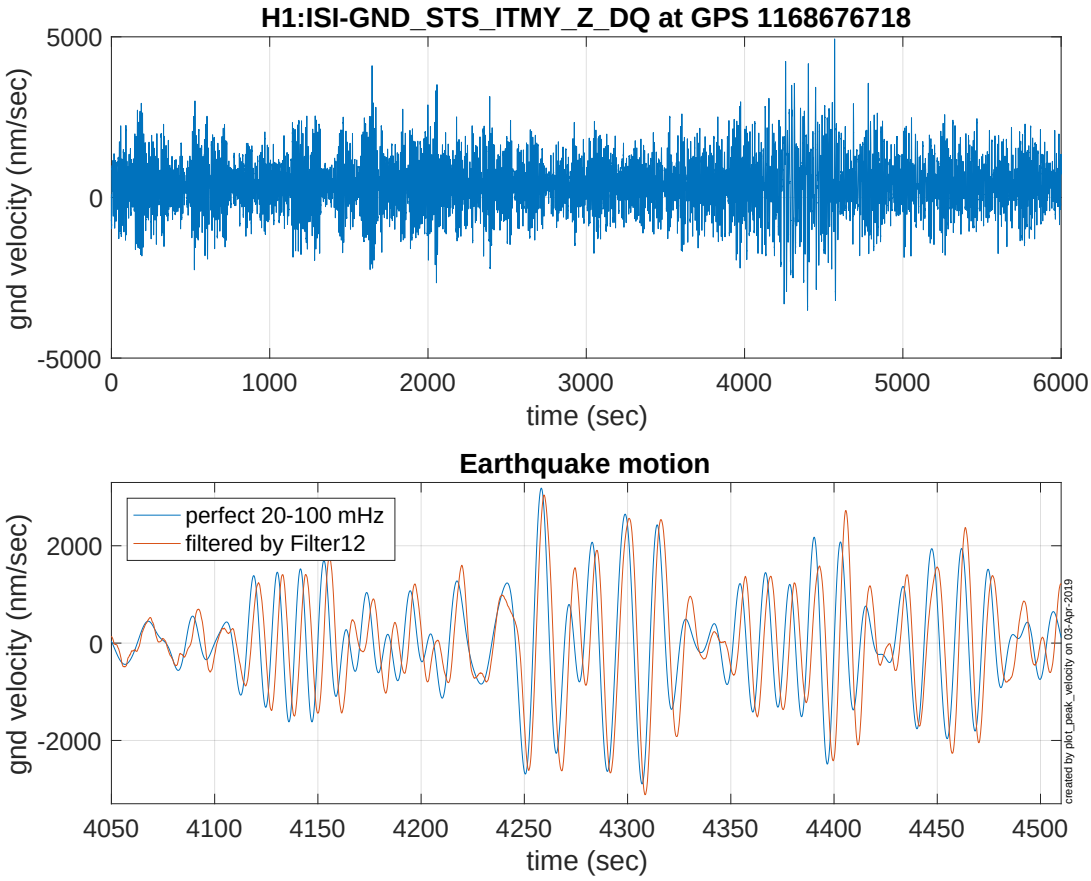


Figure 8: Detail for the Guisa earthquake. Filter 12 is doing a reasonably good job of measuring the earthquake.

### 3.2 Envelope Performance

The second part of the peak monitor is the jumpy low-pass, and is implemented with c-code (included at the end of this document). This starts by taking the absolute value of the signal. This is followed by a 1-pole low pass filter which, at each time step, computes the low-pass of the history, compares that to the current input, and keeps the larger of the two. Thus, it jumps up immediately on big inputs, then falls off with a 50 sec. time constant (which is user set-able).

Figures 9 and 9 show how the code responds to step functions and sine waves.

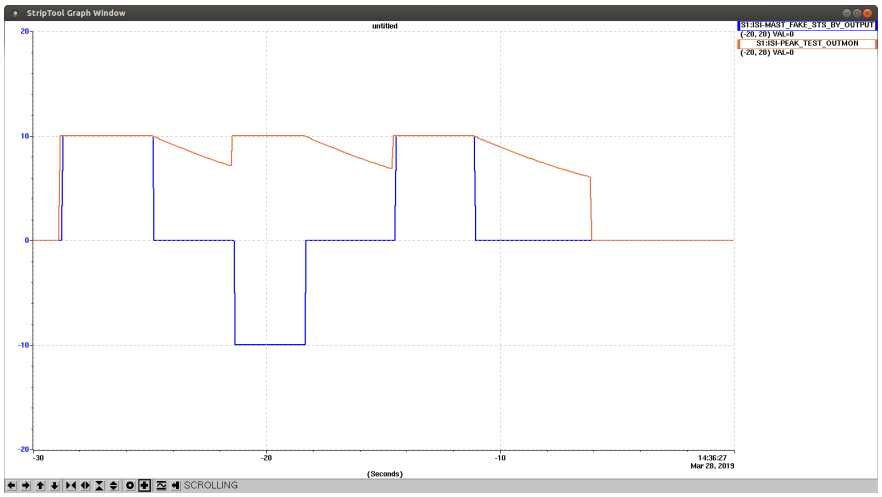


Figure 9: Response of the envelope function to steps. It jumps up immediately, and falls off with a simple time constant.

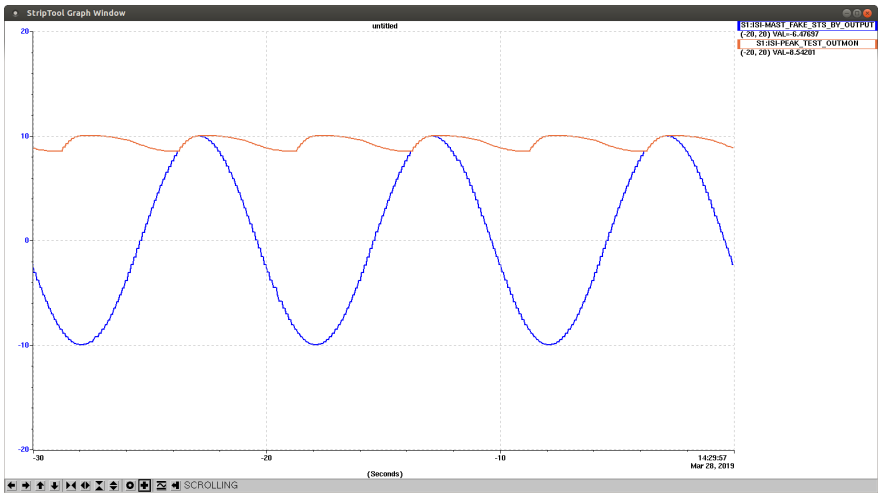


Figure 10: Response of the envelope function to a 0.1 Hz sine wave. It jumps up immediately, and the decay includes averaging from the ongoing signal.

Here we show a few examples of the performance of the bandpass and envelope function combined to make the peak monitor. Figures 11 and 12 show the final output of the peak monitor for the Valparaiso earthquake. Figure 13 compares the peak monitor to the BLRMS.

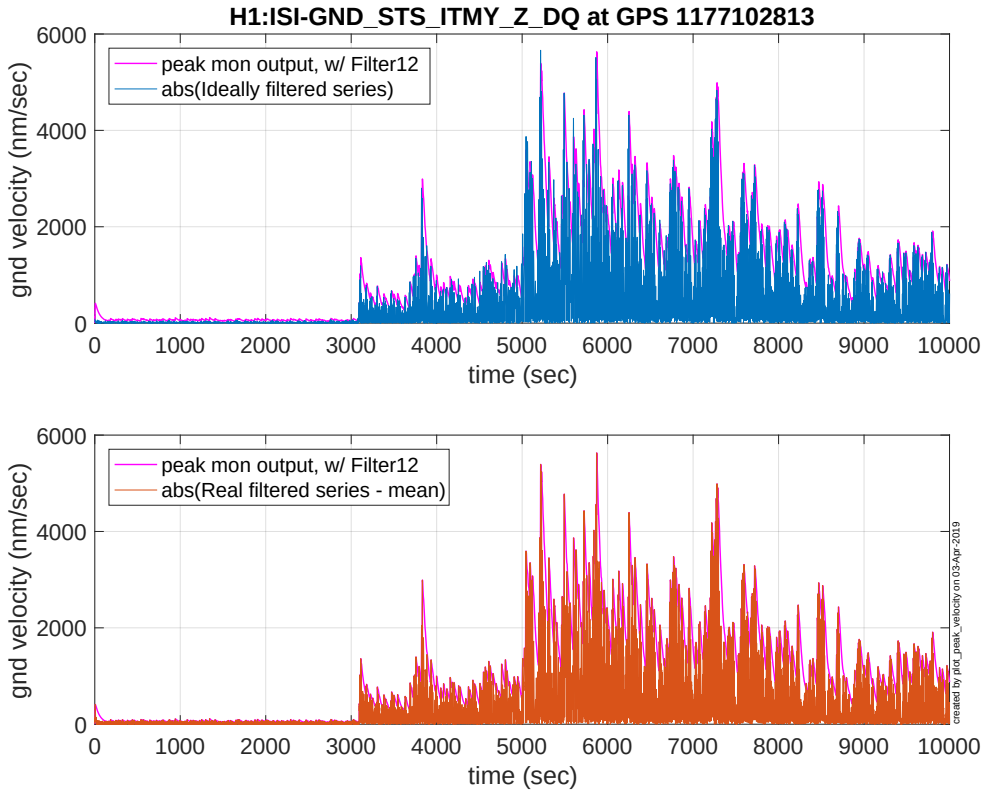


Figure 11: Peak finder output for the Valparaiso earthquake. The upper trace shows the absolute value of the ideally filtered earthquake and the output of the peak finder. The lower trace shows the peak finder and the actually filtered signal. At this time scale, the peak finder looks good.

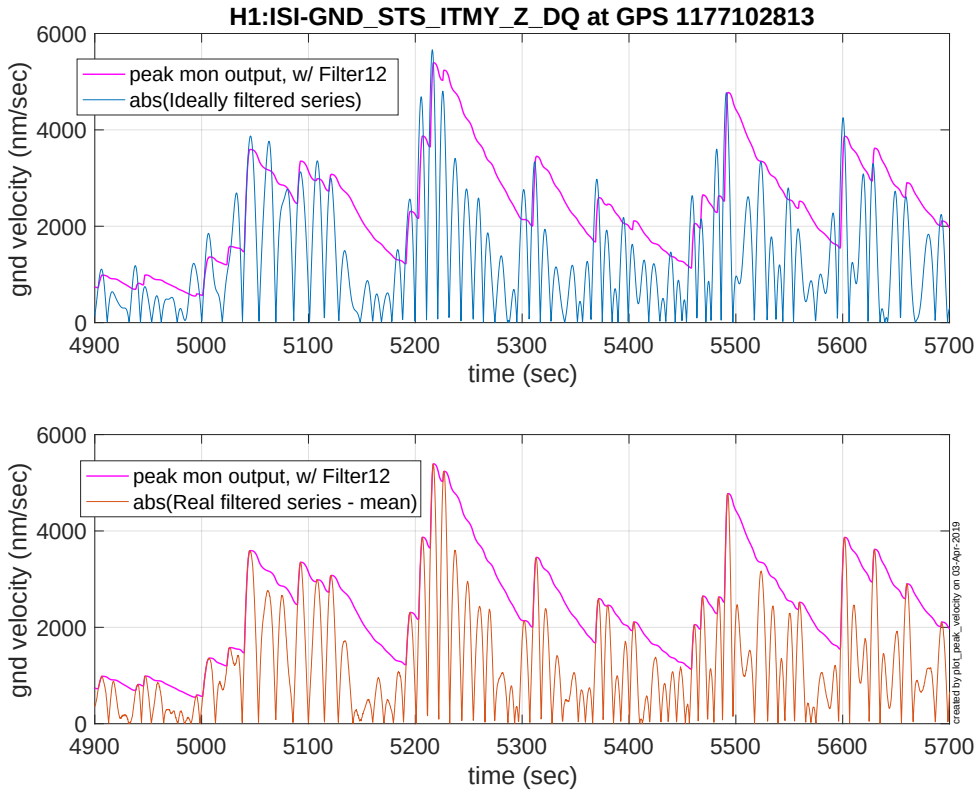


Figure 12: Detail for the peak monitor output for the Valparaiso earthquake. The upper trace shows the absolute value of the ideally filtered earthquake and the output of the peak monitor. By comparing the traces in the upper plot, we get a sense of how well the peak monitor algorithm is doing. The lower plot shows the peak finder and the actually filtered signal. Here we see how well the jumpy-lowpass is performing. The difference between the upper and lower plots comes from the errors introduced by the realtime performance of Filter 12. At this time scale, we can see that this technique is good, but not perfect.

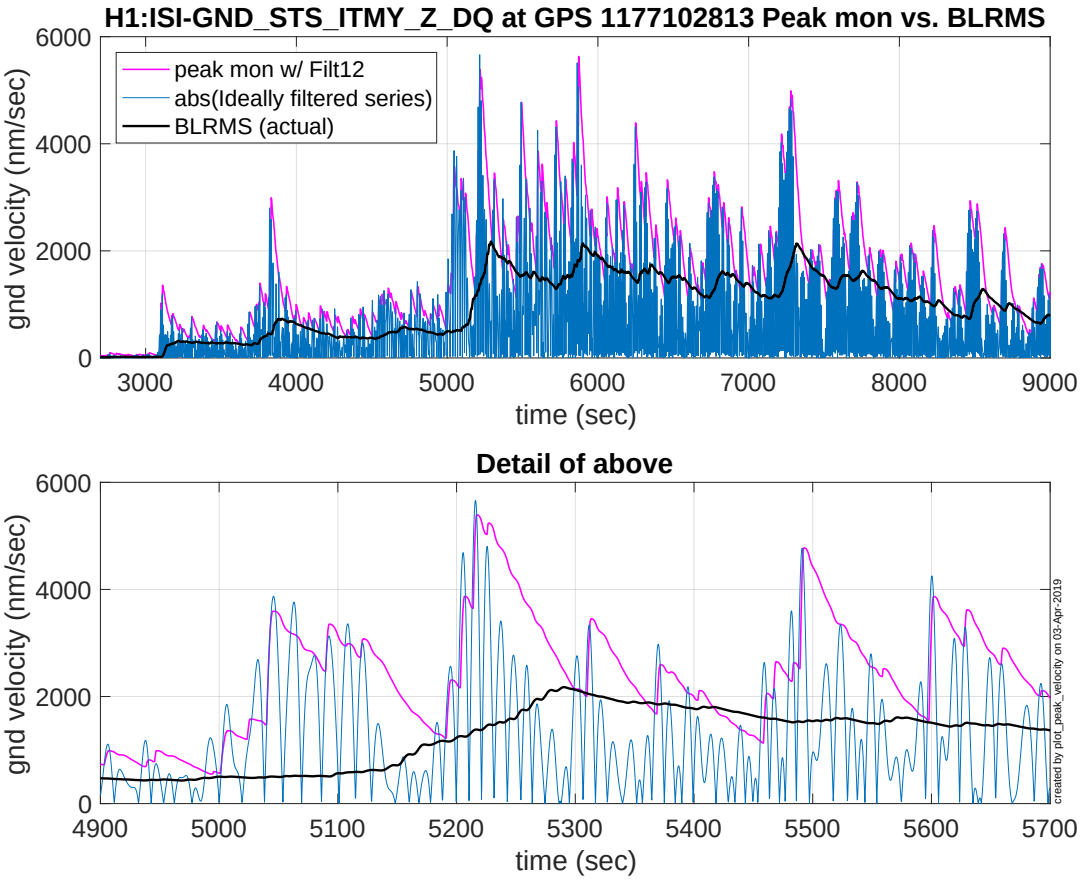


Figure 13: Comparison of the Peak Monitor with BLRMS. It is interesting to compare the peak monitor (magenta) with the BLRMS tool (real data in black). The BLRMS has a single pole with a 147 sec. time constant. BLRMS tracks the in-band RMS, while the peak monitor is much better at showing the peaks.

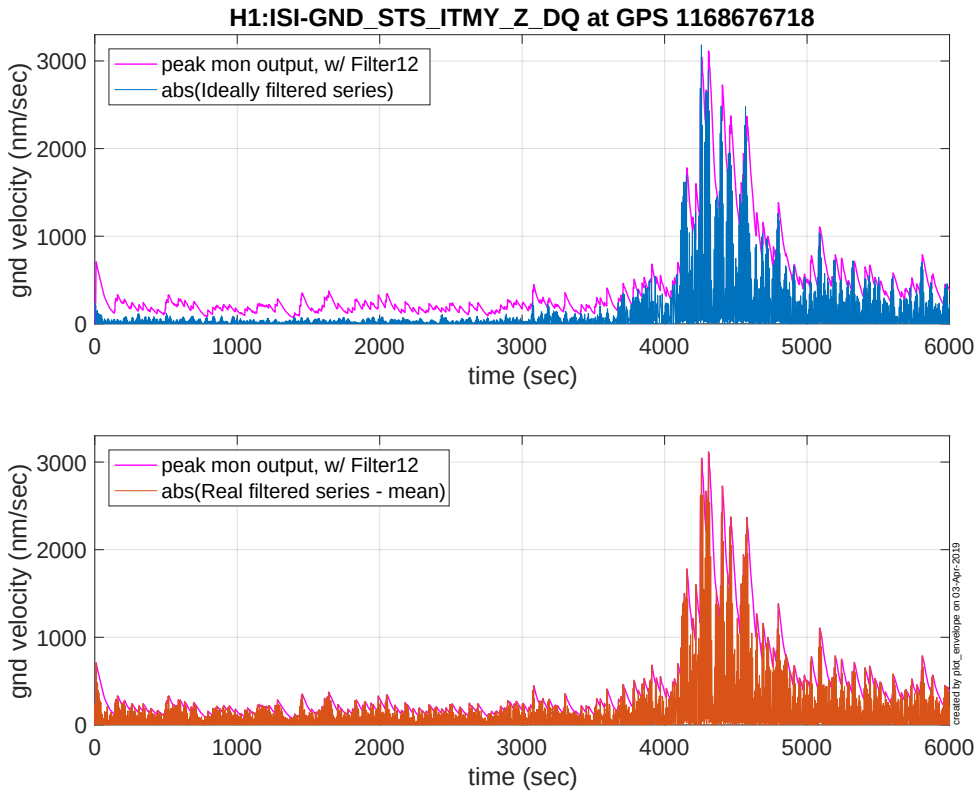


Figure 14: Peak monitor output for the Guisa earthquake. Prior to the earthquake start, one sees that the upper trace has a gap between the peak monitor output and the ideally filtered signal. This comes from leakage of the microseism. This leakage is hopefully acceptable, but may require additional tuning for LHO. It will certainly need to be checked, and probably retuned, for LLO.

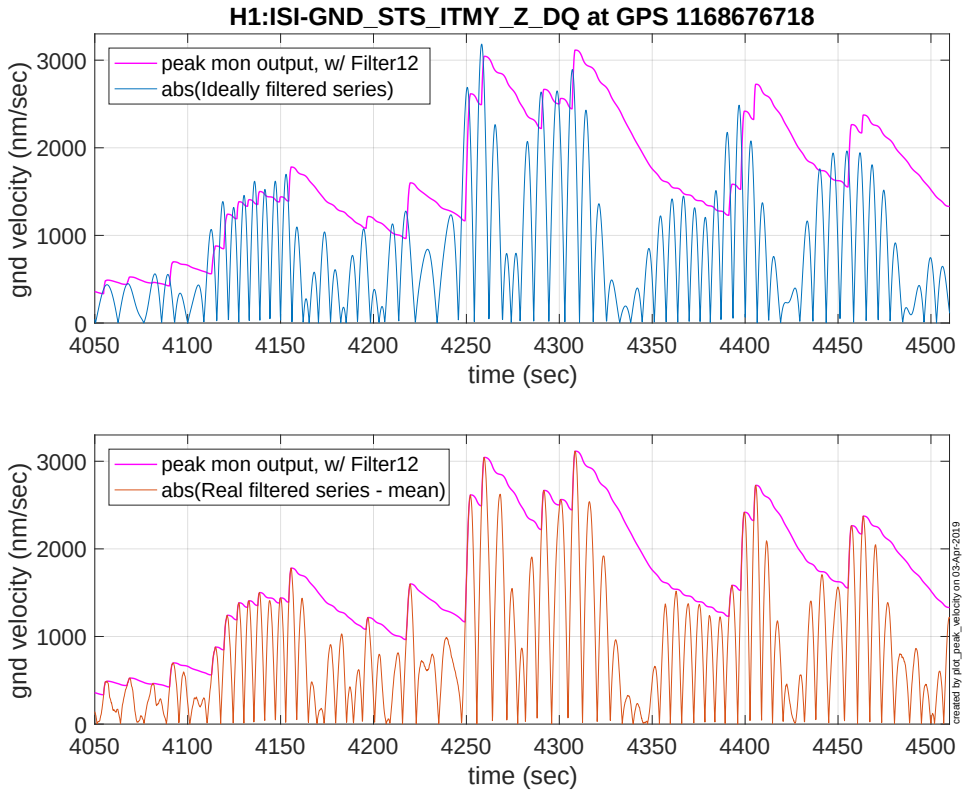


Figure 15: Detail for the peak monitor output for the Guisa earthquake.

### 3.3 The bandpass filter

The bandpass filter is created by the matlab function:
{SeismicSVN}/seismic/Common/MatlabTools/EQ_filters/make_EQ_Filt12

The foton filter is now saved in a small foton file:
{SeismicSVN}/seismicCommon/MatlabTools/EQ_filters/EQfilt4foton.txt.

You can call it yourself with filter_struct = make_EQ_filter12;

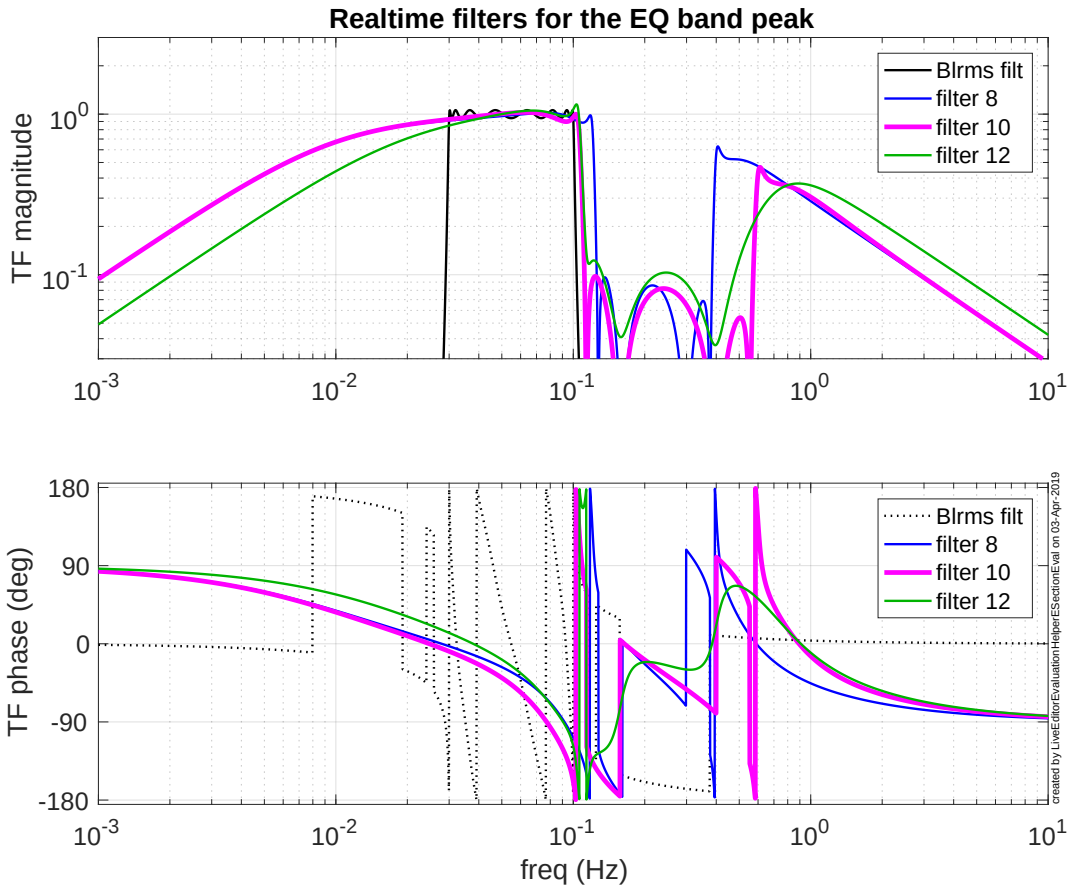


Figure 16: Various filters to transmit the earthquake motion but suppress the microseism. Filter 12 (green) seems like a good choice for LHO. It knocks down the microseism, passes the EQ band, and has the least phase distortion at 50 mHz (-18.3 degrees). The gain is set to 1 at 50 mHz. Note the significant phase distortion of the BLRMS filter. This distortion changes the shape of the time series, distorting the peak values and times.

To make the filter, I started with an elliptic stopband around the microseism, then tweaked it to give a less abrupt high frequency edge, and less phase distortion in the EQ band. The Matlab definition of the filter is:

```
p1 = pair(4.4034, 60);
```

```
v12.pp_c = [ -1*p1,...
    -0.2424 + 0.5102*1i,...
    -0.2424 - 0.5102*1i,...
    -0.0238 + 0.6619*1i,...
    -0.0238 - 0.6619*1i].';

zz1 = pair(0.714526,88);
zz2 = pair(0.98915, 85);
zz3 = pair(2.5114, 85);
v12.zz_c = [-1*zz1, -1*zz2, -1*zz3].';

v12.ellip_v3 = zpk(v12.zz_c, v12.pp_c, 0.76);
v12.thing = zpk(-2*pi*0, -2*pi*[0.020, 0.500], 3.497);  % manually set gain=1 at 50 mHz
filt12 = v12.ellip_v3 * v12.thing;
```

Below, we show the impact of filter 12 on the ASD of the ground motion during the earthquakes. For the big Valparaiso earthquake, the RMS is completely dominated by the earthquake band. For the smaller Guisa quake, there is still some contamination of the integrated RMS by the microseism. This is because the EQ is smaller, and the microseism is at a low frequency. Still, the filter does pretty well.

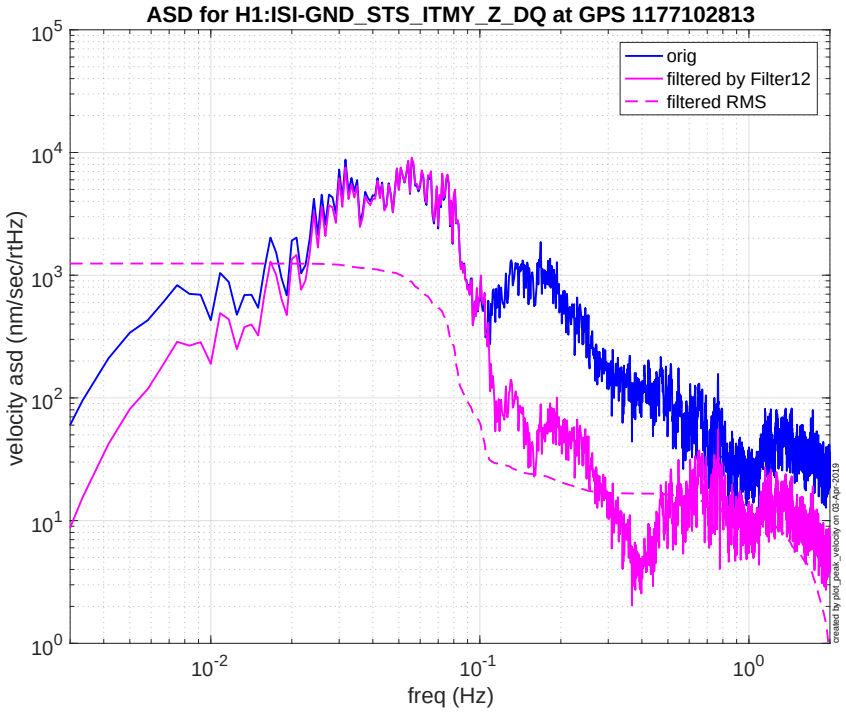


Figure 17: ASD of the Valparaiso quake before (blue) and after the filter (magenta). The filtered signal is dominated by the surface wave motion.

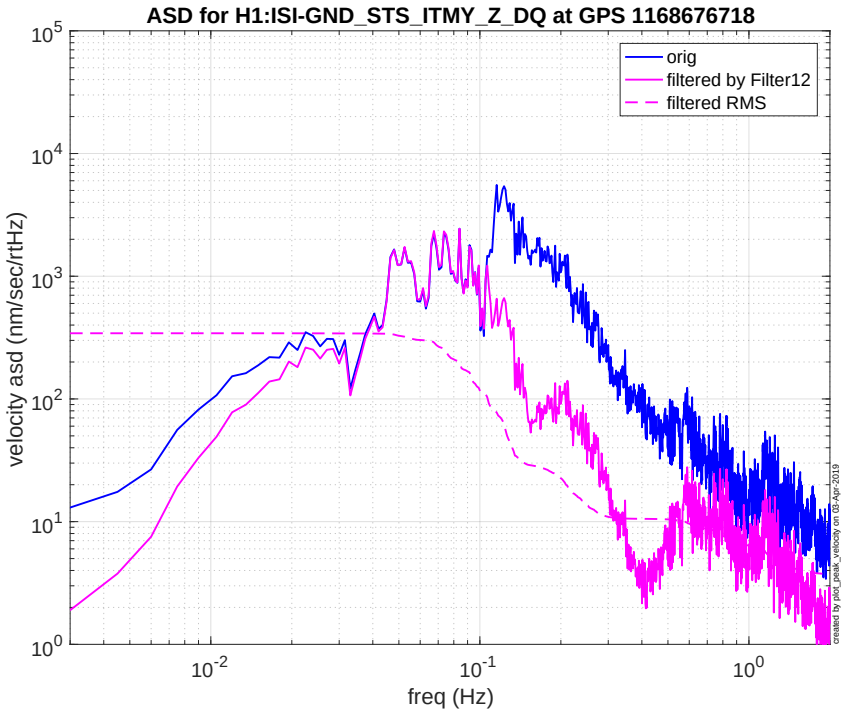


Figure 18: ASD of the Guisa quake before (blue) and after the filter (magenta). The microseismic motion contributes to the RMS, but the earthquake motion still dominates.

## 3.4 PEAK Tool in Simulink

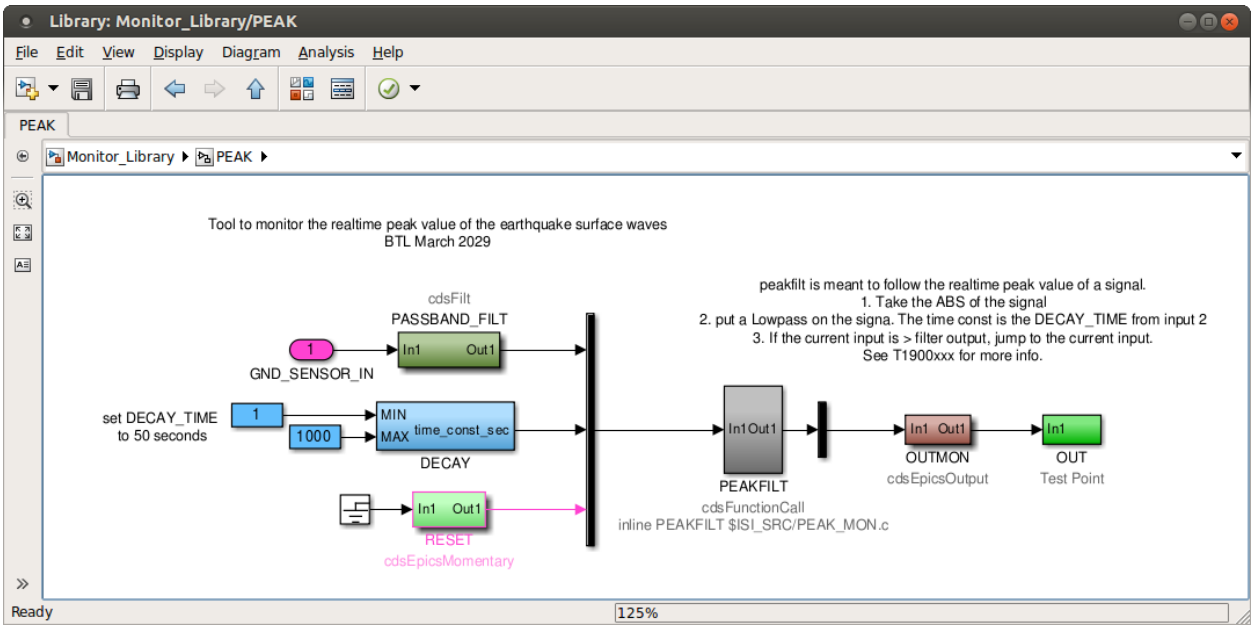


Figure 19: This is the Peak monitor tool in simulink. The Blue DECAY block is one of those new epics input w/ limits. The c-code in the PEAKFILT block is included below. The tool is pretty simple.

## 3.5 C-code

The c-code is pretty simple. It has 3 inputs and 1 output. The code is in the file
/opt/rtcds/userapps/trunk/isi/common/src/PEAK_MON.c
The called function is PEAKFILT.
One thing to note is that the limits of 0 seconds and 1000 seconds are hard coded in to help prevent bad stuff from happening. There are ALSO limits at the simulink level, so please be careful. Limits are hard coded because a time-constant of less than -dT will cause the output to diverge. Yikes! I suppose it could be stretched to more than 1000 seconds. That would not break anything.

/opt/rtcds/userapps/trunk/isi/common/src/PEAK_MON.c

```
/* PEAK_MON.c Function: PEAKFILT
 *
 * filter to show the max level of a signal. Jumps up to the max input, and decays
    down.
 *
 * Inputs:
 *
 * (0) double input_val: the input data stream to monitor
 * (1) double Time_const: decay time in seconds for the lowpass
 * (2) int: Reset - when =1, reset the history to 0
  *
 * Outputs:
 *
 * (0) double output_val: the output
```

```
 *
 *
 * Authors: BTL
 * March 27, 2019
 *
 * pseudo_code
 * look for the "current" peak value of a signal:cd
 * apply ABS() to the input.
 * Apply a 1-pole low-pass filter to the abs-input.
 * If the current abs-input is bigger than the filtered input, use the current abs-
    input as the output,
 * otherwise use the filtered input as the output.
 * This will jump up to the current max right away, then decay down.
 *
 */


#define MODEL_RATE FE_RATE
#define IN_VALUE        0  // these are the control input channel numbers (for
    convenience)
#define IN_TIME_CONST   1
#define IN_RESET        2

#define OUT_SIGNAL      0  // these are the output channel numbers (for convenience)

#define MIN_TIME_CONST 0.0         // limit the time const between these values
#define MAX_TIME_CONST 1000.0      // time in sec.


void PEAKFILT(double *argin, int nargin, double *argout, int nargout){
        static double previous_output = 0.0;  // initial history to be zero.
        static double dT = 1.0 / MODEL_RATE;
        float this_input;
    float abs_input;  // abs of the input.
    double this_output;
    double time_const;   // decay time in seconds
    double alpha;        // for the LP calculation
    int reset_request;
    double filter_val;   // value of the LP filter calc.

    // part 1: read/ evaluate the inputs

    this_input = argin[IN_VALUE];
    abs_input = (this_input >= 0) ? this_input : -this_input;  // take the abs of
        the input

    time_const = argin[IN_TIME_CONST];
    if (time_const < MIN_TIME_CONST) {
        time_const = MIN_TIME_CONST;
    } else if (time_const > MAX_TIME_CONST) {
        time_const = MAX_TIME_CONST;
    }

    reset_request = argin[IN_RESET];
    if (reset_request == 1) {
        previous_output = 0.0;
    }

    // part 2: do the calc.
```

```
    alpha = dT / ( dT + time_const);
    filter_val = (alpha * abs_input) + ((1.0 - alpha) * previous_output);
    if (abs_input > filter_val) {
        this_output = abs_input;
    } else {
        this_output = filter_val;
    }
    previous_output = this_output;  // for the next cycle.


    // part 3: set the outputs
    argout[OUT_SIGNAL]  = this_output;
}
```