Seismic Event MatLAB Suite (SEMS) Cookbook

Dane Ketner
Alaska Volcano Observatory
dketner@usgs.gov

February 22, 2011

Contents

1 Introduction to SEMS **3**

2 Helicorder Class **4**

2.1 Helicorder Multi-waveform display . . . 5

2.2 Helicorder event overlay . . . 8

2.3 Helicorder VLP overlay example . . . 9

2.4 Valid helicorder property name/value combinations . . . 11

2.5 Interacting with a helicorder figure . . . 12

3 Event Detection Using STA/LTA **13**

4 Event Pick **15**

5 Event Metrics **16**

1 Introduction to SEMS

The Seismic Event MatLAB Suite is a collection of MatLAB classes, functions, and GUIs used to capture, manipulate, and display large numbers of discrete seismic events. SEMS is build upon the Waveform Suite and other classes found in the GISMO Suite. SEMS commonly references event times in following formats depending on the application:

1 SST - Event Start/Stop Times, event times are listed as a Nx2 array of MatLAB datenum values. This is the bare bones format for tracking periods of interest from a given waveform. The SCNL object is the other piece needed to recover waveform data from SST format.

2 WFA - Event Waveform Array, events are contained in a 1xN array of waveform objects. This is the neccessary format for most operations.

3 NAN - Event NaN Form: Events are contained in a longer waveform object with non-event times filled with NaN values. This is a useful format primarily for plotting purposes.

There are a handful of functions that quickly fascilitate conversion between these formats:

- `sst2wfa`
- `sst2nan`
- `wfa2sst`
- `wfa2nan`
- `nan2sst`
- `nan2wfa`

SEMS includes a few functions additionally that operate only on SST arrays including:

- `add_sst`
- `delete_sst`
- `extract_sst`
- `merge_sst`
- `search_sst`
- `sort_sst`
- `compare_sst`

Typing `help function_name` for any of the functions discussed in this manual should reveal more specific information about its functionality. The functions in SEMS are designed to be both modular and customizeable.

2 Helicorder Class

A helicorder (or seismogram) is a multi-line display of ground motion traditionally recorded onto paper around a rotating drum. The MatLAB helicorder class is designed to generate such a multi-line display and includes a number of customizeable properties that can be defined by the user, or ignored. One can think of a helicorder object `h` as a blueprint for the construction of a helicorder figure. The `build` function is then used to create the helicorder figure based on the properties in `h`. User-defined properties can override default properties if included in a call to the heliorder constructor, though all that is needed for a basic helicorder is a single waveform object input argument. In the following example, `w` is a waveform object containing three hours of data from REF:EHZ starting from March 29, 2009 starting at 07:00 UTC. Figure 1 displays the output of the `build` function, with `fh` being the handle of the figure generated.

```
h = helicorder(w);
fh = build(h);
```

We see that the resulting helicorder figure contains 10 minutes of data per line, and that waveform data is printed in black. Also notice the amplitude scaling that occurs in the plot. This scaling sets the max datapoint value at the center of the trace two traces above (or below) the current trace. The max value in this plot appears to be the first earthquake on line 6 (Line 07:50:00) which reaches up to the center of line 4 (Line 07:30:00). Minutes per line, trace color, and scale are all properties that can be manually set. A user-defined property can be set upon initial creation of `h` in the following way: `h = helicorder(w, 'mpl', 15);`. This call to the constructor would override the default value of 10 with 15 minutes per line. Property values can also be set after `h` is constructed using the `set` function as follows: `h = set(h, 'mpl', 15);`.

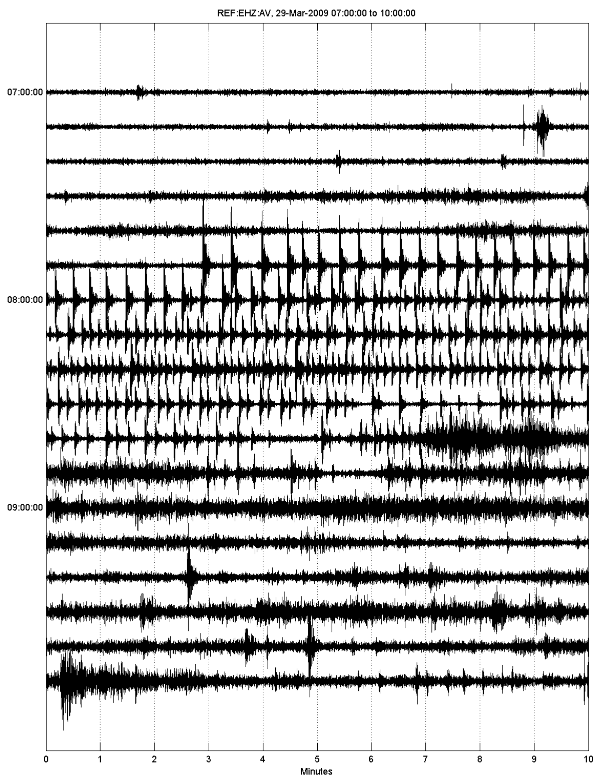

Figure 1: Default helicorder display for a single waveform

2.1 Helicorder Multi-waveform display

Next, let's test the default display created from an input `w` which contains multiple waveforms. In this case `w` is a 1x3 waveform object containing all three components from station REF for the same time span as the previous case. We know there will be more data to display than the previous example, so we set 20 minutes per line: `build(helicorder(w,'mpl',20))`. EHZ data is displayed in blue, EHE in black, and EHN in green in Figure 2. This multi-waveform display type is called 'alternate' which can be changed by the user in addition to the color scheme.

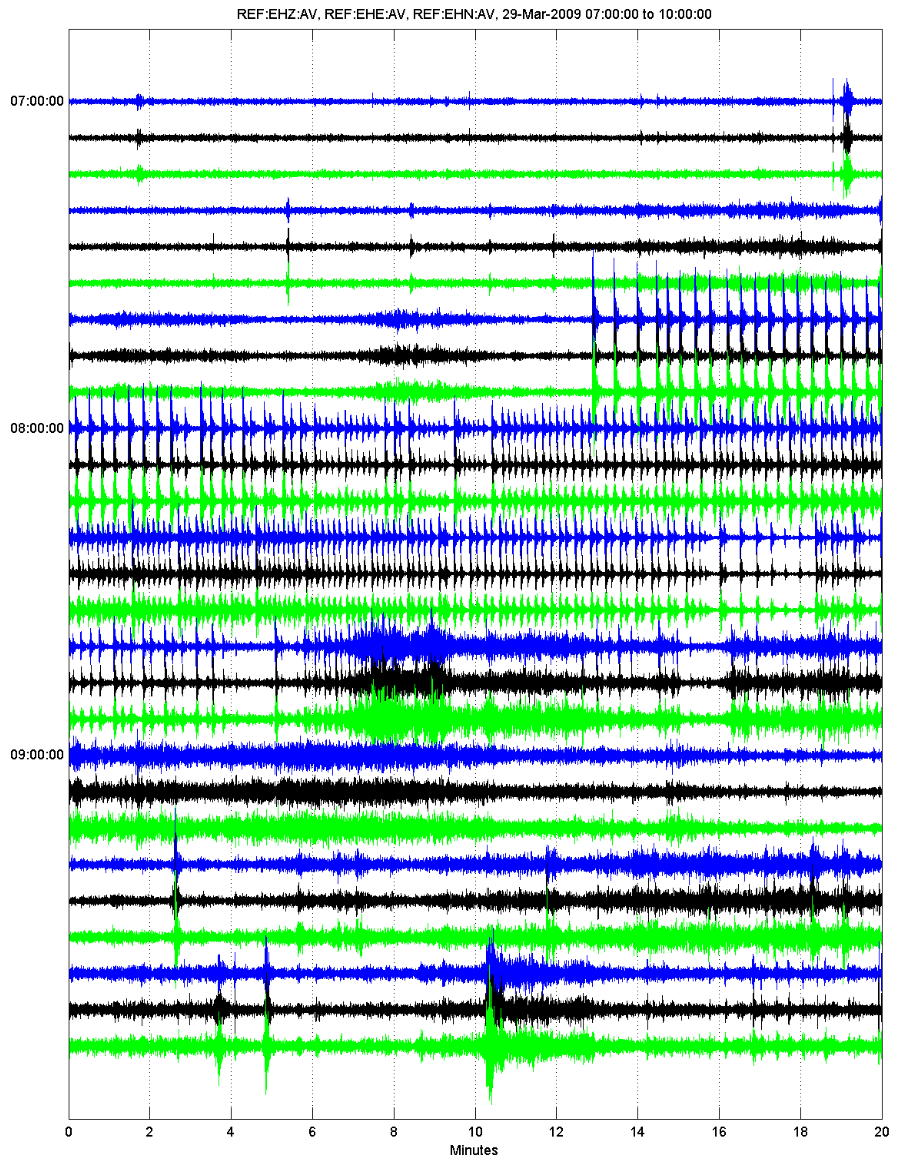

Figure 2: Default helicorder display for a 1x3 waveform

The multi-waveform display type can be changed from 'alternate' to 'group' or 'stack'. Setting display to 'group' will plot all trace data from the first element in `w` before starting the next. Figure 3 displays the output from `build(helicorder(w,'mpl',20,'display','group'))`.

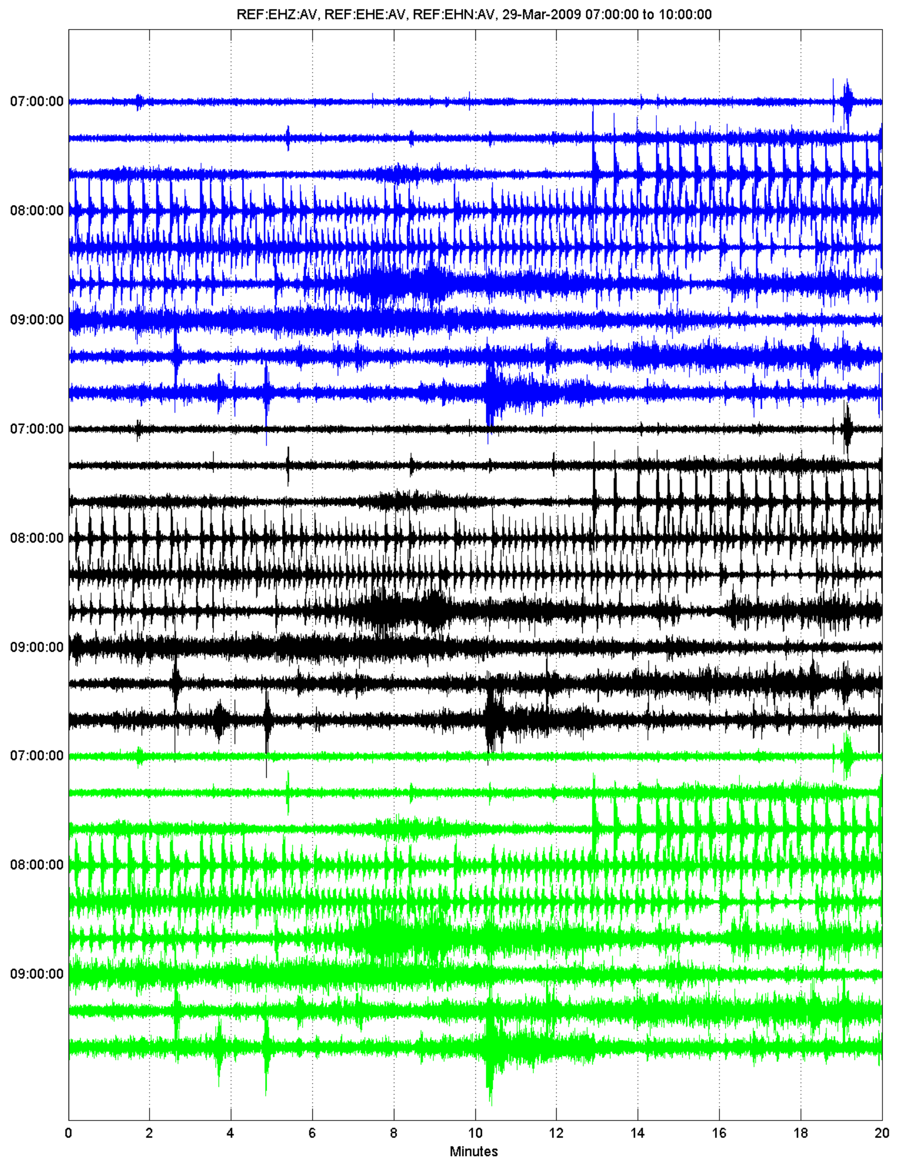

Figure 3: Multi-waveform 'group' display

The final multi-waveform display type is 'stack' which will plot waveforms one on top of the other. It should be noted that the default amplitude scaling of all waveforms is determined by the first waveform plotted in 'stack' mode, in this case `w(1)` is REF:EHZ. If separate amplitude scaling is desired for waveforms in w, it must be defined by the user. Figure 4 displays the output from
`build(helicorder(w,'display','stack'))`.

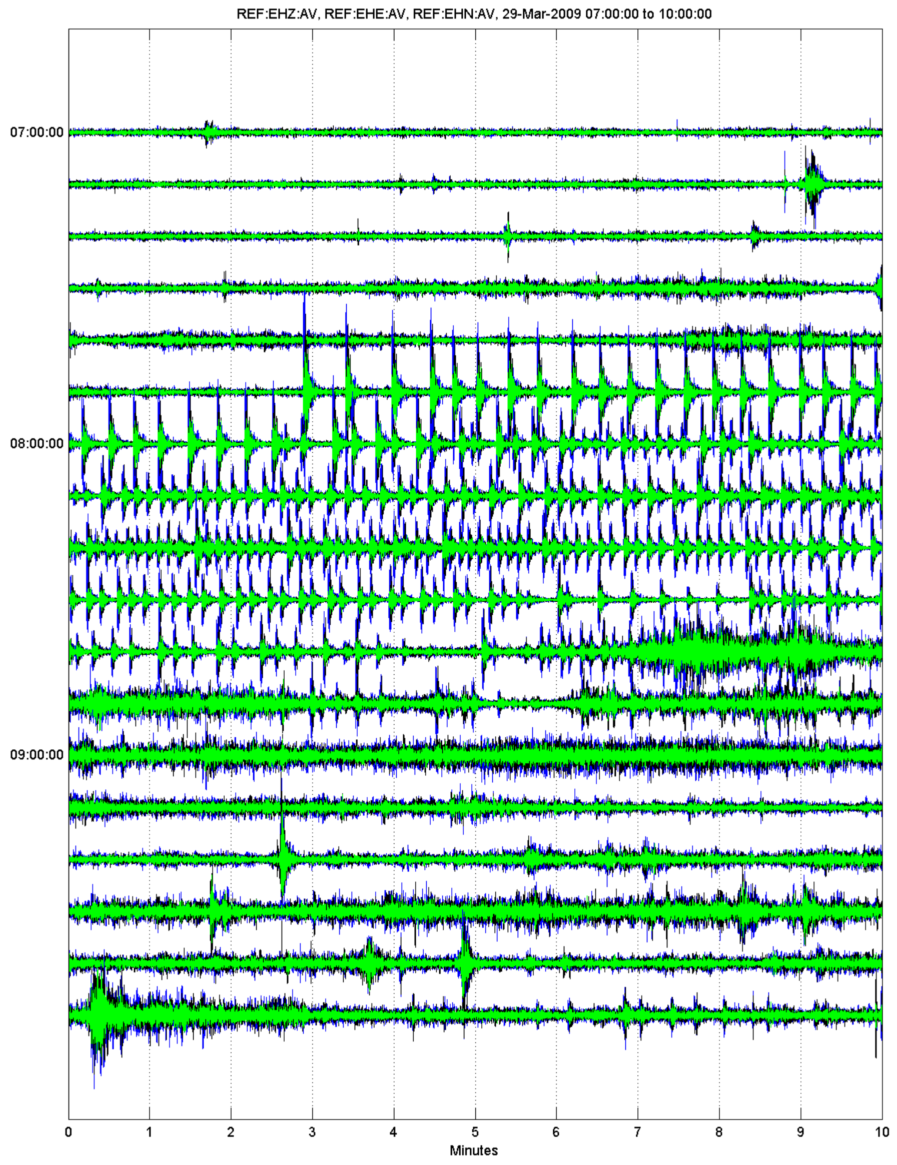

Figure 4: Multi-waveform 'stack' display

2.2 Helicorder event overlay

Helicorder can also be used to display discrete events from a waveform object. To do this, a set of event start/stop times (`sst1`) is required. Start/stop times can be manually picked, or can be the result of an auto-detection algorithm such as STA/LTA. 212 events are detected from the waveform data in Figure 5. In our case, `sst1` is a 212x2 array of MatLAB times defining the begining and end of each event. Figure 5 displays the output from `build(helicorder(w,'mpl',15,'e_sst',sst1))` where `w` is broadband data from RDWB:BHZ from March 27, 2009 that has been band-pass filtered between 1 Hz and 10 Hz.

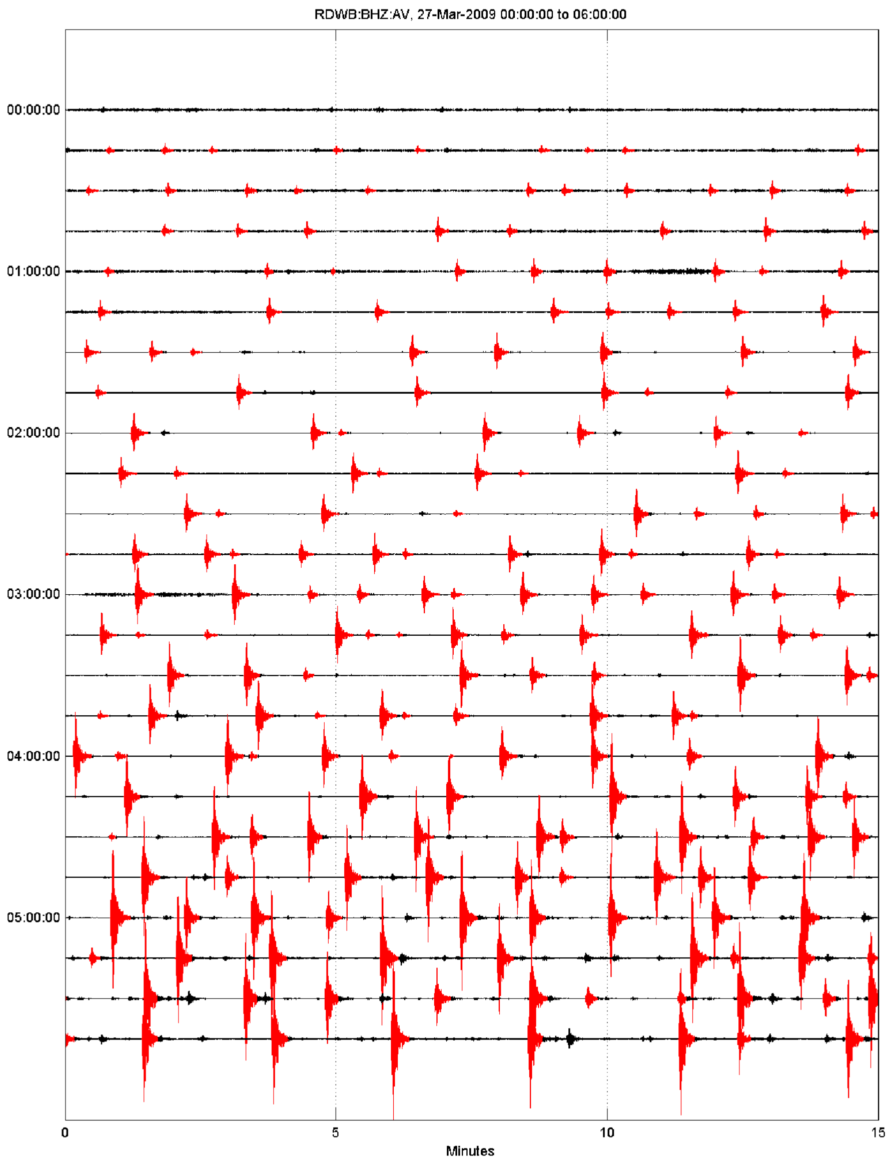

Figure 5: Helicorder with event overlay

2.3 Helicorder VLP overlay example

In the next example, `w` contains the unfiltered data from the previous event overlay example. Calling `build(helicorder(w,'mpl',15))` returns the following display:

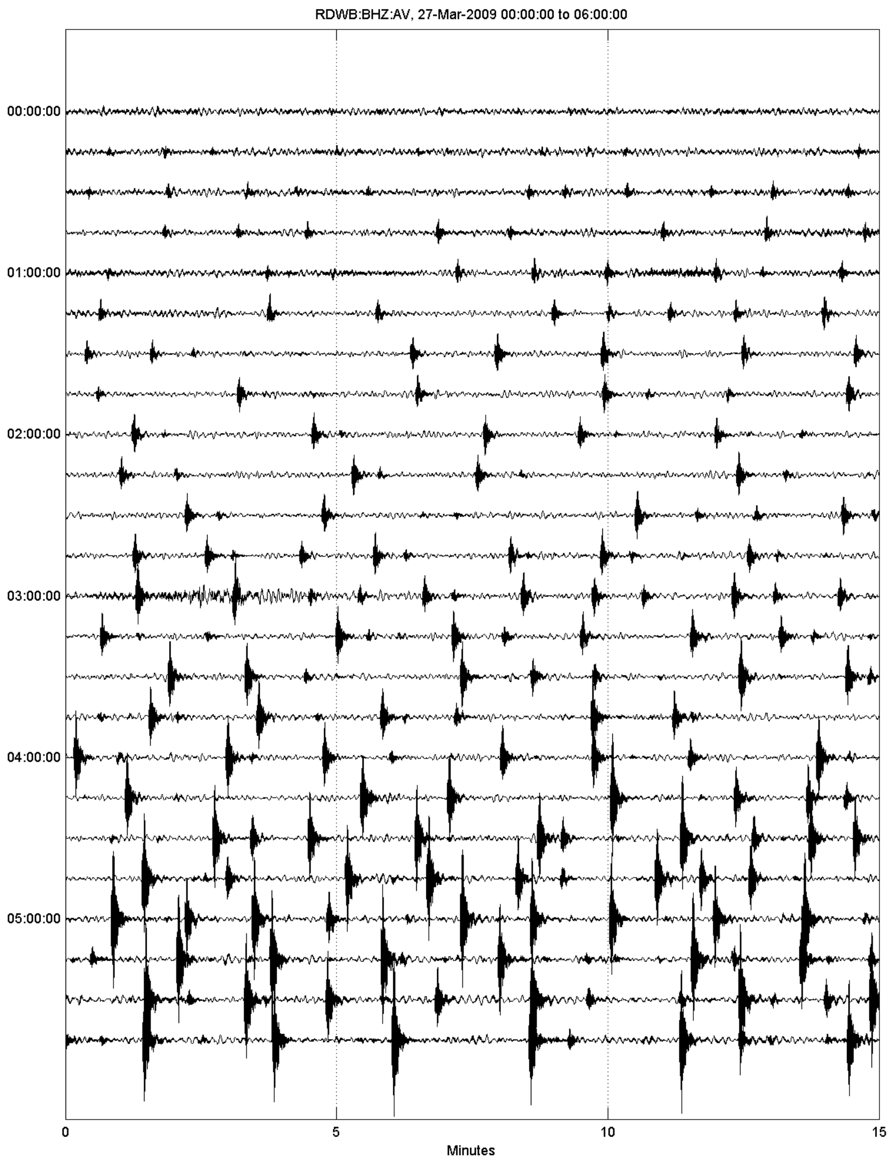

Figure 6: RDWB:BHZ helicorder

We suspect that there may be VLP energy in this data set, so we integrate `w` data to move from velocity to displacement. A second waveform `w_vlp` is created by filtering `w` over the VLP range (whatever you believe that might be). Next we combine `w` and `w_vlp` as follows: `w = [w w_vlp]`. This guarantees that VLP data is plotted on top of displacement data as seen in Figure 7. We aren't satisfied with the default trace colors so we set them ourselves:

```
color = {[0 0 0],[1 0 0]}; % Normalized RGB value for black and red
build(helicorder(w,'mpl',15,'trace_color',color))
```

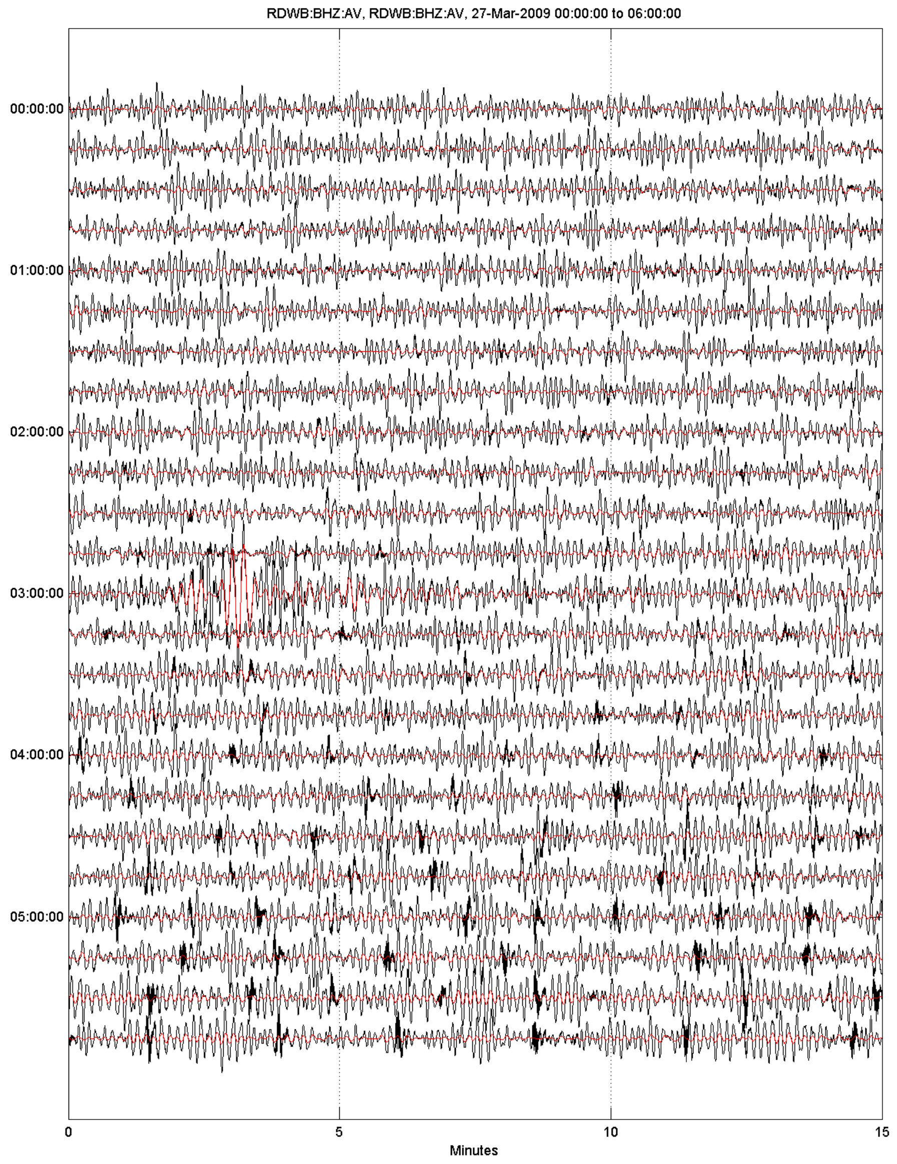

Figure 7: Helicorder VLP overlay

2.4 Valid helicorder property name/value combinations

- `e_sst` - Event Start/Stop Times
  If wave contains only one wavefrom object, `e_sst` can be entered as an Nx2 array of matlab times. If there is more than one waveform in wave (M waveforms), `e_sst` must be a 1xM cell array, each cell containing Nx2 array of start/stop times. A single waveform can also have multiple sets of start/stop times as in a waveform with multiple event families. In this case `e_sst` should be entered as a 1xL cell array for a particular waveform (with L separate groups to be highlighted). EXAMPLE: `h = helicorder(w,'e_sst',A)` where `w` is a 1x2 waveform object
  `A` is a 1x2 cell array
  `A(1,2)` is a 1x3 cell
  `A(1,2){1}` is a Nx2 numeric array
  `A(1,2){2}` is a Nx2 numeric array
  `A(1,2){3}` is a Nx2 numeric array
  DEFAULT = [] (No events)

- `mpl` - Minutes Per Line
  single numeric value specifying number of minutes per helicorder line. (This applies to all waveforms in the helicorder)
  DEFAULT = 10

- `trace_color` - Color Scheme of Trace Data
  If `w` contains only one wavefrom object, `trace_color` can be entered as a 1x3 array of RGB values (between 0 and 1). For wave arguments longer than 1, `trace_color` should be entered as a 1xN cell array, each containing a 1x3 array of RGB values.

- `event_color` - Color Scheme of Highlighted Event Data
  Only use this property when `e_sst` property exists. `event_color` follows the same structure as `e_sst`. If wave contains only one waveform object, `event_color` can be entered as a 1x3 array of RGB values (between 0 and 1). For wave arguments longer than 1, `event_color` should be entered as a 1xN cell array, each containing a 1x3 array of RGB values. If multiple event sets exist for a single waveform, `event_color` must be specified accordingly.
  EXAMPLE: `h = helicorder(w,'e_sst',A,'event_color',B)`
  `A` is the same as the previous example
  `B(1,1)` is a 1x3 RGB array
  `A(1,2)` is a 1x3 cell
  `A(1,2){1}` is a 1x3 RGB array
  `A(1,2){2}` is a 1x3 RGB array
  `A(1,2){3}` is a 1x3 RGB array
  DEFAULT = [] (No events)

- `display` - Multi-Waveform Display Type
  `'single'` - Used when there is only one waveform in wave. None of the multi-waveform display types can be set unless multiple waveforms are passed.
  `'stack'` - Plots multiple waveforms over top of each other. (Multiple motivations could exist for wanting to display data in this way such as overlaying VLP energy).
  `'alternate'` - Alternate through waveforms (grouped by time).
  `'group'` - Plot all trace data in a waveform before starting the next (grouped by waveform source).
  DEFAULT = 'single', 'alternate' (1 or multiple waveforms)

2.5 Interacting with a helicorder figure

The `build` fuction also provides a degree of interactivity with the helicorder figure. Clicking the cursor on any trace in the helicorder figure will call the function `traClick` from within `build`. The user can define what takes place after a helicorder trace line, or a highlighted event is clicked. By default, the GUI `sst_pick` is opened for the clicked waveform generating a minute-long waveform and spectral display.

3 Event Detection Using STA/LTA

The most common technique used in detecting discrete seismic events on both single and multiple stations is Short-Term Average, Long-Term Average Ratio (STA/LTA). In this approach, a short window of length L_{sta} overlaps a longer window of length L_{lta}. The two windows share the same leading edge and are continuously updated as they are slid temporally through a seismic signal. An event is triggered when the time-averaged amplitude of the short data window A_{sta} divided by the time-averaged amplitude of the long data window A_{lta} exceeds a user-defined threshold t_{on}. When the ratio A_{sta}/A_{lta} drops below a second user-defined threshold t_{off}, the detected event is considered to be over. Post-trigger discriminators are often used alongside this mechanism to either keep or discard the triggered events. Common examples of such discriminators are minimum allowable duration DUR_{min} and minimum allowable root mean square RMS_{min}, criteria by which events that are too short or too low in amplitude respectively would be filtered out of the resulting set of events.

The use of STA/LTA is a simple and imperfect technique for automatically recording and quantifying discrete earthquakes. The algorithm doesn't begin to approach the abilities of the human brain in recognizing patterns, especially across a variety of different types of seismicity. To select tens of thousands of discrete events by hand, however, is less than practical. Speed is the primary advantage to using an automated approach. User-defined settings represent the controlled input of a system that can be tuned to optimize performance. On one side of the performance spectrum, STA/LTA sensitivity is set high, resulting in a very large array of detected events including an abundance of false triggers from seismic and telemetry noise. On the other end, we see a neat and small set of large amplitude events with many of the medium-sized and less impulsive earthquakes completely vacated.

The function `sta_lta` implements this algorithm for waveform object inputs. Like many other functions in SEMS, `sta_lta` has a mandatory waveform input as well as a number of property values that can be customized by the user. The first is the STA/LTA Event Detection Parameters (EDP) which are contained in a 1x8 numeric array. Calibrabting EDP for a given station may require some trial and error. The order of these parameters along with descriptions are included in the code below, and a more in depth discussion on STA/LTA parameter setting can be found in [1].

```
l_sta = 1;      % STA window length (s)
l_lta = 8;      % LTA window length (s)
t_on = 2;       % STA/LTA trigger on threshold
t_off = 1.6;    % STA/LTA trigger off threshold
skip_int = 0;   % Skip ahead after end of event (s)
min_dur = 1.7;  % Minimum event duration (s)
pre_t = 0;      % Added pre-event time (s)
post_t = 0;     % Added post-event time (s)

your_edp = [l_sta l_lta th_on th_off skip_int min_dur pre_t post_t];
event_sst = sta_lta(w,'edp',your_edp)
% The above EDP values are the default values,
% which makes the above code equivalent to:
event_sst = sta_lta(w);
```

The user may also define the output type of `sta_lta` to be a waveform array as opposed to the default type (start/stop times). Someone interested in only examining events on a helicorder, or archiving times might choose the default. Returning events as a waveform array, however, can fascilitate post-detection descrimination or analysis. Examples of this might be event RMS, peak frequency, or cross-correlation. User-defined return type is demonstrated below.

```
events = sta_lta(w); % returns events as start/stop times
events = sta_lta(w,'return','wfa'); % returns events as waveform array
```

If there exist periods in waveform `w` that the user wishes the STA/LTA algorithm to skip, then the 'skip' property can be used: `events = sta_lta(w,'skip',skip_sst)`. Here, `skip_sst` is a Nx2 array of start/stop times defining unwanted pieces of waveform `w`. These pieces may be calibration pulses, noise, or other artifacts. `sta_lta` will covert those periods to NaN values before event detection begins. When `sta_lta` encounters a section of NaN values, it simply advances the detection window forward in time until it is out of the NaN gap, then resumes detection.

The final property that can be specified by the user is post-trigger LTA window behavior. The first behavior, 'frozen', involves holding the position of the LTA window static while the STA window moves forward through time. In this case, the ratio $A_{sta}/A_{lta}(t)$ simply becomes a function of $A_{sta}(t)$. In the second behavior, 'continuous', the LTA window advances with the STA window and both are updated in the same fashion as before an event trigger occurred. Both of these LTA settings offer some distinct advantage over the other, but either can also incur undesired effects. The frozen setting has the advantage of capturing more completely the coda from events, especially those higher in energy with longer durations. Continuous LTA behavior will often flip an event trigger off prematurely once a significant amount of the LTA window overlaps the event. This can especially be an issue when L_{lta} is not long. The primary disadvantage of the frozen setting, however, is the possibility for turning a trigger on that can't be turned off. The presence of spiky and noisy data, data gaps, and a background seismic level that changes rapidly over a short duration can initiate a trigger in the presence of a low A_{sta} making it difficult for the A_{sta}/A_{lta} value to drop below t_{off}.

The third and final post-trigger LTA window behavior is called 'grow' and represents a compromise between both 'frozen' and 'continuous'. In this scheme, the tail end of the LTA window is frozen, and the leading-edge of the LTA window is continuous. This means that L_{lta} is expanding and A_{lta} is averaging more and more data. When the trigger is turned off, the LTA resumes its original length. The default method is 'frozen'. All three methods are invoked in the code below, then displayed

```
e_1 = sta_lta(w,'lta_mode','frozen');      % Default, equivalent to: e_1 = sta_lta(w)
e_2 = sta_lta(w,'lta_mode','continuous'); % Continuous mode
e_3 = sta_lta(w,'lta_mode','grow');        % Grow mode
w = [w w w];                        % Replicate w for helicorder plotting
e = {e_1 e_2 e_3};                  % Place events in cell array, length(w)=length(e)
build(helicorder(w,'e_sst',e))      % helicorder magic
```

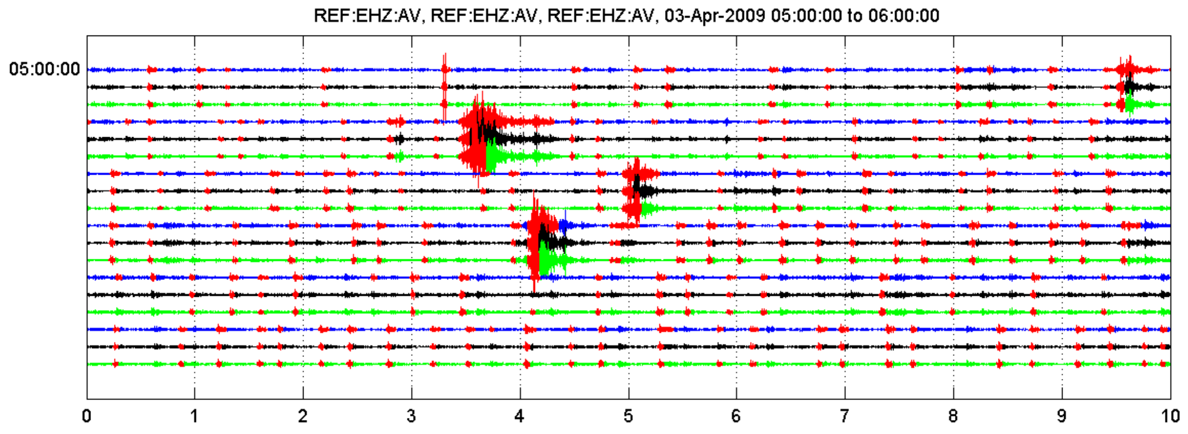

Figure 8: 'Frozen', 'Continuous',and 'Grow' LTA modes

4 Event Pick

The GUI `event_pick` can be used to manually pick event start/stop times from a waveform object `w`. The GUI creates a split display of waveform trace data and waveform spectral data seen in Figure 9. The user can rapidly scroll through `w` using the left and right arrow keys, as well as zoom in and out with the up and down arrow keys. Each pair of mouse clicks will highlight an event to be later output by the GUI. The command `sst1 = event_pick(w)` will open the GUI seen in Figure 9.

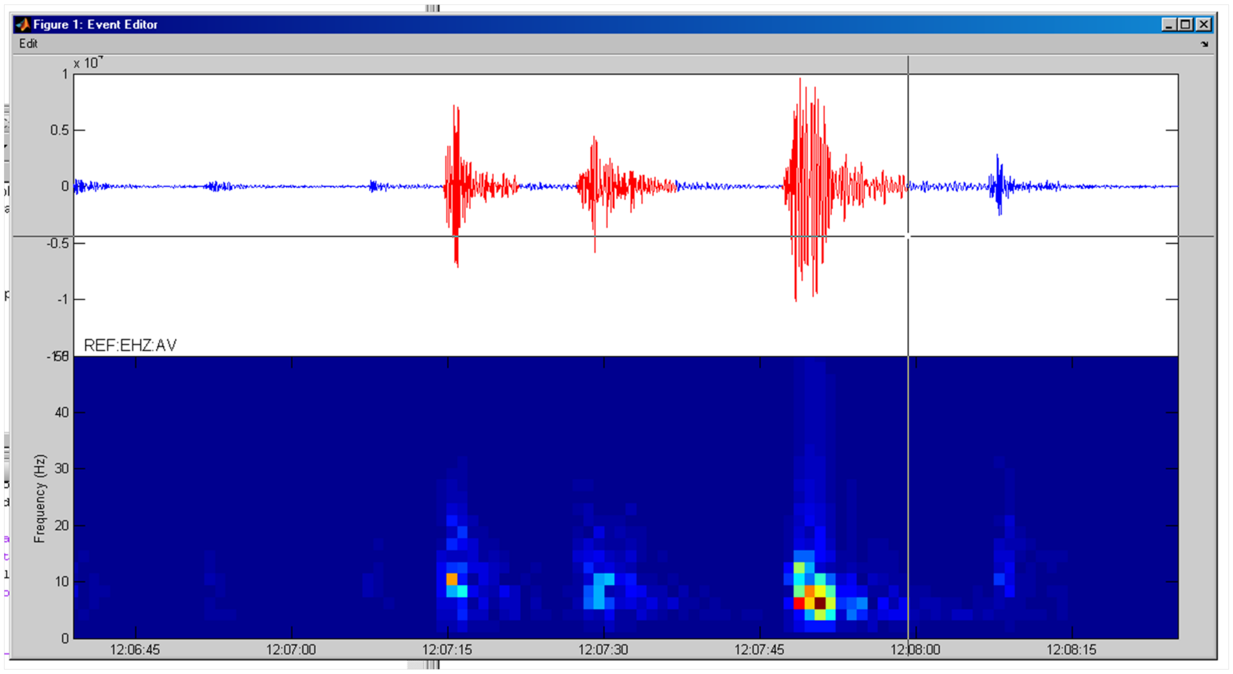

Figure 9: Event Pick GUI

5 Event Metrics

A few metrics are available in SEMS by which a single numeric value can be used to describe a particular event. Root Mean Square (RMS) is one such metric that you are probably familar with. This is an average measure of the energy within the event waveform. SEMS offers a way to quickly compute, display, and interact with this information. All of the event metric functions can provide the user with a numerical array output, or can directly generate an interactive figure. As with the helicorder figure, data points in any event metric function can be clicked to execute code from within that function. For a large set of automatically detected events, an event metric plot might provide relevent seismic trend information. It also provides a quick view of outlier data. The following four metrics are currently available in SEMS:

- `event_rms` - Event RMS ()
  `erms = event_rms(e_wfa,op)`
- `event_space` - Inter-event spacing (Temporal distace between event start times)
  `es = event_space(e_sst,unit,op)`
- `peak_freq` - Peak Frequency (Most energetic single frequency present via the Discrete Fourier Transform)
  `pf = peak_freq(e_wfa,op)`
- `freq_index` - Frequency Index (Ratio of low to high frequency content as defined in [2])

References

[1] A. Trnkoczy, “Understanding and parameter setting of STA/LTA trigger algorithm,” *IASPEI New Manual of Seismological Observatory Practice*, 2002.

[2] H. Buurman and M. West, “Seismic precursors to volcanic explosions during the 2006 eruption of Augustine Volcano,” in *The 2006 eruption of Augustine Volcano, Alaska: U.S. Geological Survey Professional Paper 1769* (J. Power, M. Coombs, and J. Freymueller, eds.), ch. 2, 2010.