COMPUTING SIMILARITY ATTRIBUTES – PROGRAM similarity3d

Contents

Overview

One of the most useful volumetric attributes is to compute the reflector continuity along structural dip and azimuth. Originally called "coherence" by developers at Amoco in the mid-1990s, the coherence family of attributes now has many variations, some that are algorithmic, and some that are simply renaming to avoid conflicts with the original Amoco patents. Such names include generic names such similarity and dissimilarity as well as names more indicative of the implementation including crosscorrelation, semblance, Manhattan distance, eigenstructure, variance, phase discontinuity, chaos, Sobel filters, Laplacian filters, and so forth. Whatever you call it, a coherence algorithm compares adjacent waveforms along structure within an analysis window. Windows in which the data are similar to each other are considered to be coherent and plotted as white. Windows in which the data show significant variation are considered to be incoherent or dissimilar and plotted as gray and black. Such discontinuities provide a means of mapping both structural and stratigraphic edges, areas that are chaotic, such as mass transport complexes and karst collapse, and areas that are contaminated by either random or crosscutting coherent noise. With the exception of salt welds, there are no coherent reflectors internal to a salt dome, so they too will appear to be incoherent.

Program similarity3d provides a wide range of coherence algorithms including those based on semblance, eigenstructure analysis, and Sobel filters. Program similarity3d also outputs estimates of the total and coherent energy within a 3D analysis window, as well as lateral gradients and Laplacian filters along structure of the coherent amplitude. All computations are made along structure using the analytic (complex) trace. The algorithms can be combined with a suite of filter banks to generate attributes from bandpassed filtered data. Stacking the covariance matrix of the bandpassed filtered versions of the data provides a robust multispectral attribute analysis.

Computation flow chart

The input to program **similarity3d** includes inline and crossline components of reflector dip and a seismic amplitude volume. The inline and crossline components of dip are either the original components computed from program **dip3d,** or a filtered version computed from program **filter_dip_components**. Likewise, the seismic amplitude can be the original seismic amplitude, a structure-oriented filtered amplitude from program **sof3d**, spectrally balanced amplitude from program **spec_cmp** or **spec_cwt**, or even impedance inversion computed using a commercial software package. The output files include several types of similarity (coherence), the total and coherent energy of the analysis window, as well as inline and crossline coherent energy gradient components useful for direct analysis or as input to subsequent amplitude curvature computations. As of January 2017, one can compute these attributes for a suite of filter banks and combine the results to compute "multispectral" attributes.

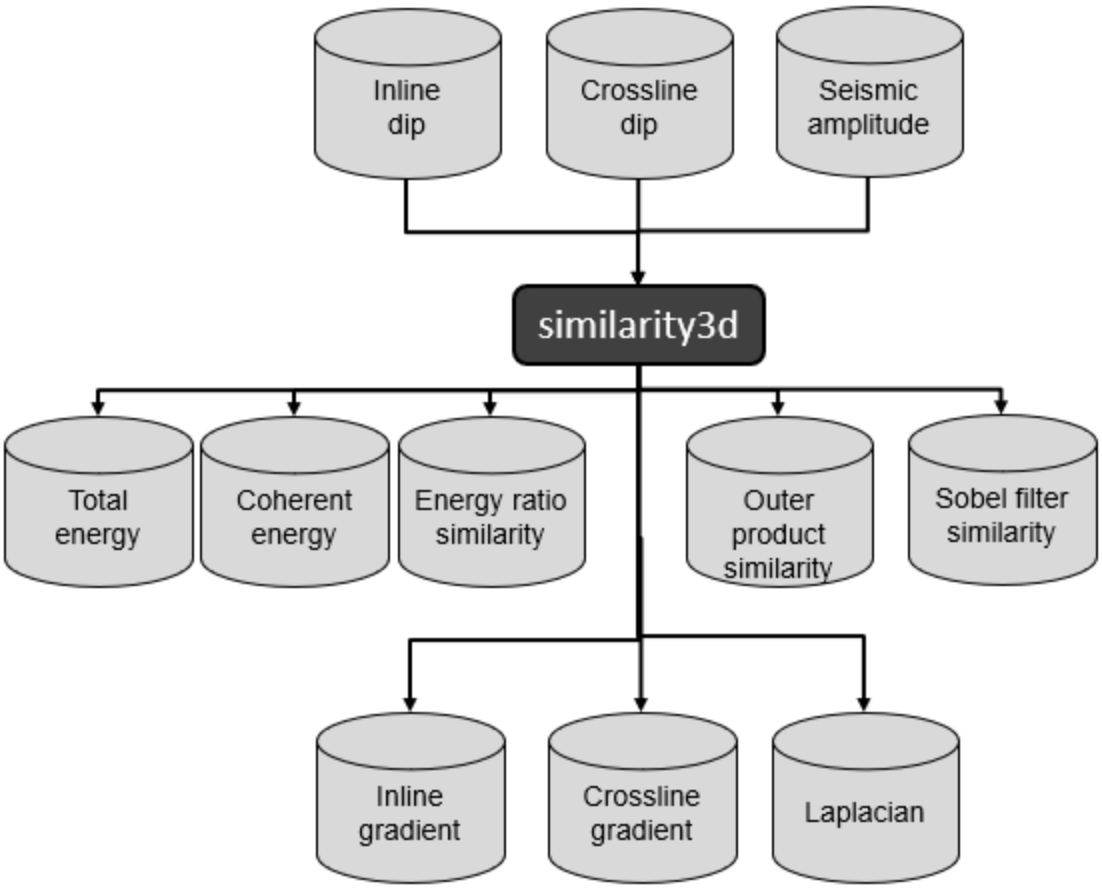


Figure 1.

Output file naming convention

Program **similarity3d** will always generate the following output files:

Output file description	**File name syntax**
Program log information	similarity3d_*unique_project_name_suffix*.log
Program error/completion information	similarity3d_*unique_project_name_suffix*.err

where the values in red are defined by the program GUI. The errors we anticipated will be written to the **.err* file and be displayed in a pop-up window upon program termination. These errors, much of the input information, a description of intermediate variables, and any software trace-back errors will be contained in the **.log* file.

Geometric Attributes: Program **similarity3d**

Unless the defaults are modified, **similarity3d** will also generate these "broadband" output files computed over the entire bandwidth of the data provided:

Output file description	File name syntax
Broadband energy-ratio similarity	energy_ratio_similarity_*unique_project_name_suffix*_broadband.H
Broadband outer-product similarity (semblance)	outer_product_similarity_*unique_project_name_suffix*_broadband.H
Broadband Sobel-filter similarity	sobel_filter_similarity_*unique_project_name_suffix*_broadband.H

We can also generate attributes that are either functions of the energy or of the RMS amplitude within the analysis window. If we choose to *Output energy and weight gradients by coherent energy*, the following files will be called:

Output file description	File name syntax
The total energy in the analysis window	total_energy_*unique_project_name_suffix*.H
The coherent energy in the analysis window	coherent_energy_*unique_project_name_suffix*_broadband.H
Broadband inline energy-weighted gradient	inline_energy_weighted gradient_*unique_project_name_suffix*_broadband.H
Broadband crossline energy-weighted gradient	crossline_energy_weighted_gradient_*unique_project_name_suffix*_broadband.H
Broadband energy-weighted Laplacian	energy_weighted_laplacian_*unique_project_name_suffix*_broadband.H

If we choose to *Output RMS amplitude and weight gradients by coherent RMS amplitude*, the following files will be called:

Output file description	File name syntax
The total (original data) RMS amplitude in the analysis window	total_RMS_amplitude_*unique_project_name_suffix*.H
The coherent RMS amplitude in the analysis window	coherent_RMS_amplitude_*unique_project_name_suffix*_broadband.H
Broadband inline RMS amplitude weighted gradient	inline_RMS_amplitude_weighted_gradient_*unique_project_name_suffix*_broadband.H

Geometric Attributes: Program **similarity3d**

Broadband crossline RMS amplitude weighted gradient	crossline_RMS_amplitude_weighted_gradient_*unique_project_name_suffix*_broadband.H
Broadband RMS amplitude weighted Laplacian	RMS_amplitude_weighted_laplacian_*unique_project_name_suffix*_broadband.H

We can also compute multispectral attributes where the choice of energy-weighted or RMS amplitude weighted gradients is dependent on the chosen parameters:

Output file description	**File name syntax**
Multispectral energy-ratio similarity	energy_ratio_similarity_*unique_project_name_suffix*_multi-spectral.H
Multispectral outer-product similarity (semblance)	outer_product_similarity_*unique_project_name_suffix*_multi-spectral.H
Multispectral Sobel-filter similarity	sobel_filter_similarity_*unique_project_name_suffix*_multi-spectral.H
Multispectral inline energy-weighted gradient	inline_energy_weighted_gradient_*unique_project_name_suffix*_multi-spectral.H
Multispectral crossline energy-weighted gradient	crossline_energy_weighted_gradient_*unique_project_name_suffix*_multi-spectral.H
Multispectral inline RMS amplitude weighted gradient	inline_RMS_amplitude_weighted_gradient_*unique_project_name_suffix*_multi-spectral.H
Multispectral crossline RMS amplitude weighted gradient	crossline_RMS_amplitude_weighted_gradient_*unique_project_name_suffix*_multi-spectral.H
Multispectral energy weighted Laplacian	energy_weighted_laplacian_*unique_project_name_suffix*_multi-spectral.H
Multispectral RMS amplitude weighted Laplacian	RMS_amplitude_weighted_laplacian_*unique_project_name_suffix*_multi-spectral.H

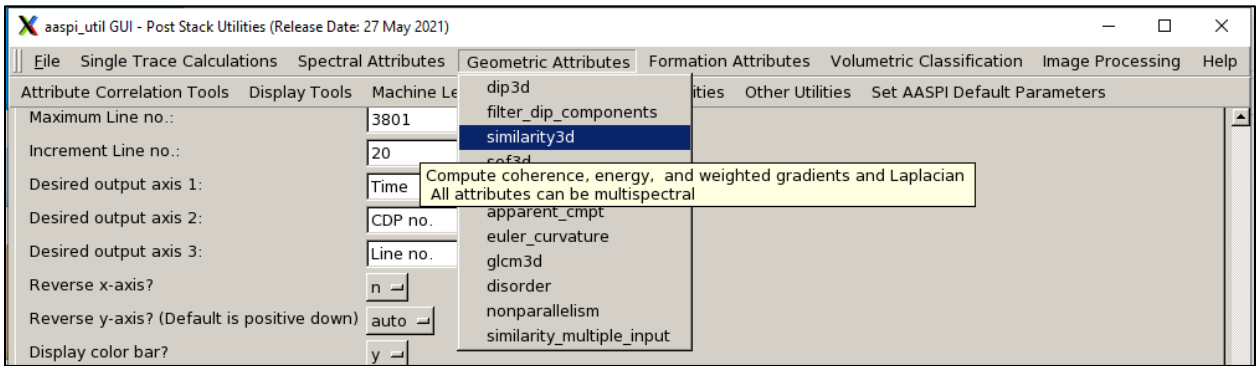
Filter bank and broad band spectra	filter_banks_similarity3d_ *unique_project_name_suffix*_.H

Finally, we may choose to compute attributes for a suite of filter banks defined by an Ormsby filter with corner points f_1, f_2, f_3, and f_4:

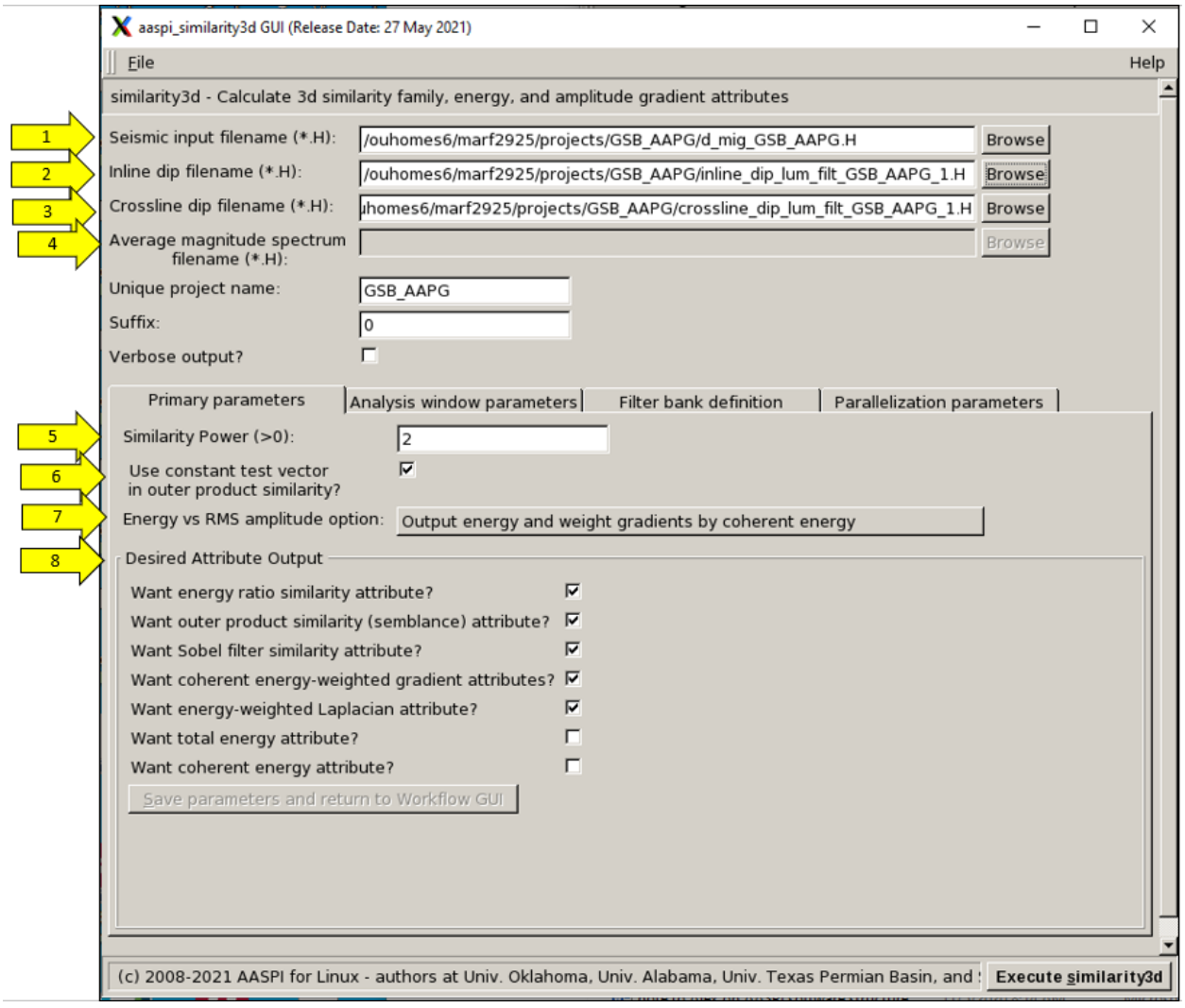
Output file description	**File name syntax**
Filter bank energy-ratio similarity	energy_ratio_similarity_*unique_project_name_suffix_f1_f2_f3_f4*.H
Filter bank outer-product similarity (semblance)	outer_product_similarity_*unique_project_name_suffix_f1_f2_f3_f4*.H
Filter bank Sobel-filter similarity	sobel_filter_similarity__*unique_project_name_suffix_f1_f2_f3_f4*.H
Filter bank Sobel-filter similarity	laplacian_filter_similarity__*unique_project_name_suffix_f1_f2_f3_f4*.H
Filter bank inline energy-weighted gradient	inline_energy_weighted_gradient_*unique_project_name_suffix_f1_f2_f3_f4*.H
Filter bank crossline energy-weighted gradient	crossline_energy_weighted_gradient_*unique_project_name_suffix_f1_f2_f3_f4*.H
Filter bank inline RMS amplitude weighted gradient	inline_RMS_amplitude_weighted_gradient_*unique_project_name_suffix_f1_f2_f3_f4*.H
Filter bank crossline component of RMS amplitude weighted amplitude gradient	crossline_RMS_amplitude_weighted_gradient_*unique_project_name_suffix_f1_f2_f3_f4*.H
Filter bank energy-weighted Laplacian	energy_weighted_laplacian_*unique_project_name_suffix_f1_f2_f3_f4*.H
Filter bank RMS amplitude weighted Laplacian	RMS_amplitude_weighted_laplacian_*unique_project_name_suffix_f1_f2_f3_f4*.H
Filter bank coherent energy	coherent_energy_*unique_project_name_suffix_f1_f2_f3_f4*.H
Filter bank coherent RMS amplitude	coherent_RMS_amplitude_*unique_project_name_suffix_f1_f2_f3_f4*.H

Computing similarity attributes

Program **similarity3d** is launched from the same *Geometric Attributes* tab as **dip3d** and **filter_dip_components** from the **aaspi_util** GUI under:



Double-clicking ***similarity3d*** generates the following GUI:



Use the browser on the first three lines to choose (1) the input seismic data file (*d_mig_GSB_small.H*), (2) the inline dip (*inline_dip_lum_filt_GSB_small_1.H)* and crossline dip (*crossline_dip_lum_filter_GSB_small_1.H*) files generated previously by program **dip3d** and subsequently filtered by program **filter_dip_components**. The (4) average power spectrum is an advanced option that allows the definition of a data-adaptive analysis window, which may be useful for depth-migrated data where the dominant wavelength changes significantly from shallow to deeper depths. For now, use a fixed window.

The Primary Parameters Tab

Similarity values will range between 0 and 1, with most of the values biased towards 1. If one (5) takes the power of the similarity, the distribution moves towards zero, providing improved contrast in the resulting image. This option remains from a time when many workstation software programs had only limited color bar manipulation capabilities. In modern interpretation software, it makes no difference whether one maps nonlinearly (powers) the similarity and plots it against a linear gray scale color bar, or if one plots the unscaled similarity against a nonlinear gray scale color bar.

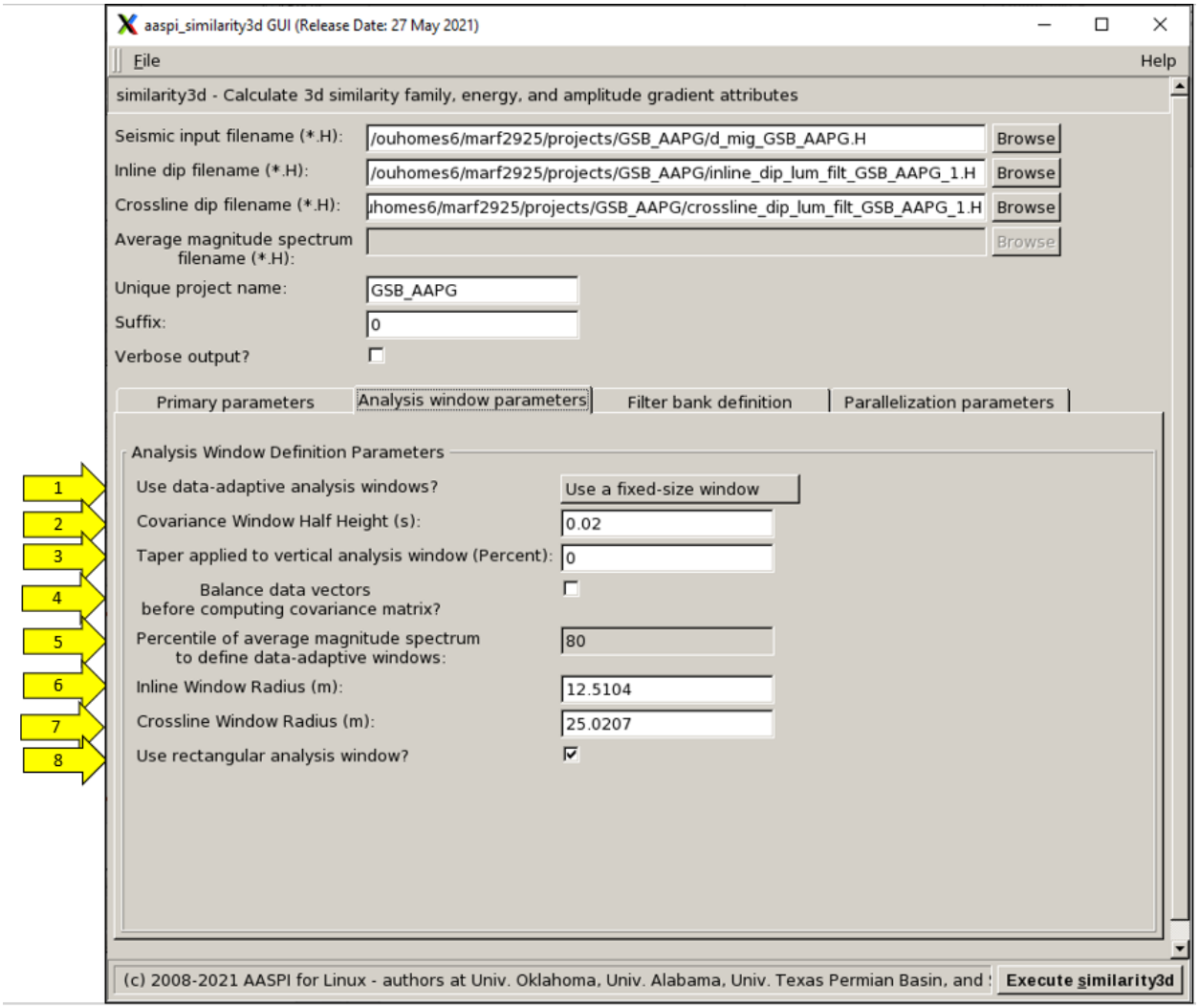
All but the Sobel filter similarity attribute are computed using a covariance matrix computed from the analytic (or complex trace). If one takes the outer product of the covariance matrix with the first eigenvector, the result is an eigenstructure coherence. In contrast, if one takes the outer product of the covariance matrix with a test vector that (6) has J equal components of $J^{-1/2}$, one obtains a semblance estimate of coherence. If this box is not checked, the algorithm computes the outer product of the sample vector passing through the analysis point. Mathematical details are provided in the theory boxes. By defining semblance in this manner one can compute the semblance in laterally and vertically tapered windows, which is critical to data-adaptive similarity computation.

The gradient of the eigenvector can be weighted either by the energy or by the square root of the energy (the RMS amplitude) of the coherent component of the data within the analysis window (see theory box). The RMS amplitude weighting option was added in April 2017 to provide an image that exhibits a comparable range to that of the original seismic amplitude data.

Moving down to (8) the *Desired Attribute Output* box, check those attributes you wish to compute. In this example, **similarity3d** will compute all of them to illustrate their use. However, when computing very large volumes, you may wish to limit your output to those attributes you find most useful. Be forewarned that if you wish to compute amplitude curvature, you need to generate the intermediate energy-weighted amplitude gradient output.

The Analysis Window Parameters Tab

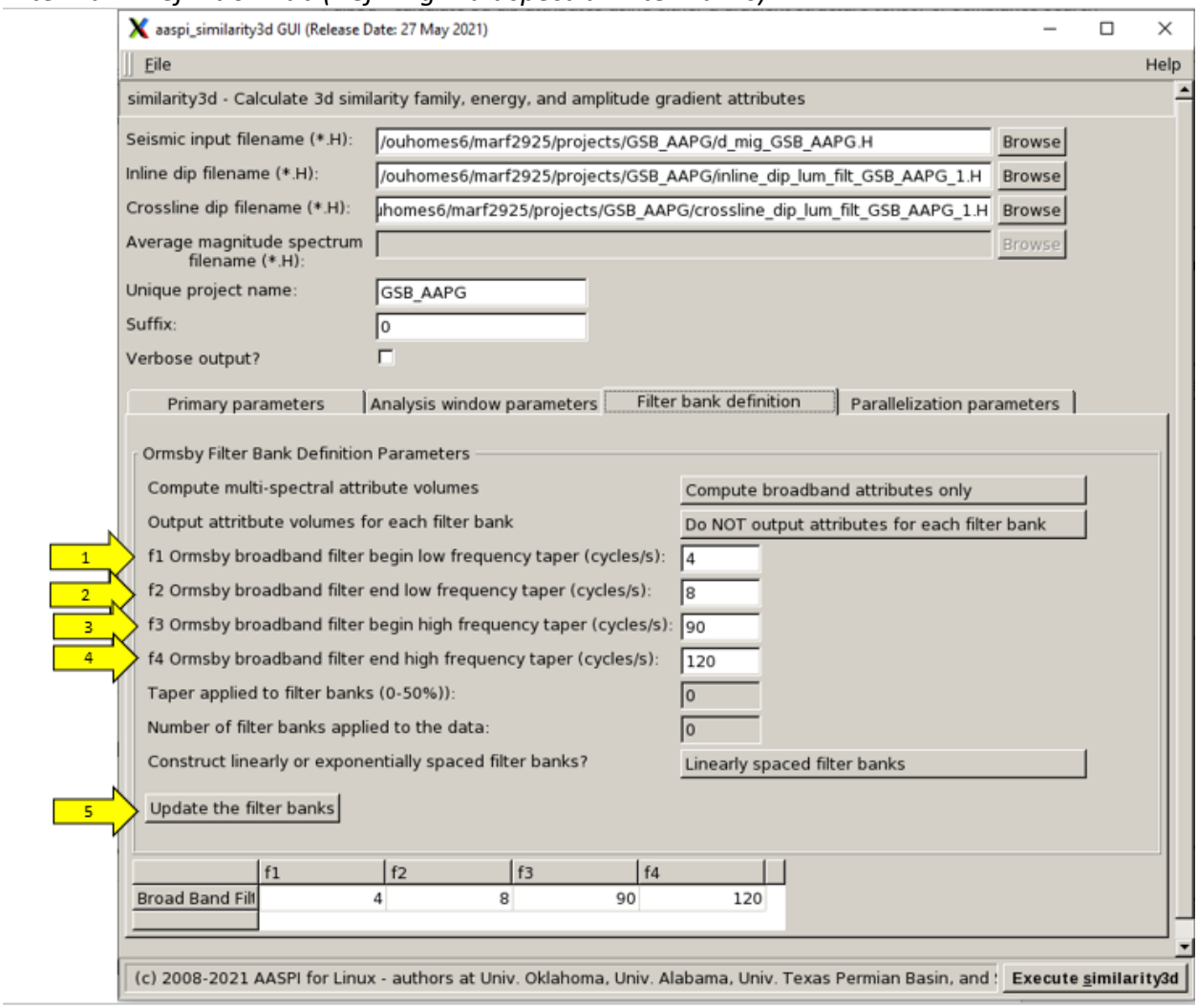
The image below shows the GUI when the *Analysis window parameters* tab has been selected. By default, **similarity3d** will use (1) a fixed-size analysis window. We will investigate what happens if we toggle this button shortly. For now, define the (2) covariance window half height, (3) percent tapers applied to the samples vertically, and (4) whether you wish to balance the covariance matrix sample vectors prior to constructing it (see the appropriate Theory box below). At present, since the *Use fixed-size window* has been selected the (5) reference frequency option is disabled. Drop down to (6) the inline and (7) crossline window radii. The default will be ±1 trace in each direction, which for the GSB survey are 12.5 and 25 m. The default is also to use (8) a rectangular vs. an elliptical analysis window. For small windows like this, the rectangular window provides more robust estimates of inline and crossline coherent energy gradient components.



The Filter Bank Definition Tab (Default Parameters)

Geometric Attributes: Program **similarity3d**

The image below shows the GUI when the *Filter bank definitions* tab has been selected. By default, **similarity3d** will use generate a single "broadband" volume for each of the selected volumes that has been band-pass filtered using an Ormsby filter defined by the four corner points (1) *f1*, (2) *f2*, (3) *f3*, and (4) *f4*. If you wish to change the values of these four corners points, be sure to click (5) *Update the filter banks*. The four corners points for the broad band filter are shown in the bottommost table. A description of the other parameters on this tab used to compute multispectral attributes can be found later in this documentation under *The Filter Bank Definition Tab (Defining Multispectral Filter Banks)*.



The Impact of Analysis Window Size

In general, the computation time increases linearly with the number of traces *M*, that fall within the analysis window, such that the cost increases as the square of the window radius. The increase in time with the analysis window height is smaller, since it only impacts the covariance matrix, and not the eigenvector and principal component computations. Larger lateral windows may also smear the results, leading to decreased lateral resolution. Larger vertical

windows may vertically mix stratigraphy. A good rule of thumb is to define the vertical and lateral analysis windows to be a fraction of the shortest period and wavelength. Windows larger than a period or wavelength not only smear or mix the results, they do so at increased cost. For high quality data, window heights of 0 s (one "sample") are permissible. This small value is possible since the covariance matrix is computed from the analytic trace, and thus has *two* sample vectors. Even for small vertical windows, an annoying stair-step artifact often occurs in similarity images. I will address this issue in the *Coherence Artifacts* section of this document.

Theory: Sample Vectors and the Covariance Matrix

Sample Vectors

The covariance matrix is constructed from a suite of sample vectors. In **similarity3d**, the "vectors" take the form of a suite of *2K+1 M*-trace maps parallel to structure, centered about the analysis point. The objective is to map vertically consistent (high coherence) or inconsistent (low coherence) patterns across the suite of local maps. To achieve this objective, one needs to compute a covariance matrix.

The Covariance Matrix

The covariance matrix, **C**, is constructed by comparing each sample vector to itself and all its neighbors**. similarity3d** uses not only the 2*K+1 M*-trace sample vectors through the original seismic amplitude, **d**, but also an additional *2K+1* sample vectors through its Hilbert transform, $\mathbf{d^H}$:

$$C_{mn} = \sum_{k=-K}^{K} \left[d(t_k, x_m, y_m) d(t_k, x_n, y_n) + d^H(t_k, x_m, y_m) d^H(t_k, x_n, y_n) \right]. \quad (1a)$$

These additional (90^0-phase rotated) sample vectors fall in the same window and thus do not modify the vertical resolution. However, they ameliorate areas of low signal-to-noise ratio about zero crossings, where the original absolute amplitude is smallest, but also where the corresponding Hilbert transform is largest, thereby suppressing low coherence artifacts commonly called "structural leakage" corresponding to amplitude zero crossings when viewed on coherence time slices. One can also taper the analysis windows using a weight w_k to minimize vertical edge effects

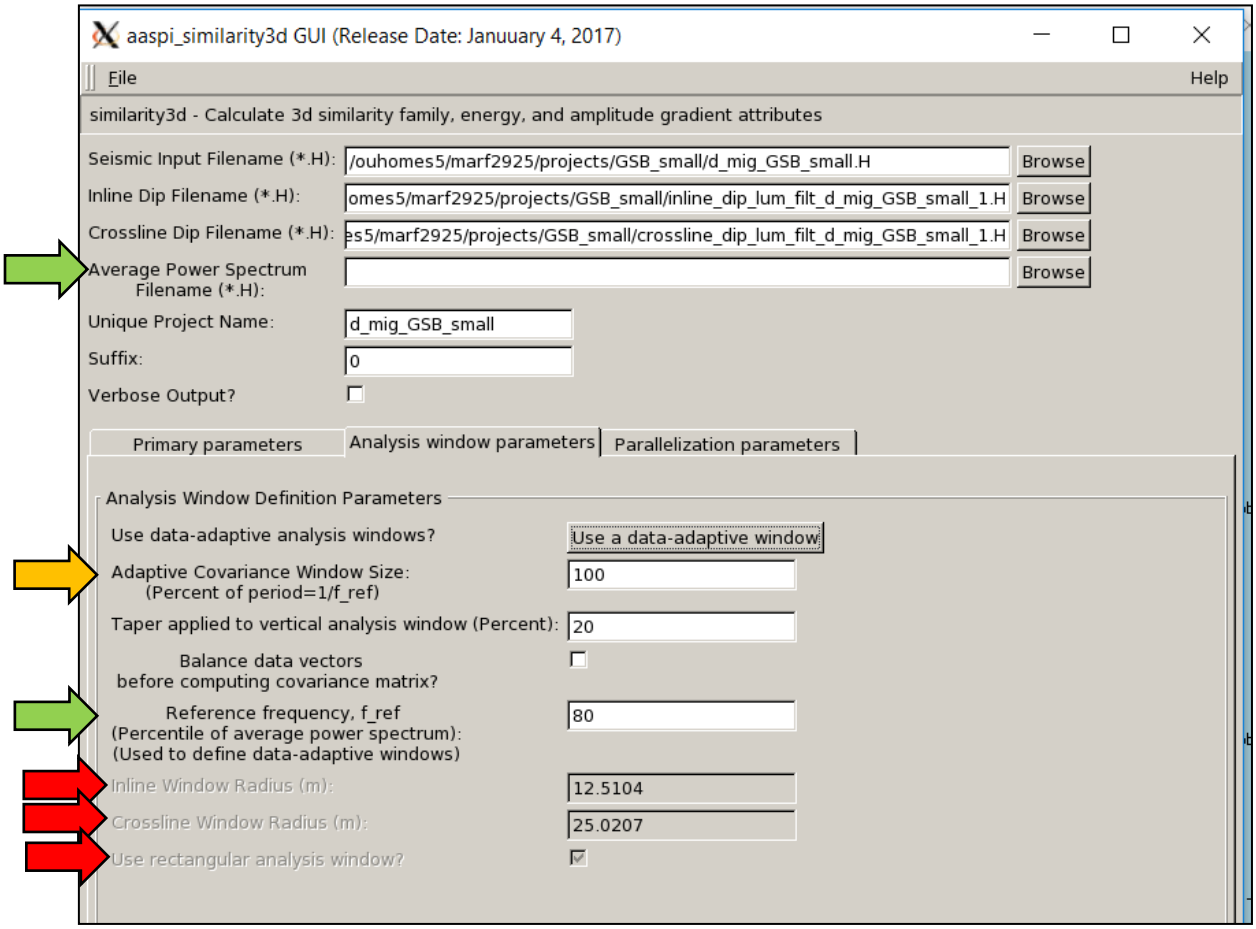
$$C_{mn} = \sum_{k=-K}^{K} w_k \left[d(t_k, x_m, y_m) d(t_k, x_n, y_n) + d^H(t_k, x_m, y_m) d^H(t_k, x_n, y_n) \right]. \quad (1b)$$

Anomalously large samples may dominate the covariance matrix, which for large vertical analysis windows may result in shallower or deeper strong events dominating the computation, giving rise to vertical smear. For this reason, there is an option *Balance sample vectors* that uses the RMS amplitude of each sample vector to ameliorate this problem.

Data Adaptive Windows

In his dissertation on attribute analysis of depth-migrated data, Lin (2016), realized that the significant change in wavelength from shallow to deep can cause significant problems. If the window is too small to encompass a smeared discontinuity, the geology of deeper, longer-

wavelength data may look overly continuous and thus exhibit higher coherence. Alternatively, if the size of the window is too large, shallower, shorter wavelength features accurately imaged in the seismic data may be undesirably smeared or mixed by the coherence operator. An ancillary question arises: when is a coherence anomaly significant (against the background values) and when is it insignificant? To address these issues, Lin et al. (2015) uses an *F*-test to assign levels of significance to the result, which required estimation of the data bandwidth and signal-to-noise ratio in addition to the size of the analysis window. In that work they used the AASPI program **disorder** to estimate the signal-to-noise ratio, which treats locally linear discontinuities as geologic signal.



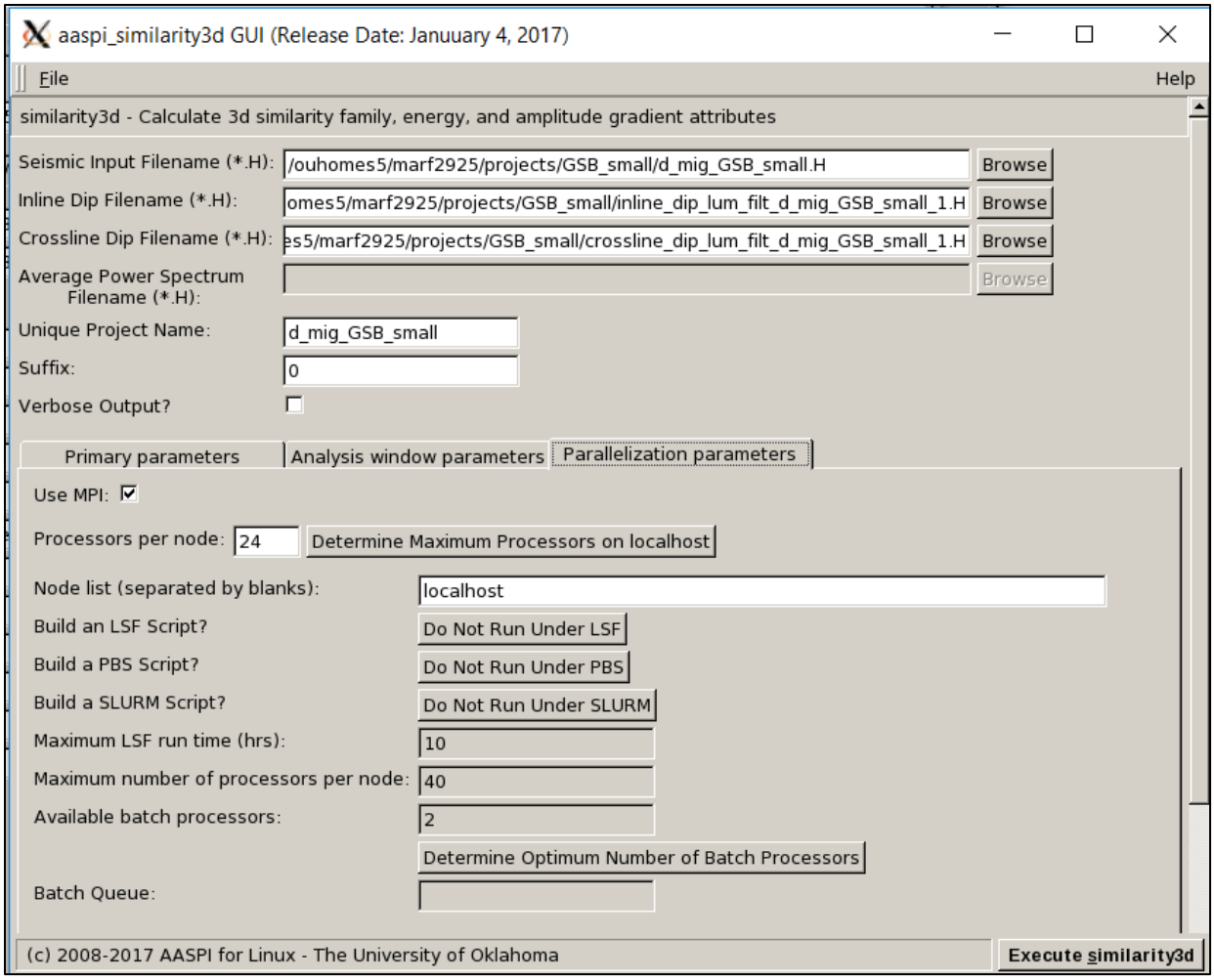
Clicking the *Use data-adaptive analysis* windows toggle changes the active areas of the GUI as shown above. First, several of the fixed window definition options are disabled (red arrows). Two of previously inactive options (green arrows) have been activated, the first of which is the *Average Power Spectrum Filename* created by a previous computation of **spec_cmp** or **spec_cwt**. The second previously inactive option, *Reference frequency, fref,* defines a percentile of the average spectrum as it changes with time or depth. For spectrally balanced data, resolution is determined not by the peak frequency (which in the ideal case of a perfectly flat spectrum is ill-defined), but by the highest useable frequency. In this example, this frequency was chosen to be the *p*=80 percentile of the spectrum at each time or depth sample. Given this (time and depth variant) frequency, one can define a corresponding period, and (using the

reference velocity for time-migrated data) the lateral wavelength. For depth-migrated data, wavelengths are used to define the window size in *x, y*, and *z.* The option that previously defined the temporal size of the fixed analysis window is now changed to be a percent of this period and/or wavelength (orange arrow). Since the *p*=80 frequency varies continuously, the size of the analysis window varies continuously, and thus needs to use tapered analysis windows in *x, y,* as well as in *z,* resulting in scaled traces. Lin (2016) shows that by applying these tapers to the covariance matrix, that one can generate accurate, yet flexible results for both energy ratio coherence and outer product similarity.

The Parallelization Parameters Tab

The parallelization parameters tab for **similarity3d** is the same for all AASPI applications. One can run across multiple processors and cores on a single node, across nodes if that is allowed by your IT department, or on large batch supercomputers using LSF, PBS, or SLURM scripts. Details on parallelization can be found in the parallelization section of the documentation *Overview: AASPI Software Parallelization*. **similarity3d** uses a stencil-based parallelization scheme; this results in suboptimal performance for surveys that do not approximate a rectangular shape, where cores assigned to dead or padded traces lie idle.

Geometric Attributes: Program **similarity3d**



After selecting all the parameters, you wish to change, click *Execute similarity3d* and generate several output files:

```
 1401 Jan  5 16:47 similarity3d.parms
   37 Jan  5 16:47 live_processor_list
 2325 Jan  5 16:47 inline_energy_gradient_GSB_small_0_broadband.H@@
 4981 Jan  5 16:47 inline_energy_gradient_GSB_small_0_broadband.H
 2328 Jan  5 16:47 crossline_energy_gradient_GSB_small_0_broadband.H@@
 4990 Jan  5 16:47 crossline_energy_gradient_GSB_small_0_broadband.H
 2316 Jan  5 16:47 sobel_filter_similarity_GSB_small_0.H@@
 4744 Jan  5 16:47 sobel_filter_similarity_GSB_small_0.H
 2327 Jan  5 16:47 outer_product_similarity_GSB_small_0_broadband.H@@
 4777 Jan  5 16:47 outer_product_similarity_GSB_small_0_broadband.H
 2326 Jan  5 16:47 energy_ratio_similarity_GSB_small_0_broadband.H@@
 4774 Jan  5 16:47 energy_ratio_similarity_GSB_small_0_broadband.H
 2313 Jan  5 16:47 filter_coeff_similarity3d_GSB_small_0.H@@
 4625 Jan  5 16:47 filter_coeff_similarity3d_GSB_small_0.H
82042 Jan  5 17:01 similarity3d_GSB_small_0.out
```

The *similarity3d.parms* file contains the parameters defined by the GUI, which in turn is read by a python script called aaspi_similarity.py that invokes the executable program **similarity3d** (see the documentation on AASPI software structure for more details). The *live_processor_list*

contains a list of the 24 cores that I wished to run on. This list is checked by the software to assure that they indeed exist and can be used.

Next in the above list appear the AASPI-format output files. The names echo the previously checked attributes under the *Desired Attribute Output tab*. Notice that several of these have the suffix "broadband" attached to them. These are attributes computed for the entire frequency band defined by the broadband filter entered in the *Filter bank definition* tab which is typically the original bandwidth of the input data. The last file listed, *similarity3d_GSB_small_0.out* is the "output" file that contains information internal to the program. This is the file to examine if the software encounters errors. The AASPI software team works hard to provide useful error messages for errors we have encountered to date. More important, the team works even harder to capture these potential errors in the interactive GUI where they can be readily addressed. The beginning of this **.out* file echoes the python script parameters:

```
item[use_mpi]=y
item[processors_per_node]=24
processors_per_node= 24
item[node_list]=localhost
item[use_mpi]=y
item[verbose]=n
item[input_fn]=/ouhomes5/marf2925/projects/GSB_small/d_mig_GSB_small.H
item[average_power_spectrum_fn]=
item[inline_dip_fn]=/ouhomes5/marf2925/projects/GSB_small/inline_dip_lum_filt_d_mig_GSB_small_1.H
item[crossline_dip_fn]=/ouhomes5/marf2925/projects/GSB_small/crossline_dip_lum_filt_d_mig_GSB_small_1.H
item[energy_ratio_similarity_fn]=energy_ratio_similarity_GSB_small_0.H
item[sobel_filter_similarity_fn]=sobel_filter_similarity_GSB_small_0.H
item[outer_product_similarity_fn]=outer_product_similarity_GSB_small_0.H
item[inline_energy_gradient_fn]=inline_energy_gradient_GSB_small_0.H
item[crossline_energy_gradient_fn]=crossline_energy_gradient_GSB_small_0.H
item[coherent_energy_fn]=
item[total_energy_fn]=
item[filter_coeff_fn]=filter_coeff_similarity3d_GSB_small_0.H
item[error_fn]=similarity3d_GSB_small_0.err
item[inline_window_radius]=12.5104
item[crossline_window_radius]=25.0207
item[unique_project_name]=GSB_small
item[suffix]=0
item[cov_window_height]=0.02
item[f_ref_percentile]=80
item[percent_taper]=20
item[dtheta_interp]=0.25
item[similarity_power]=2
item[similarity_mean]=0
item[constant_vector]=y
item[remove_data_mean] does NOT exist!
item[wr_band_limited_attributes]=n
item[wr_multispectral_attributes]=n
item[f_low]=5
item[f_high]=120
item[f_width]=0
item[balance_cov]=n
item[output_coh_at_envelope_peaks] does NOT exist!
item[percentile_avg_spectrum] does NOT exist!
item[use_adaptive_windows]=n
item[full_window_energy] does NOT exist!
item[rectangular_window]=y
item[build_lsf_script]=n
item[build_pbs_script]=n
```

which are then put into the calling arguments of the message-passing interface routine **mpirun**:

```
[mpirun]
[--machinefile]
[./live_processor_list]
[-v]
[/ouhomes/marf2925/AASPI_GIT/bin64/similarity3d]
["average_power_spectrum_fn="]
["balance_cov=n"]
["coherent_energy_fn="]
["constant_vector=y"]
["cov_window_height=0.02"]
["crossline_dip_fn=/ouhomes5/marf2925/projects/GSB_small/crossline_dip_lum_filt_d_mig_GSB_small_1.H"]
["crossline_energy_gradient_fn=crossline_energy_gradient_GSB_small_0.H"]
["crossline_window_radius=25.0207"]
["dtheta_interp=0.25"]
["energy_ratio_similarity_fn=energy_ratio_similarity_GSB_small_0.H"]
["error_fn=similarity3d_GSB_small_0.err"]
["f_high=120"]
["f_low=5"]
["f_ref_percentile=80"]
["f_width=0"]
["filter_coeff_fn=filter_coeff_similarity3d_GSB_small_0.H"]
["inline_dip_fn=/ouhomes5/marf2925/projects/GSB_small/inline_dip_lum_filt_d_mig_GSB_small_1.H"]
["inline_energy_gradient_fn=inline_energy_gradient_GSB_small_0.H"]
["inline_window_radius=12.5104"]
["input_fn=/ouhomes5/marf2925/projects/GSB_small/d_mig_GSB_small.H"]
["outer_product_similarity_fn=outer_product_similarity_GSB_small_0.H"]
["percent_taper=20"]
["rectangular_window=y"]
["similarity_mean=0"]
["similarity_power=2"]
["sobel_filter_similarity_fn=sobel_filter_similarity_GSB_small_0.H"]
["suffix=0"]
["total_energy_fn="]
["unique_project_name=GSB_small"]
["use_adaptive_windows=n"]
["use_mpi=y"]
["verbose=n"]
["wr_band_limited_attributes=n"]
["wr_multispectral_attributes=n"]


         18 send process_name =
 tripolite.ou.edu



         19 send process_name =
 tripolite.ou.edu



         21 send process_name =
 tripolite.ou.edu



         22 send process_name =
 tripolite.ou.edu



 this program will be run as multiple processes under MPI!
number of processes                    25
          0 receive process_name =
 tripolite.ou.edu
```

Lower down in the image above, note that some of the processors declare that they been accessed. Internal to the output, the software echoes out its progress (all this output also appears in the Linux xterm or Windows window from which **aaspi_util** was launched):

```
0: first_line_out,current_line,last_line_out,ETA     2501    2511    3801     0.290 h
0: first_line_out,current_line,last_line_out,ETA     2501    2513    3801     0.293 h
0: first_line_out,current_line,last_line_out,ETA     2501    2515    3801     0.292 h
0: first_line_out,current_line,last_line_out,ETA     2501    2517    3801     0.292 h
0: first_line_out,current_line,last_line_out,ETA     2501    2519    3801     0.291 h
0: first_line_out,current_line,last_line_out,ETA     2501    2521    3801     0.290 h
0: first_line_out,current_line,last_line_out,ETA     2501    2523    3801     0.287 h
0: first_line_out,current_line,last_line_out,ETA     2501    2525    3801     0.284 h
0: first_line_out,current_line,last_line_out,ETA     2501    2527    3801     0.283 h
0: first_line_out,current_line,last_line_out,ETA     2501    2529    3801     0.282 h
0: first_line_out,current_line,last_line_out,ETA     2501    2531    3801     0.280 h
0: first_line_out,current_line,last_line_out,ETA     2501    2533    3801     0.279 h
0: first_line_out,current_line,last_line_out,ETA     2501    2535    3801     0.278 h
0: first_line_out,current_line,last_line_out,ETA     2501    2537    3801     0.277 h
0: first_line_out,current_line,last_line_out,ETA     2501    2539    3801     0.275 h
0: first_line_out,current_line,last_line_out,ETA     2501    2541    3801     0.274 h
0: first_line_out,current_line,last_line_out,ETA     2501    2543    3801     0.273 h
```

where we see the first and last line to be run, the current line, and the “estimated time of arrival” or ETA for job completion in hours. The AASPI programming convention is to preface any printout by the processor number followed by a colon, “:”. The lines in the printout above are all preceded by “0:”, indicating that these messages are generated by the master process.

At the bottom of the output file, you will find the files being closed by the master process, normal completion messages by all processes, and some statistics on the computation by each process. Note the prefixes ranging from “1:” to “24:” in the screen capture below, indicating normal completion on each of the 24 slaves. The times are wall-clock times, such that if other compute-intensive processes are using the cycles on even one process, inter-process synchronization will cause the entire program to slow down. In this case, the test program took 0.287 hr to complete running on 12 dual core processors (for a total of 24 CPUs).

Theory: Eigenvectors and Eigenvectors

While all students of linear algebra learn the mathematical definition and numerical computation of eigenvalues and eigenvectors, very few have learned their physical significance. Along with simple measures of statistics such as the mean, median, percentiles, and variance, eigenvectors and eigenvalues are among the most useful mathematical tools in seismic data analysis and have particular value in image processing and pattern recognition. Any matrix can be decomposed into eigenvalues and eigenvectors (Kirlin and Done, 1999). Other than for least-squares fitting, most of the matrices we encounter in attribute analysis will be square. The covariance matrices described in the previous theory box will be real, square, symmetric matrices resulting in non-negative eigenvalues. For square matrices, one can write:

$$\sum_{m=1}^{M} C_{nm} v_m^{(k)} = \lambda_k v_n^{(k)}, \tag{2}$$

or in matrix form

$$\mathbf{C}\mathbf{v}^{(k)} = \lambda_k \mathbf{v}^{(k)}, \tag{3}$$

where **C** is an *M*-by-M square covariance matrix, λ_k is the k^{th} eigenvalue and $\mathbf{v}^{(k)}$ is its corresponding eigenvector. In general, there are *M* eigenvalue-eigenvector pairs (equal to the number of traces used to construct **C**), though not all of them need to be different, and where some of the eigenvalues λ_k may equal 0, indicating a rank-deficient matrix. By convention, the eigenvectors are normalized to be vectors of unit length

$$\sum_{j=1}^{J} \left(v_j^{(k)} \right) = 1, \tag{4a}$$

while the eigenvalue-eigenvector pairs are sorted from largest to smallest

$$\left|\lambda_1\right| \geq \left|\lambda_2\right| \geq \ldots \geq \left|\lambda_J\right|. \tag{4b}$$

Be forewarned that the positive and negative direction of the eigenvector is undefined. This direction will be defined when forming a principal component.

Geometric Attributes: Program **similarity3d**

```
 19:                    read and scale data          0.000          0.000
 19:                      send data via MPI          0.000          0.000
 19:                   receive data via MPI          0.063          0.008
 19:                   send results via MPI          0.001          0.000
 19:                receive results via MPI          0.000          0.000
 19:                     Hilbert transform           0.001          0.000
 19:                   calculate cov matrix          0.000          0.000
 19:                  calculate eigenvectors         0.008          0.001
 19:                    project data onto v1          0.001          0.000
 19:                   write results to disk          0.000          0.000
 19:                              total time          0.287          0.039
        19 : memory residing only on slaves deallocated
        19 : shared arrays residing on both master and slave deallocated
        18 number of traces processed:       26691
 process                                    task       time (hr)  time/trace (s)
 18:                    read and scale data          0.000          0.000
 18:                      send data via MPI          0.000          0.000
 18:                   receive data via MPI          0.058          0.008
 18:                   send results via MPI          0.001          0.000
 18:                receive results via MPI          0.000          0.000
 18:                     Hilbert transform           0.001          0.000
 18:                   calculate cov matrix          0.000          0.000
 18:                  calculate eigenvectors         0.008          0.001
 18:                    project data onto v1          0.001          0.000
 18:                   write results to disk          0.000          0.000
 18:                              total time          0.287          0.039
        18 : memory residing only on slaves deallocated
        18 : shared arrays residing on both master and slave deallocated
otal data written to disk:     3258255 traces         251 samples
otal data written to disk:   -1023.679 Mbytes
ransfer rate            :      -8.684 Mbytes/s
         0 : memory deallocated residing only on master deallocated
         0 : shared arrays residing on both master and slave deallocated
         3 :normal completion. program similarity3d
         2 :normal completion. program similarity3d
         6 :normal completion. program similarity3d
         5 :normal completion. program similarity3d
         8 :normal completion. program similarity3d
         9 :normal completion. program similarity3d
         7 :normal completion. program similarity3d
        23 :normal completion. program similarity3d
         1 :normal completion. program similarity3d
        11 :normal completion. program similarity3d
        14 :normal completion. program similarity3d
        10 :normal completion. program similarity3d
        12 :normal completion. program similarity3d
        13 :normal completion. program similarity3d
        17 :normal completion. program similarity3d
        24 :normal completion. program similarity3d
        16 :normal completion. program similarity3d
        18 :normal completion. program similarity3d
        21 :normal completion. program similarity3d
        19 :normal completion. program similarity3d
         0 :normal completion. program similarity3d
losing file: /ouhomes5/marf2925/projects/GSB_small/crossline_dip_lum_filt_d_mig_GSB_small_1.H
       Closing file: /ouhomes/marf2925/AASPI_Data/crossline_dip_lum_filt_d_mig_GSB_small_1.H@
       Closing file: /ouhomes5/marf2925/projects/GSB_small/crossline_dip_lum_filt_d_mig_GSB_small_1.H
       Closing file: /ouhomes/marf2925/AASPI_Data/crossline_dip_lum_filt_d_mig_GSB_small_1.H@@
       Closing file: /ouhomes5/marf2925/projects/GSB_small/crossline_dip_lum_filt_d_mig_GSB_small_1.H@@
losing file: /ouhomes5/marf2925/projects/GSB_small/d_mig_GSB_small.H
       Closing file: /ouhomes/marf2925/AASPI_Data/d_mig_GSB_small.H@
```

The output contains a fairly long list of files:

Geometric Attributes: Program **similarity3d**

```
5014 Jan 10 08:19 sobel_filter_similarity_GSB_small_0_broadband.H
5059 Jan 10 08:19 sobel_filter_similarity_GSB_small_0___97.0-120.0-120.0-143.0.H
5059 Jan 10 08:19 sobel_filter_similarity_GSB_small_0___74.0--97.0--97.0-120.0.H
5059 Jan 10 08:19 sobel_filter_similarity_GSB_small_0___51.0--74.0--74.0--97.0.H
5059 Jan 10 08:19 sobel_filter_similarity_GSB_small_0____5.0--28.0--28.0--51.0.H
5059 Jan 10 08:19 sobel_filter_similarity_GSB_small_0___28.0--51.0--51.0--74.0.H
5059 Jan 10 08:19 sobel_filter_similarity_GSB_small_0____0.0---5.0---5.0--28.0.H
4873 Jan 10 08:19 outer_product_similarity_GSB_small_0____5.0--28.0--28.0--51.0.H
4873 Jan 10 08:19 outer_product_similarity_GSB_small_0____0.0---5.0---5.0--28.0.H
4840 Jan 10 08:19 energy_ratio_similarity_GSB_small_0_multi-spectral.H
4825 Jan 10 08:19 energy_ratio_similarity_GSB_small_0_broadband.H
4870 Jan 10 08:19 energy_ratio_similarity_GSB_small_0___97.0-120.0-120.0-143.0.H
4870 Jan 10 08:19 energy_ratio_similarity_GSB_small_0___74.0--97.0--97.0-120.0.H
4870 Jan 10 08:19 energy_ratio_similarity_GSB_small_0___51.0--74.0--74.0--97.0.H
4870 Jan 10 08:19 energy_ratio_similarity_GSB_small_0____5.0--28.0--28.0--51.0.H
4870 Jan 10 08:19 energy_ratio_similarity_GSB_small_0___28.0--51.0--51.0--74.0.H
4870 Jan 10 08:19 energy_ratio_similarity_GSB_small_0____0.0---5.0---5.0--28.0.H
4843 Jan 10 08:19 outer_product_similarity_GSB_small_0_multi-spectral.H
4828 Jan 10 08:19 outer_product_similarity_GSB_small_0_broadband.H
4873 Jan 10 08:19 outer_product_similarity_GSB_small_0___97.0-120.0-120.0-143.0.H
4873 Jan 10 08:19 outer_product_similarity_GSB_small_0___74.0--97.0--97.0-120.0.H
4873 Jan 10 08:19 outer_product_similarity_GSB_small_0___51.0--74.0--74.0--97.0.H
4873 Jan 10 08:19 outer_product_similarity_GSB_small_0___28.0--51.0--51.0--74.0.H
5050 Jan 10 08:19 inline_energy_gradient_GSB_small_0_multi-spectral.H
5035 Jan 10 08:19 inline_energy_gradient_GSB_small_0_broadband.H
5080 Jan 10 08:19 inline_energy_gradient_GSB_small_0___97.0-120.0-120.0-143.0.H
5080 Jan 10 08:19 inline_energy_gradient_GSB_small_0___74.0--97.0--97.0-120.0.H
5080 Jan 10 08:19 inline_energy_gradient_GSB_small_0___51.0--74.0--74.0--97.0.H
5080 Jan 10 08:19 inline_energy_gradient_GSB_small_0____5.0--28.0--28.0--51.0.H
5080 Jan 10 08:19 inline_energy_gradient_GSB_small_0___28.0--51.0--51.0--74.0.H
5080 Jan 10 08:19 inline_energy_gradient_GSB_small_0____0.0---5.0---5.0--28.0.H
5063 Jan 10 08:19 crossline_energy_gradient_GSB_small_0_multi-spectral.H
5048 Jan 10 08:19 crossline_energy_gradient_GSB_small_0_broadband.H
5093 Jan 10 08:19 crossline_energy_gradient_GSB_small_0___97.0-120.0-120.0-143.0.H
5093 Jan 10 08:19 crossline_energy_gradient_GSB_small_0___74.0--97.0--97.0-120.0.H
5093 Jan 10 08:19 crossline_energy_gradient_GSB_small_0___51.0--74.0--74.0--97.0.H
5093 Jan 10 08:19 crossline_energy_gradient_GSB_small_0____5.0--28.0--28.0--51.0.H
5093 Jan 10 08:19 crossline_energy_gradient_GSB_small_0___28.0--51.0--51.0--74.0.H
5093 Jan 10 08:19 crossline_energy_gradient_GSB_small_0____0.0---5.0---5.0--28.0.H
```

Where the attributes computed for each of the six filter banks have the values of the Ormsby filter appended to their names, a broad band result, and multi-spectral results where appropriate. At present, there is no multispectral result for total energy or coherent energy, though let us know if you think such output might be useful.

Examples

similarity3d provides a suite of different edge-detection attributes. The energy ratio similarity attribute produces results like the eigenstructure coherence algorithm described by Gersztenkorn and Marfurt (1999) with the difference that it is computed along structure and uses analytic traces (the original traces and their Hilbert transform or quadrature) rather than simply, the real traces in the computation. Numerically, energy ratio similarity is the ratio of the energy of the Karhunen-Loève filtered data over the total (unfiltered) energy of the input data within the analysis window. The total energy through a time slice at t=1.760 s through the GSB survey looks like the following image (see next page).

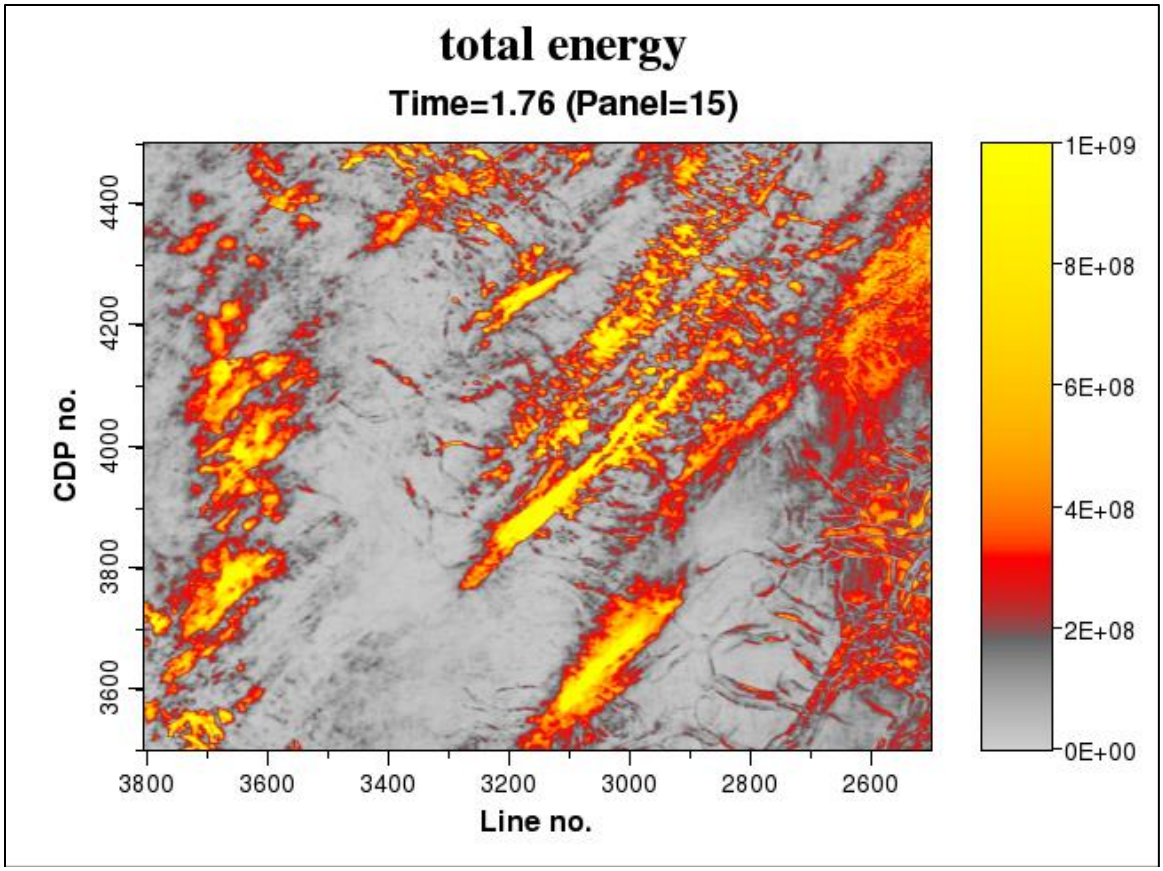


Figure 2.

The corresponding time slice through the coherent energy volume looks very similar:

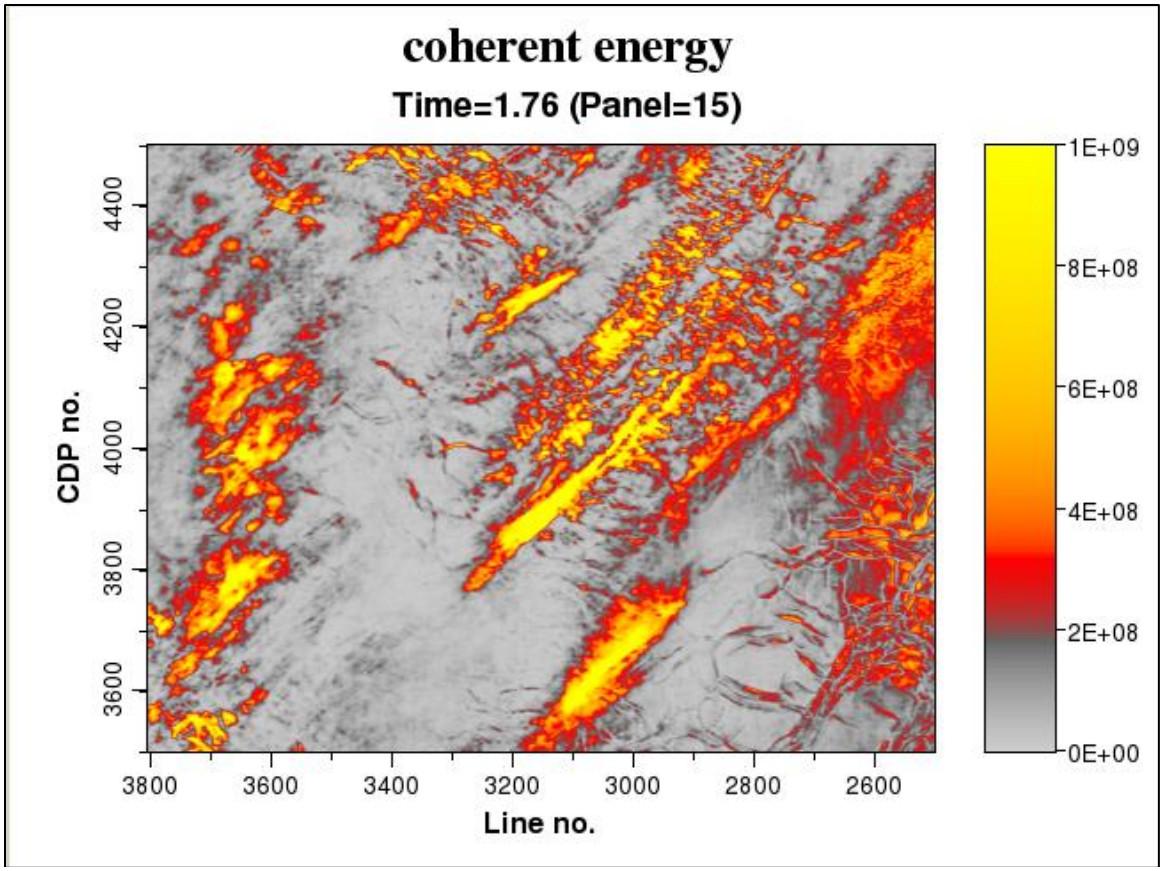

coherent energy
Time=1.76 (Panel=15)
CDP no.
4400
4200
4000
3800
3600
Line no.
3800
3600
3400
3200
3000
2800
2600
1E+09
8E+08
6E+08
4E+08
2E+08
0E+00

Figure 3.

Theory: Principal Components, Karhunen-Loève Filtering, and the Total vs. Coherent Energy Attributes

Because principal components are scaled versions of the eigenvectors, they are informally used interchangeably with each other. Recall that the eigenvectors have been scaled to be of unit length. In contrast, the j^{th} principal component is a scaled version of the j^{th} eigenvector, $\mathbf{v}^{(j)}$, where the scale factor is the inner product or correlation of the eigenvector (in this equation considered to be an eigenmap) with a sample vector (*M*-trace data slice) that contains the analysis point (t_0,x_n,y_n) at the center of the window:

$$d_{PC}^{(j)}(t_0,x_n,y_n)=\sum_{m=1}^{M}\left[v^{(j)}(x_m,y_m)d(t_0,x_m,y_m)\right]v^{(j)}(x_n,y_n), \tag{5}$$

where *M* is the number of traces in the analysis window and *J=M* is the total number of eigenvalue-eigenvector pairs. The data can be reconstructed by summing the principal components:

$$d_{PC-filt}(t_0,x_n,y_n)=\sum_{j=1}^{L}d_{PC}^{(j)}(t_0,x_n,y_n)\,. \tag{6}$$

If *L=J,* the total number of principal components, we reconstruct the original data. If *L<J*, we obtain a filtered version of the data, where the first few principal components represent the more correlated part of the data and the latter principal components represent successively less correlated parts of the data, such as random noise or cross-cutting coherent reflectors that are misaligned with the analysis window. For small analysis windows comprising 5 or 9 traces, the first principal component (where *L*=1) usually represents more than 95% of the data variability and provides a good filter. In contrast, for larger windows comprising 25 or more traces, such a simple filter may be too aggressive, where *L*=2 or 3 better represents the coherent signal.

Marfurt (2006) provides details on principal component structure-oriented filtering, while Chopra and Marfurt (2007) show the differences between principal component, mean, median, alpha-trimmed mean, and lower-upper-middle (LUM) filters. These latter four filters are a function only of the sample vector (data slice) that contains the analysis point (e.g., for the mean filter, the average of the samples along the center data slice). In contrast, the PC filter first constructs a covariance matrix from the 2*K+1* sample vectors, and then computes the eigenvectors $\mathbf{v}^{(j)}$ prior to cross-correlating it with the center sample vector. These eigenvectors are thus based on the statistics of neighboring sample vectors, and not just the single sample vector through the analysis point. Interpreters and statisticians commonly refer to the above equation as a "principal-component filter". However, this filter has been used in seismic data processing for decades where it is more commonly called the Karhunen-Loève filter (e.g., Hemon and Macé, 1978; Kirlin and Done, 1999). Depending on the data quality, one may wish to sum more than one principal component to obtain a quality image that preserves signal but rejects noise (Davogustto et al., 2011).

The total energy attribute is simply the sum of the energy of the weighted/balanced analytic traces described in equation 1b used to compute the covariance matrix:

$$E_{tot}=\sum_{k=-K}^{+K}\sum_{m=1}^{M}\left\{\left[d(t_k,x_m,y_m\right]^2+\left[d^H(t_k,x_m,y_m\right]^2\right\}. \tag{7}$$

In contrast, the coherent energy attribute is the sum of the energy of the corresponding weighted/balanced PC-filtered traces:

$$E_{coh}=\sum_{j=1}^{L}\sum_{k=-K}^{+K}\sum_{m=1}^{M}\left\{\left[d_{\mathrm{PC}}^{(j)}(t_k,x_m,y_m\right]^2+\left[d_{\mathrm{PC}}^{(j)\mathrm{H}}(t_k,x_m,y_m\right]^2\right\}. \tag{8}$$

However, when taking the ratio of the coherent energy to the total energy volumes the time slice through the energy ratio similarity volume shows significant detail:

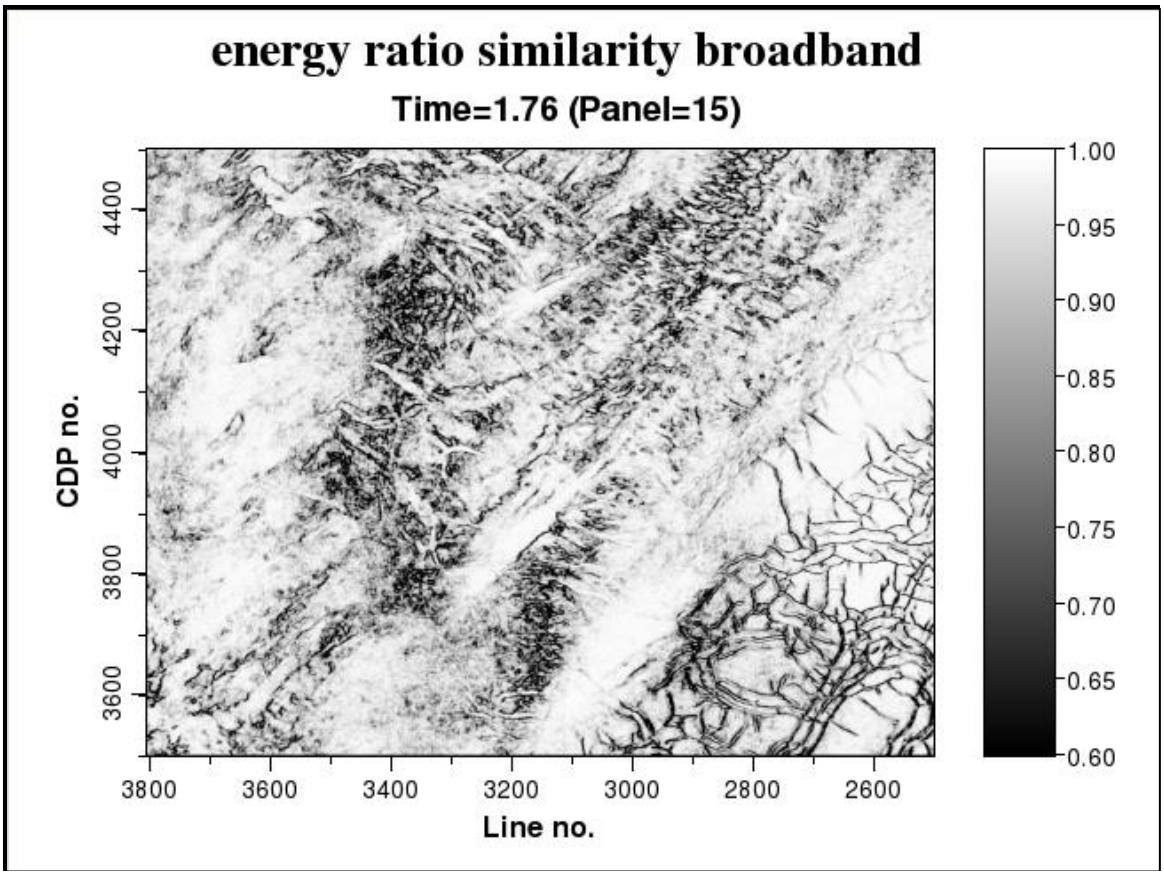


Figure 4.

Theory: Eigenstructure Coherence and Energy ratio Coherence

The eigenstructure coherence described by Gersztenkorn and Marfurt (1999) was computed from the eigenvalues of the covariance matrix:

$$c_{eigen} = \frac{\lambda_1}{\sum_{j=1}^{J} \lambda_j} . \quad (9)$$

The closely related but slightly more general energy ratio coherence described by Chopra and Marfurt (2007) is the ratio between the coherent to the total energy in the analysis window, each of which were described in a previous theory box:

$$c_{ratio} = \frac{E_{coh}}{E_{tot}} . \quad (10)$$

In principle, one could limit the sum of the two energy values to be just that of the center trace, which might provide improved lateral resolution. In practice, such estimates are noisier than the full window estimates and have been disabled in the software.

Co-rendering the coherent energy plotted against a fire color bar with the energy ratio coherence plotted against a monochrome black color bar using the AASPI program **corender** one gives:

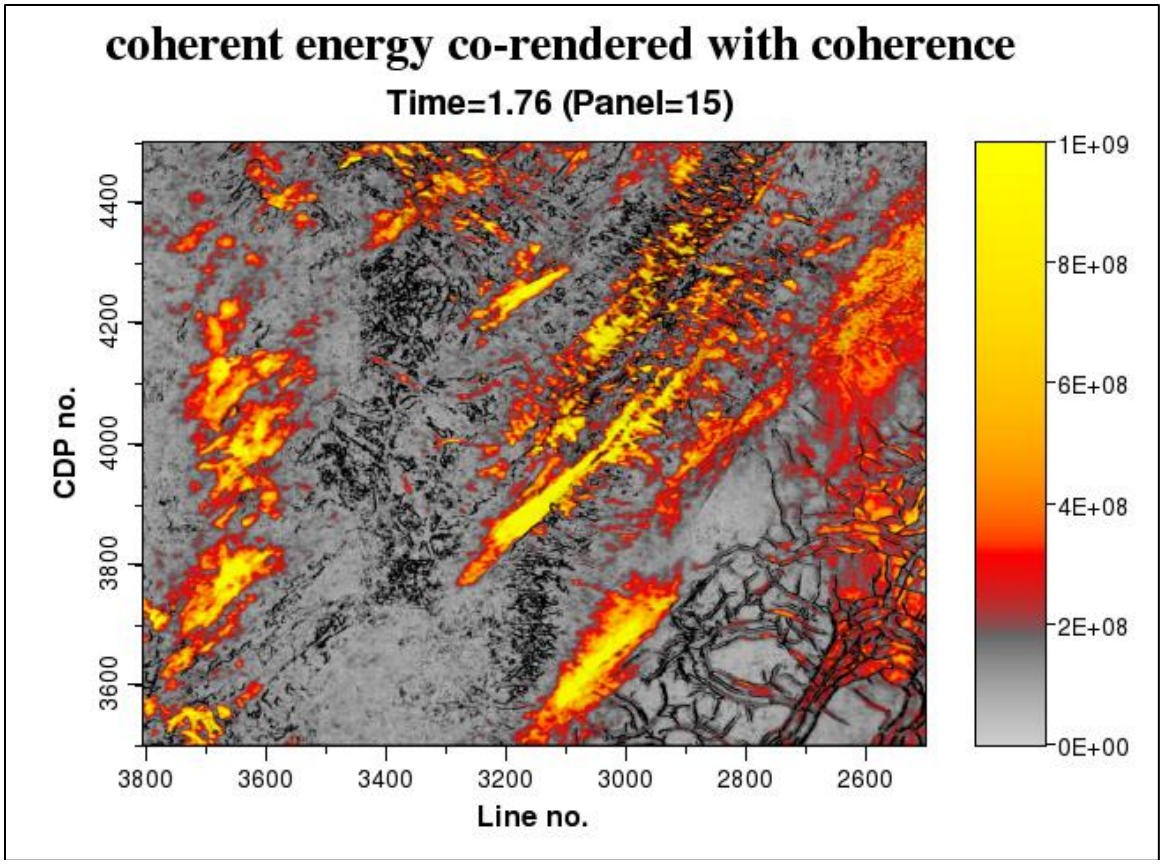
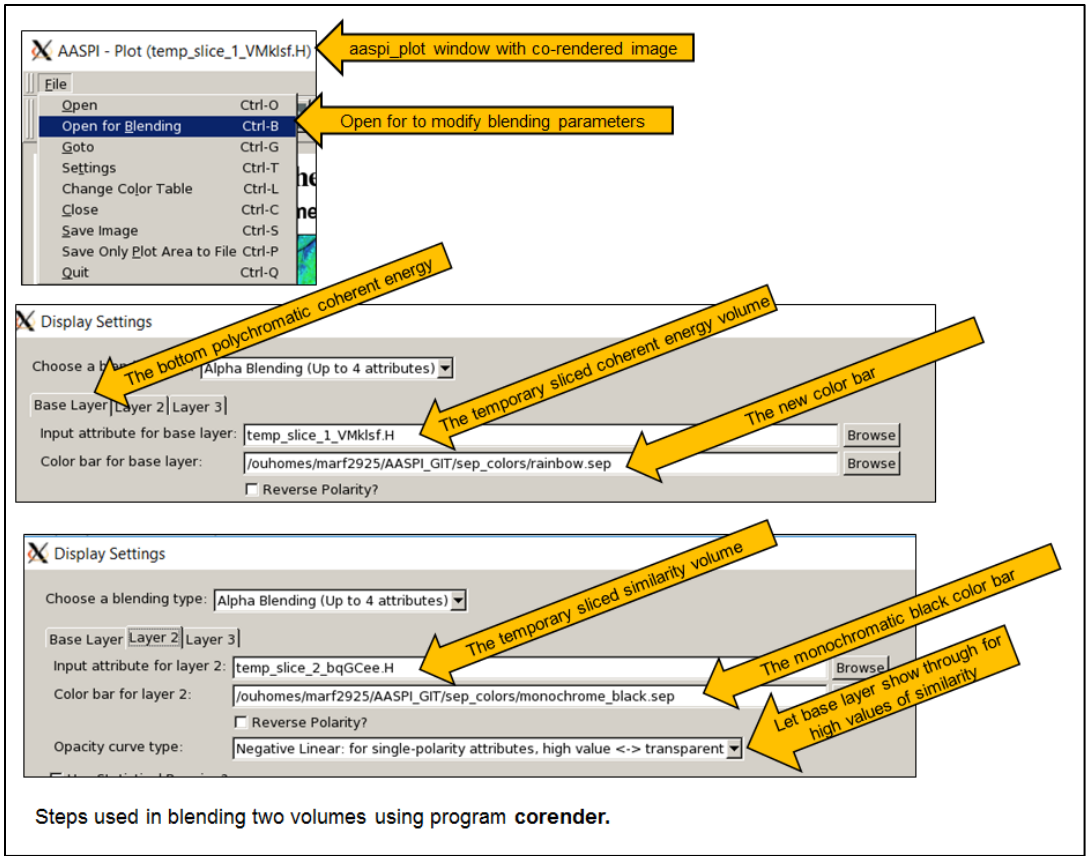


Figure 5.

The overlying gray scale on top of the gray part of the fire color bar is a poor choice since it leads to "gray on gray", thereby losing the edges. I therefore modify the settings to use a rainbow color bar as shown in the following screen capture images (see next page):



Steps used in blending two volumes using program **corender.**

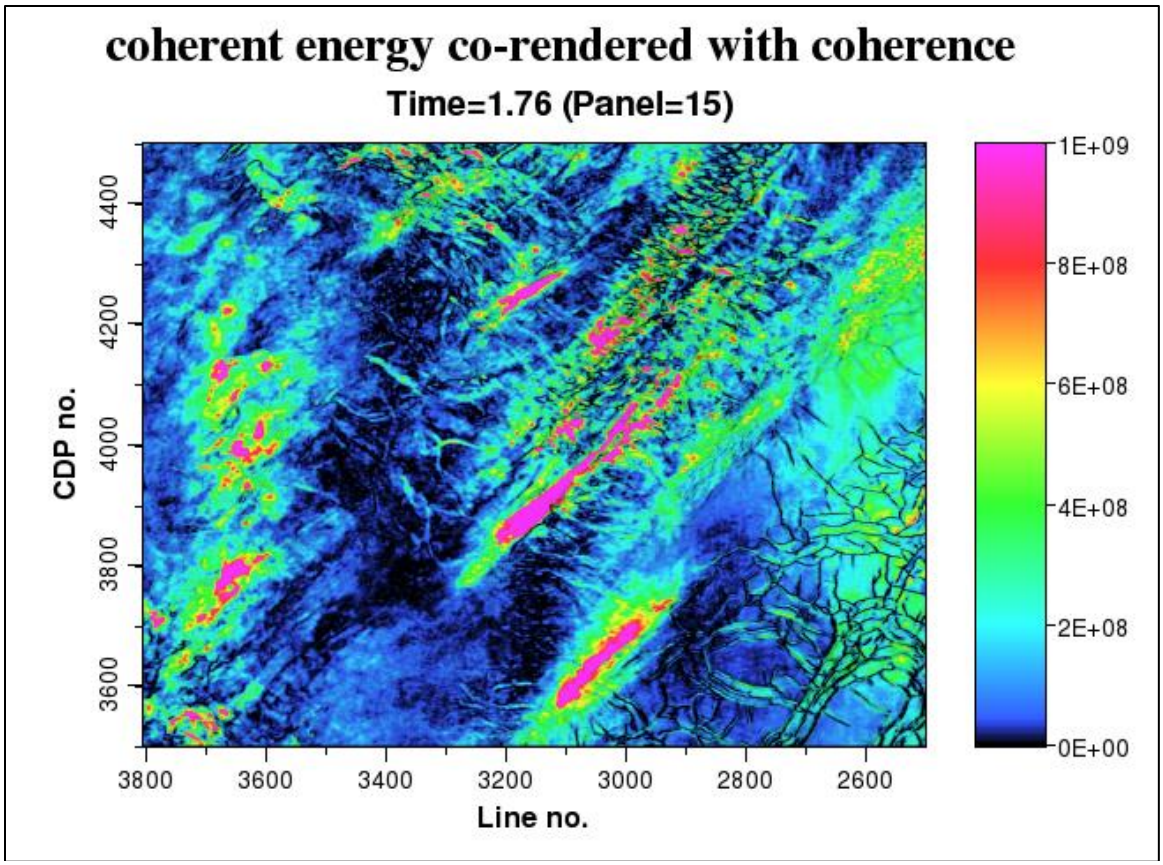


Figure 6.

Let us now turn to the outer product similarity, which, as discussed in the theory box, contains both eigenstructure coherence and semblance coherence as special cases. Selecting the constant test vector (defaulted) the structure-oriented, semblance similarity computed from the same analytic traces (indeed the same covariance matrix, **C**) as the energy ratio similarity image, one obtains the following image:

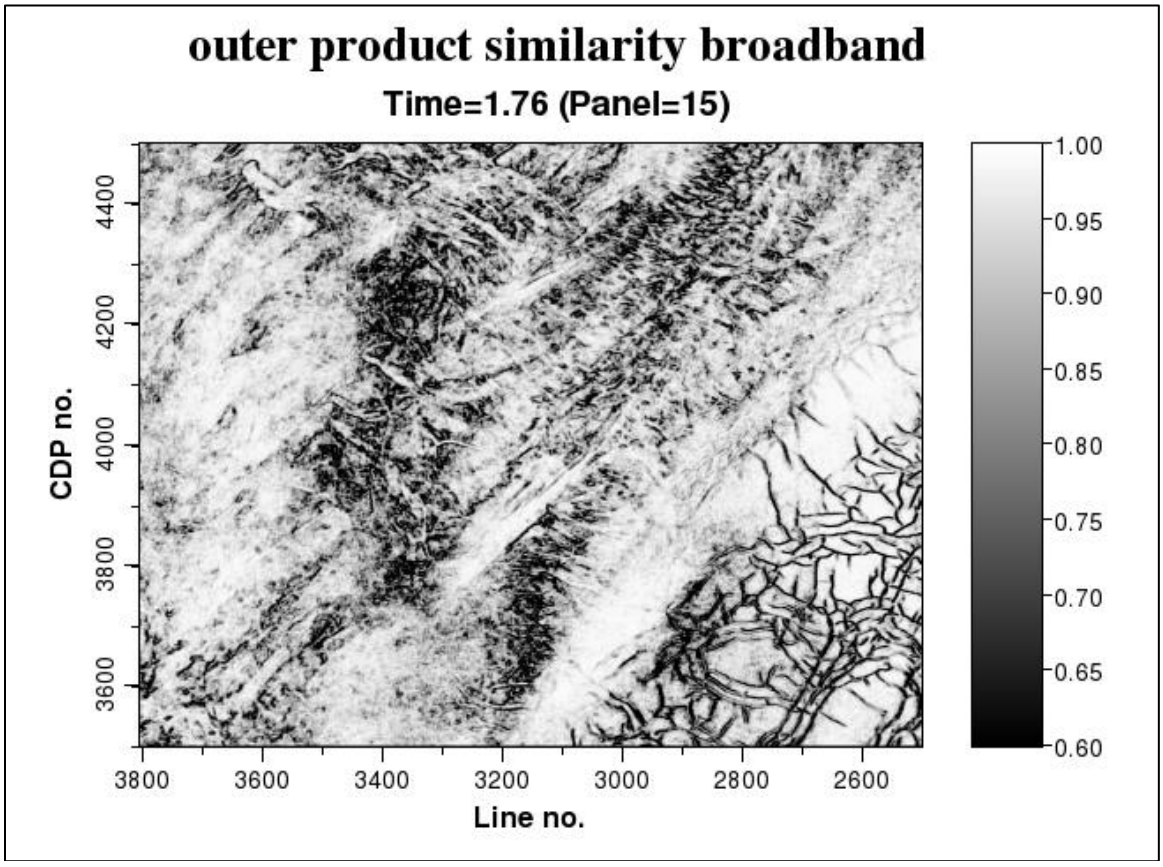


Figure 7.

The differences between energy ratio similarity and semblance are quite subtle. As discussed by Chopra and Marfurt (2007), one can also think of semblance-based coherence to be the ratio between the coherent energy and total energy in an analysis window, except that the "coherent component" of the data is defined by the average trace rather than by the first principal component. Further analysis will show that for a consistent waveform exhibiting laterally variable amplitudes, all the lateral variability can be represented by the first eigenvector, such that the energy ratio coherence will have a value of 1.0. In contrast, since the amplitudes differ between each trace, the semblance will no longer be zero. When animating between these two images, note first that the fault and channel edges are slightly sharper in the energy ratio similarity. However, also note that some of the channels appear more coherent in the energy ratio similarity image but show more internal detail in the outer product (semblance) similarity. These lateral changes in amplitude for a fixed waveform will be

expressed by the eigenvector and will be captured by the energy-weighted coherent amplitude gradient discussed later.

Theory: Outer-product and generalized semblance-based similarity

Semblance is well known to geophysicists who have done seismic velocity analysis. After its introduction by Bahorich and Farmer (1995), Marfurt et al. (1999) improved upon the original 3-trace cross-correlation algorithm by using a structure-oriented M-trace semblance coherence estimate:

$$c_{semb} = \frac{\sum_{k=-K}^{+K} \left\{ \frac{1}{M} \sum_{m=1}^{M} \left[d(t_k, x_m, y_m) \right] \right\}^2}{\sum_{k=-K}^{+K} \frac{1}{M} \left\{ \sum_{m=1}^{M} \left[d(t_k, x_m, y_m) \right]^2 \right\}}, \tag{11}$$

that can be interpreted to be the ratio of the energy of the average trace (in the numerator) to the average of the energies of each of the traces (in the denominator).

Using the more compact matrix notation, the outer-product similarity is defined to be:

$$c_{outer} = \frac{\mathbf{r}^{\mathrm{T}} \mathbf{C} \mathbf{r}}{E_{tot}}, \tag{12}$$

Where E_{tot} is the total energy defined earlier, and **r** is a test vector (e.g., Kirlin and Done, 1999). A reasonable test vector is to use the sample vector (structure-oriented amplitude slice) used to construct the covariance matrix **C** that contains the analysis point. If one sets $\mathbf{r}=\mathbf{v}^{(1)}$, that is, to be the first eigenvector, one obtains eigenstructure coherence. Finally, if one sets

$$\mathbf{r} = \frac{1}{M^{1/2}} \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix}, \tag{13}$$

One obtains a semblance estimate of coherence. The primary advantage of using this more general definition is that it allows one to compute semblance from laterally tapered windows of the analytic traces, facilitating a data-adaptive window computation of semblance (Lin, 2016).

Next, let us look at the differences between the outer product similarity computed using a constant test vector **r** described in the theory box above, to one using the sample vector containing the analysis point (i.e., uncheck the box after *Use constant test vector in outer product similarity*). The result looks like this:

For computational reasons, I think it is more efficient to compute a shorter wavelength (higher spatial resolution) inline and crossline coherent energy gradient, and then filter the results back using long wavelength amplitude curvature.

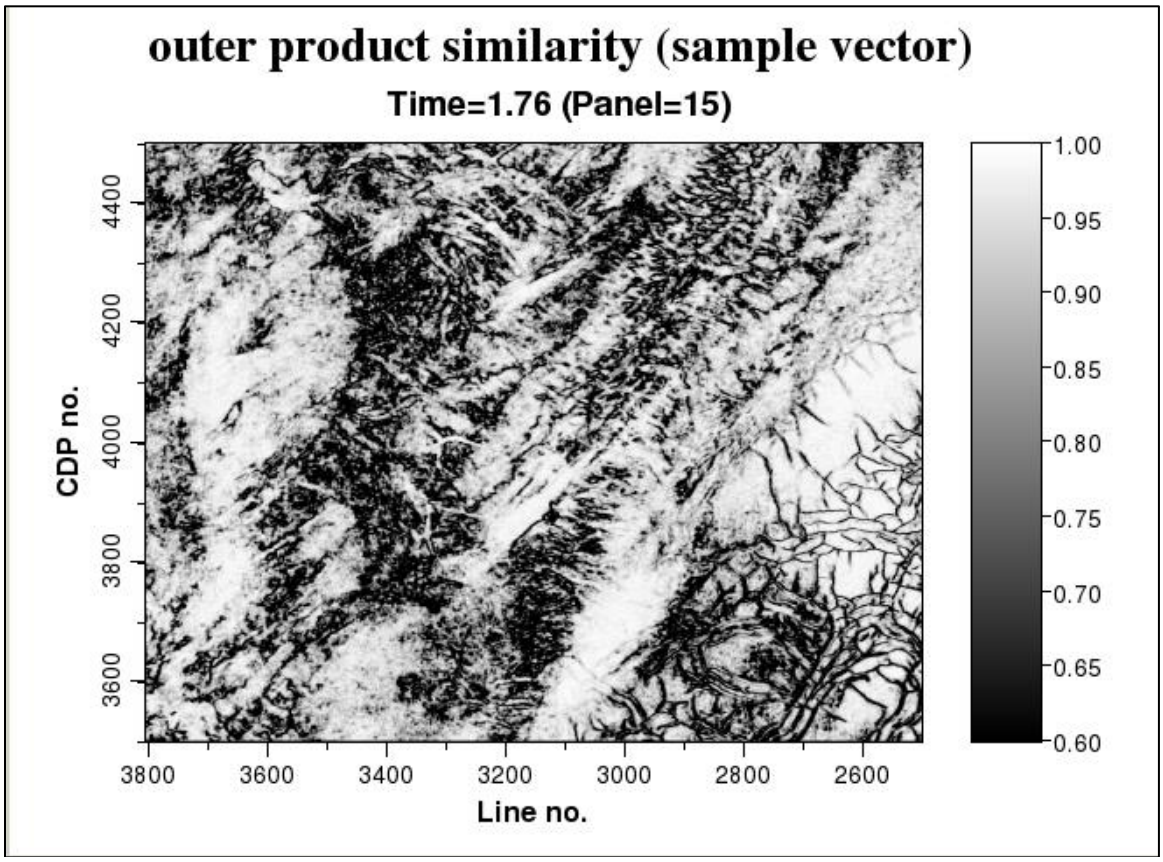


Figure 8.

Sobel-filter similarity is like that found along with other sharpening algorithms Photoshop and other image-processing software and is related to the algorithm described by Luo et al. (2003) using the generalized Hilbert transforms. Simply put, it is the magnitude of the inline and crossline derivatives of seismic amplitude along structural dip. The time slice at t=1.760 s looks like those generated by the energy ratio and semblance algorithms:

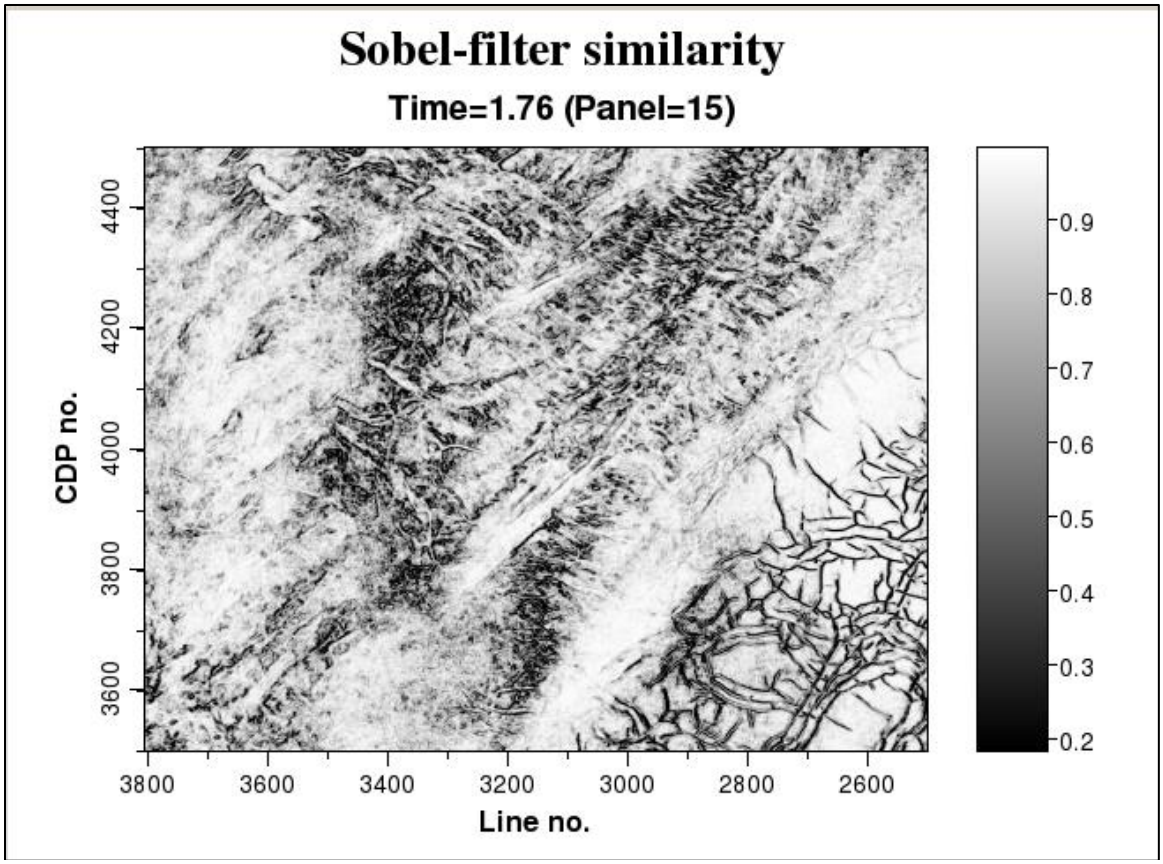


Figure 9.

Luo et al. (2003) use large operators, which will work well for relatively flat geology. However, for more complex structure, such operators should either follow dip or be applied to flattened volumes. An alternative workflow to obtain a long-wavelength amplitude-based edge detection is to first compute the inline and crossline components of the coherent energy gradient, and then use them as input to compute amplitude curvature in program **curvature3d**.

Theory: Sobel filter similarity

The Sobel filter is one of the more popular filters used in image processing and is embedded in most digital camera photo enhancement software. Luo et al. (1995) generalized the Sobel filter for application to seismic data by normalizing the inline and crossline derivatives by the energy. The details of their normalization are not defined. Generalizing the derivatives in x and y (always computed along structure) to be convolution operators $\mathbf{D^{(x)}}$ and $\mathbf{D^{(y)}}$:

$$\frac{\partial d}{\partial x} = \sum_{j=1}^{J} D_j^{(x)} d_j \text{ , and} \tag{14a}$$

$$\frac{\partial d}{\partial y} = \sum_{j=1}^{J} D_j^{(y)} d_j \text{ .} \tag{14b}$$

The Sobel filter similarity can be defined as,

$$c_{Sobel} = \frac{\sum_{k=-K}^{+K} \left\{ \left[\sum_{j=1}^{J} D_j^{(x)} d_j \right]^2 + \left[\sum_{j=1}^{J} D_j^{(y)} d_j \right]^2 + \left[\sum_{j=1}^{J} D_j^{(x)} d_j^H \right]^2 + \left[\sum_{j=1}^{J} D_j^{(y)} d_j^H \right]^2 \right\}^{1/2}}{\sum_{k=-K}^{+K} \left\{ \left[\sum_{j=1}^{J} \left| D_j^{(x)} \right| \left| d_j \right| \right]^2 + \left[\sum_{j=1}^{J} \left| D_j^{(y)} \right| \left| d_j \right| \right]^2 + \left[\sum_{j=1}^{J} \left| D_j^{(x)} \right| \left| d_j^H \right| \right]^2 + \left[\sum_{j=1}^{J} \left| D_j^{(y)} \right| \left| d_j^H \right| \right]^2 \right\}^{1/2}}, \tag{15}$$

where the values in the denominator are absolute values, forming a normalization. In the figures in this book, the spatial derivatives are computed using the spatial derivative operator defined by Marfurt (2006). As the analysis window becomes larger, the derivatives in the Sobel filter approximate the inline and crossline Hilbert transform, providing long wavelength results like that described by Luo et al. (2003).

The coherent energy gradient is a vector and has two components – an inline and a crossline component. The simplest way to understand it to is to envision a time structure map about which one has computed the RMS amplitude in a ±20 ms window. Now, using your interpretation workstation attribute calculator, compute the derivative first in the inline and then in the crossline directions, giving RMS amplitude gradients. Alternatively, compute the envelope volumetrically. Then compute an inline and crossline derivative as described by Barnes (2011). A naïve application would give rise to structural leakage artifacts, described by, and addressed by Barnes (2011), although the algorithmic details of the artifact suppression are not disclosed.

similarity3d also computes amplitude derivatives, but along structural dip. Specifically, since the data in an analysis window can be estimated by its first "eigenmap" (the first eigenvector), we can compute inline and crossline derivatives of the coherent part of the energy (see theory box) by computing the derivative of the eigenmap. **similarity3d** then allows us to weight these derivatives by either the energy or the RMS amplitude (square root of the energy given by the first eigenvalue, λ_1) for interpretation and as input to program **curvature3d**. The resulting inline

and crossline coherent energy gradients looks much like shaded relief maps, except they are shades of the lateral variation of energy, not of time structure.

Theory: The Coherent Energy (or RMS Amplitude-) Weighted Amplitude Gradient

Like most volumetric attributes, the computation is performed in a running analysis window centered and aligned with the local estimate of structural dip and azimuth. A typical analysis window may consist of 3 by 3 or *J*=9 traces and ±*K*=5 time or depth samples which are used to construct a *J-by-J* covariance matrix, **C**, as defined in equation 1 using both the original data, **d**, and its Hilbert transform, $\mathbf{d}^{\mathbf{H}}$. Next, one computes the *J* eigenvectors, $\mathbf{v}^{(j)}$ of the covariance matrix, **C** (also described in Appendix 1).
The first unit length eigenvector, $\mathbf{v}^{(1)}$, best represents the seismic amplitude variation laterally across the *J*-trace analysis window. The corresponding first eigenvalue, λ_1, represents the energy represented by this eigenvector. Since the *J*-trace analysis window extends laterally in the inline and crossline directions, the eigenvector $\mathbf{v}^{(1)}$ defines a small (e.g. 3×3) map. The derivatives of this map are simply

$$\frac{\partial v^{(1)}}{\partial x} = \sum_{j=1}^{J} D_{1j}^{(x)} v_j^{(1)} \text{ , and} \tag{16a}$$

$$\frac{\partial v^{(1)}}{\partial y} = \sum_{j=1}^{J} D_{1j}^{(y)} v_j^{(1)} . \tag{16b}$$

Where $\mathbf{D}_1^{(x)}$ and $\mathbf{D}_1^{(y)}$ are convolution operators that approximate the first derivative on the seismic grid.

Since λ_1 represents the energy of the window, the energy-weighted amplitude gradient, **g**, is simply

$$\mathbf{g} = \begin{bmatrix} g_x \\ g_y \end{bmatrix} \equiv \lambda_1 \begin{bmatrix} \dfrac{\partial v^{(1)}(x,y)}{\partial x} \\ \dfrac{\partial v^{(1)}(x,y)}{\partial y} \end{bmatrix}. \tag{17}$$

Our initial implementation used the weight λ_1, which by construction emphasized the higher energy reflectors in the seismic data volume. To obtain an image more representative of the range of seismic amplitudes, one can weigh the gradients as,

$$\mathbf{g} = \begin{bmatrix} g_x \\ g_y \end{bmatrix} \equiv \sqrt{\lambda_1} \begin{bmatrix} \dfrac{\partial v^{(1)}(x,y)}{\partial x} \\ \dfrac{\partial v^{(1)}(x,y)}{\partial y} \end{bmatrix}, \tag{18}$$

where $\sqrt{\lambda_1}$ can be interpreted as the RMS amplitude of the coherent component of the seismic data within the analysis window. Very effective displays can be made by co-rending the coherent energy and the coherent energy gradient, with the former being plotted against a heat or fire color bar and the latter being plotted against a grayscale and rendered 50% transparent.

Since it is a vector, the coherent energy gradient, **g**, can be rotated to be aligned with an azimuth φ using simple trigonometry

$$\mathbf{g}(\varphi) = \begin{bmatrix} \cos\varphi & \sin\varphi \\ -\sin\varphi & \cos\varphi \end{bmatrix} \begin{bmatrix} g_x \\ g_y \end{bmatrix} \tag{19}$$

using AASPI program **apparent_cmpt** or an attribute calculator in your interpretation workstation.

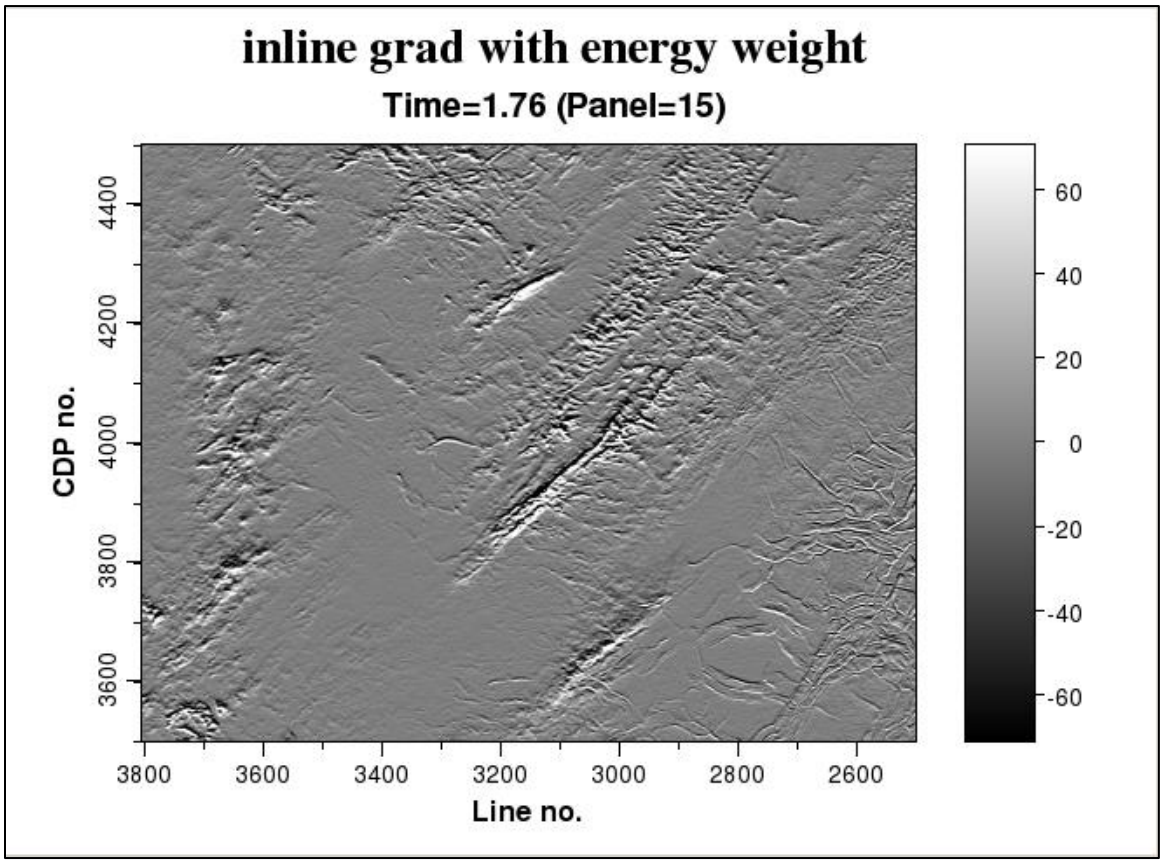

inline grad with energy weight
Time=1.76 (Panel=15)
CDP no.
4400
4200
4000
3800
3600
Line no.
3800
3600
3400
3200
3000
2800
2600
60
40
20
0
-20
-40
-60

Figure 10.

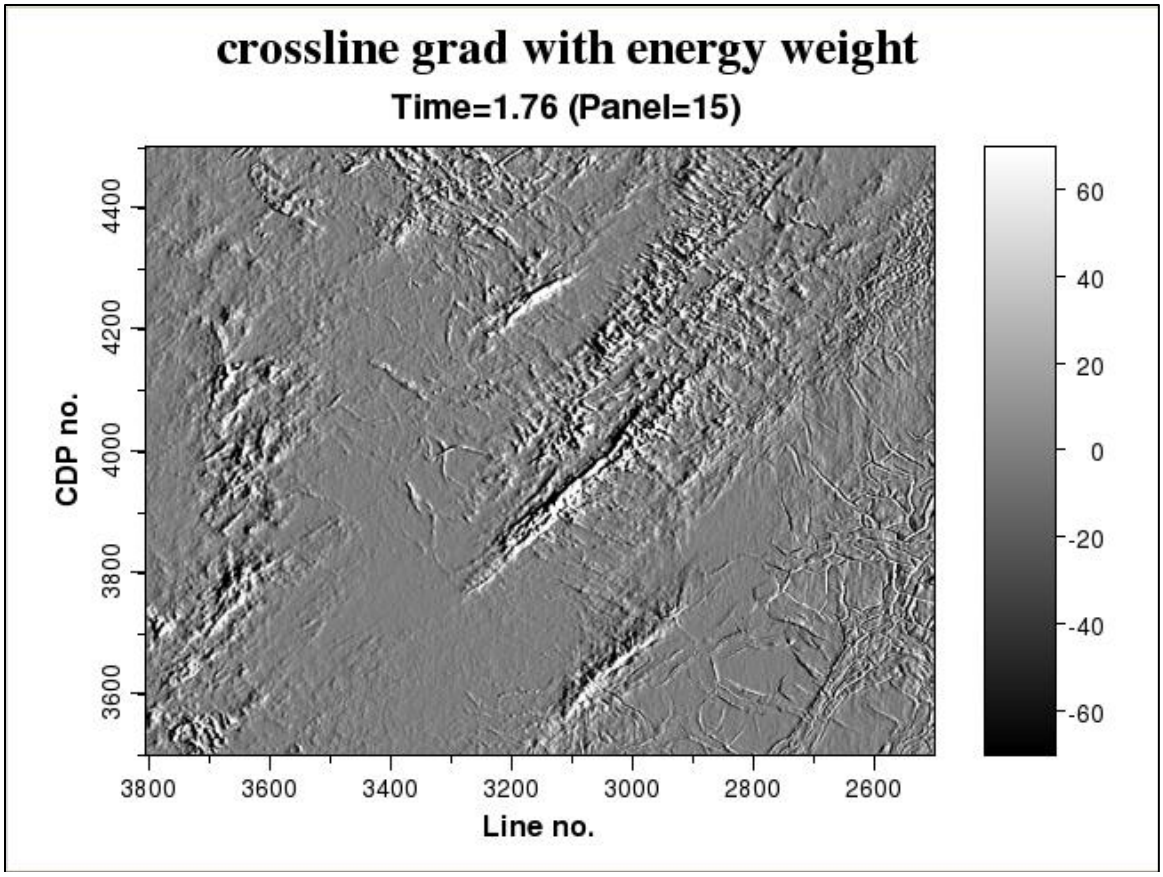

crossline grad with energy weight
Time=1.76 (Panel=15)
CDP no.
4400
4200
4000
3800
3600
Line no.
3800
3600
3400
3200
3000
2800
2600
60
40
20
0
-20
-40
-60

Figure 11.

Geometric Attributes: Program **similarity3d**

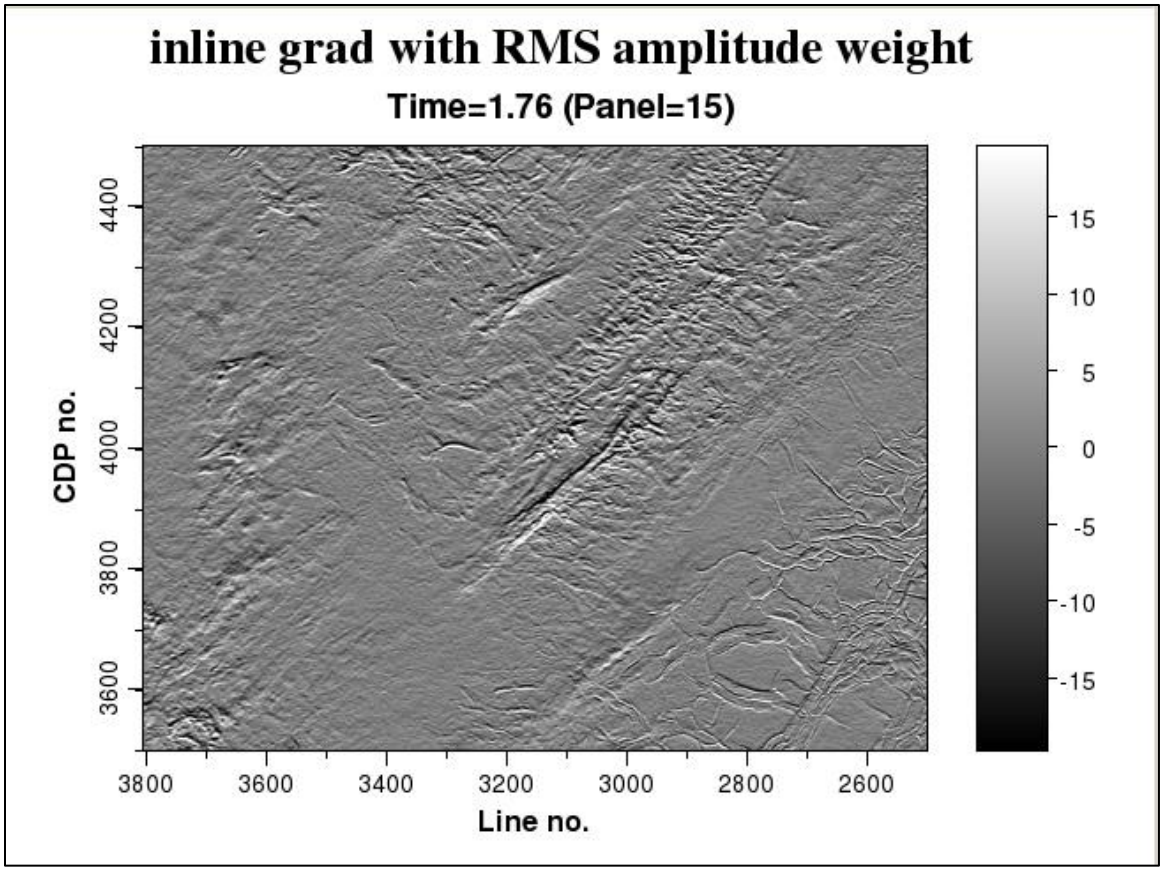


Figure 12.

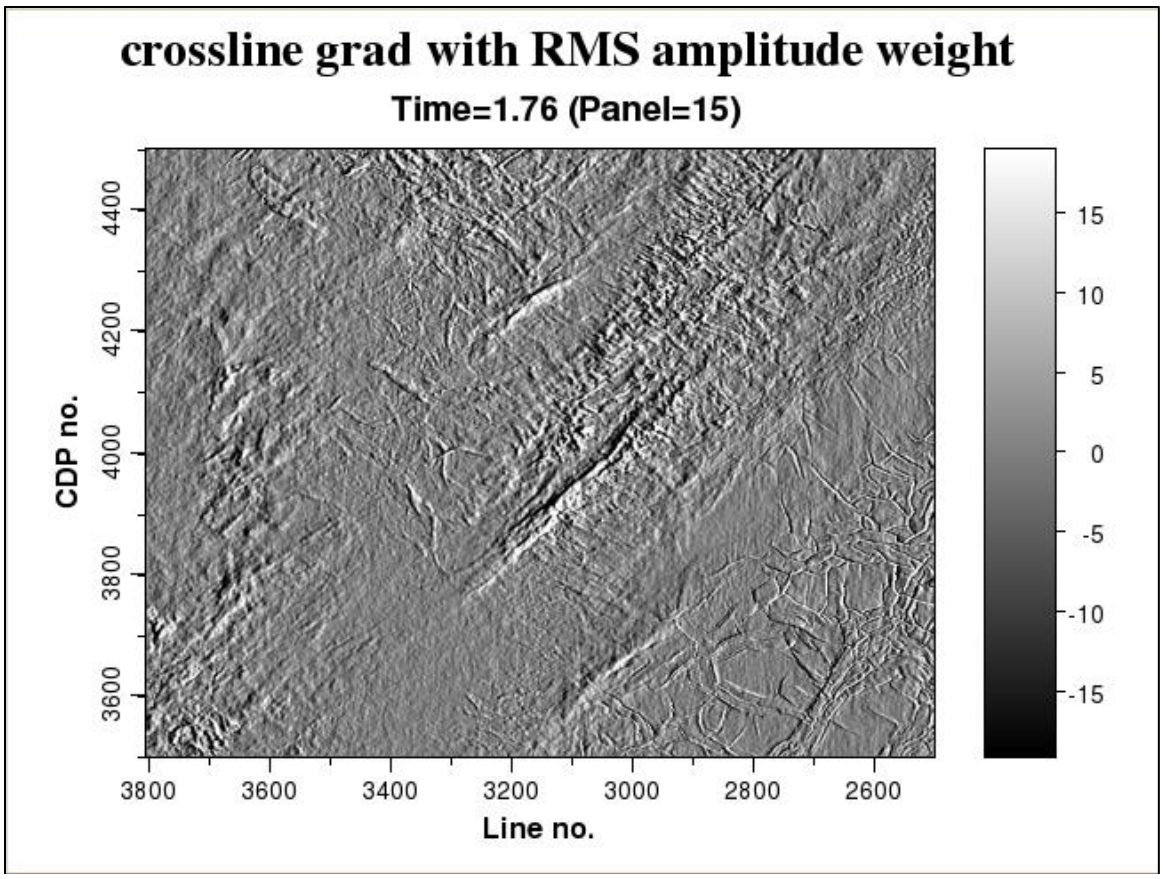


Figure 13.

Co-rendering these gradients using program **corender**, with coherent energy as the base attribute, and a binary black-white color bar for the gradient components, and transparency = 100 at gradient values of 0, gives the following two images:

Geometric Attributes: Program **similarity3d**

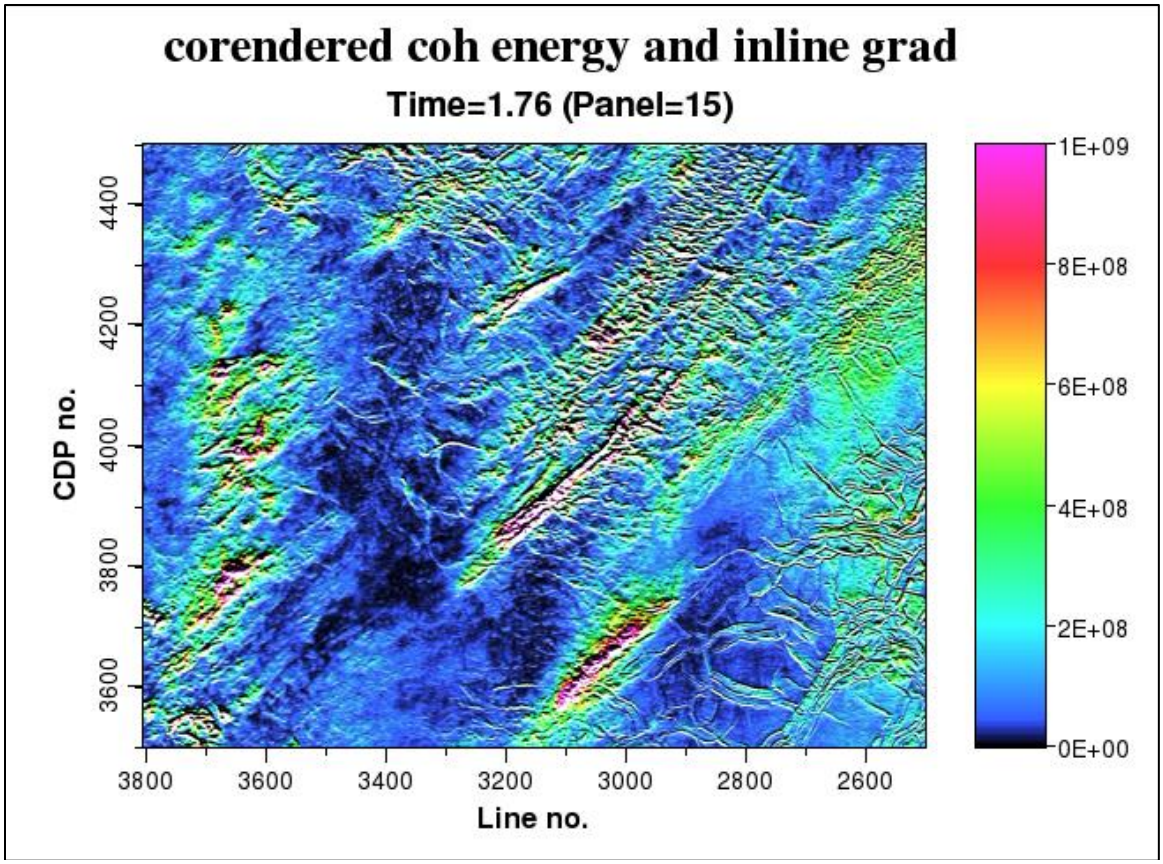


Figure 14.

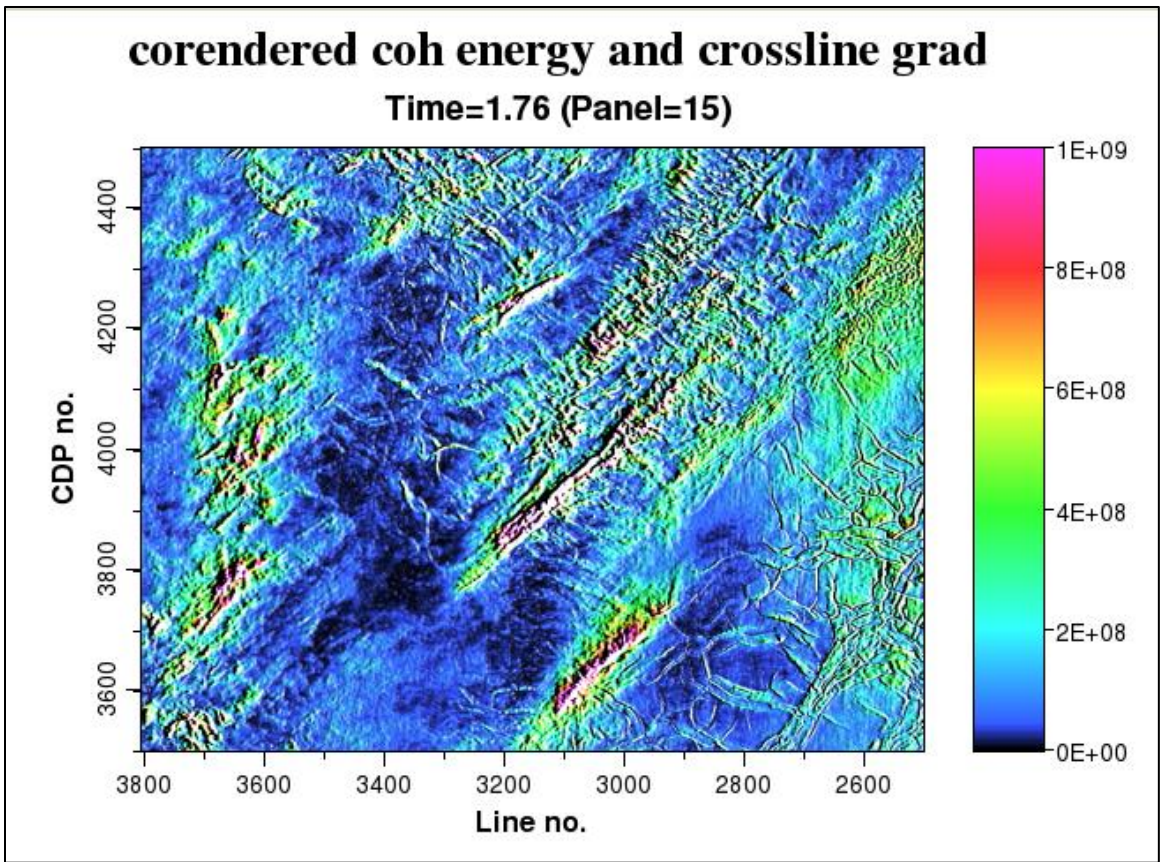


Figure 15.

Since it is a 2D vector, the coherent energy gradient can be rotated, much like a shaded relieve volume can. Ideally, one would do this using commercial software to allow a degree of interactivity. If your software does not have such functionality, use aaspi program **apparent_cmpt** to do so to obtain images like these:

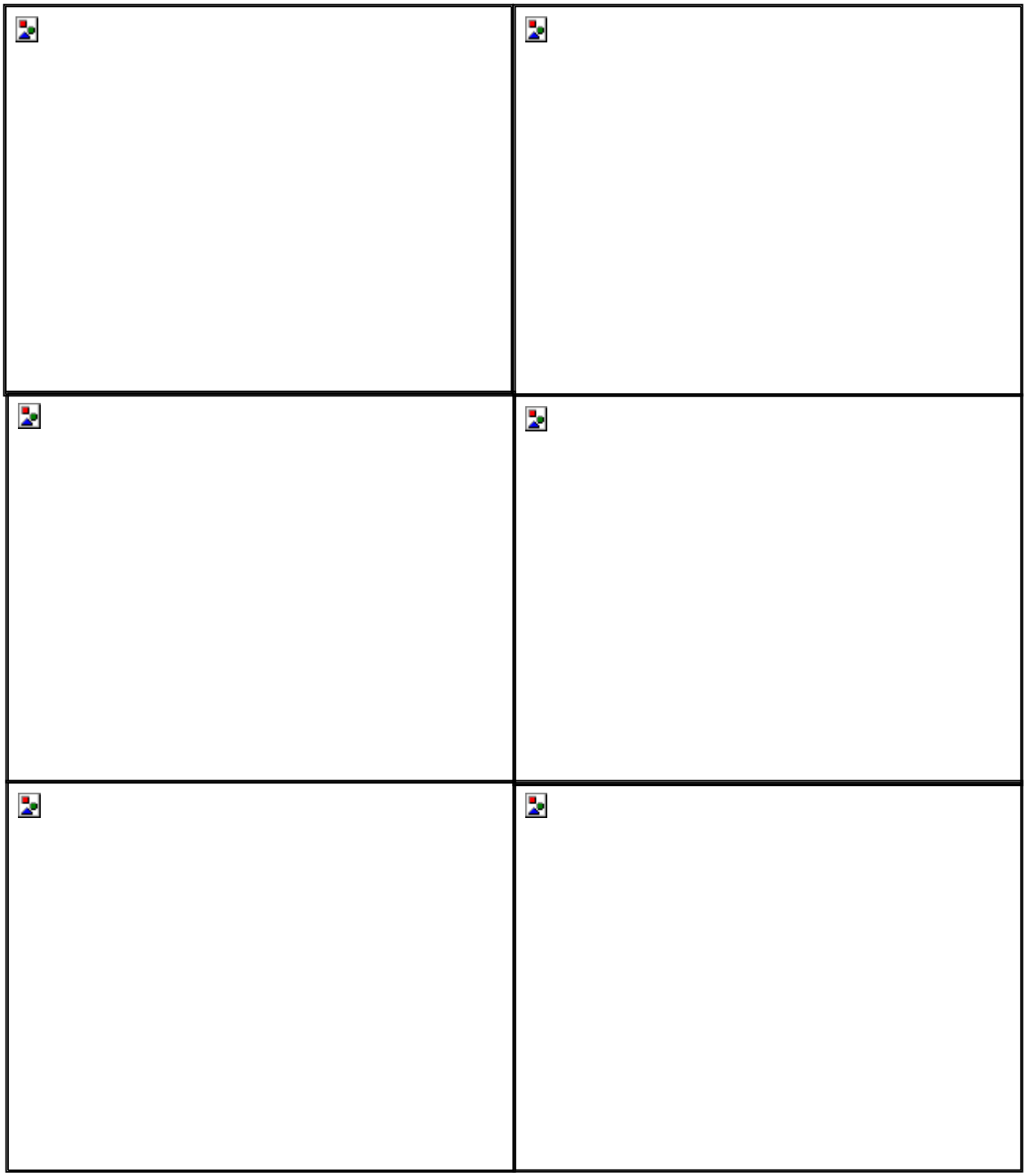


Figure 16.

Theory: Energy- and RMS amplitude weighted Laplacian filters

The Laplacian filter is another popular filter used in image processing and embedded in most digital camera photo enhancement software. Until December 2020, we computed the Laplacian filter by calculating the *mean* amplitude curvature using program **curvature3d** and the inline and crossline amplitude gradients as input. While this multistep process provides a great deal of flexibility, computing the mean amplitude curvature within similarity3d provides an opportunity to not only compute the Laplacian of multiple spectral components, but also, by summing the covariance matrices for each filter bank as described below, to compute a multispectral Laplacian filter. As with the amplitude gradient, the 2nd derivatives in in x and y (always computed along structure) are approximated by convolution operators $\mathbf{D_2^{(x)}}$ and $\mathbf{D_2^{(y)}}$ that use all the data in the analysis window:

$$\frac{\partial^2 v^{(1)}}{\partial x^2} = \sum_{j=1}^{J} D_{2j}^{(x)} v_j^{(1)} \text{ , and} \tag{20a}$$

$$\frac{\partial^2 v^{(1)}}{\partial y^2} = \sum_{j=1}^{J} D_{2j}^{(y)} v_j^{(1)} \text{ ,} \tag{20b}$$

where $\mathbf{v}^{(1)}$ is the first eigenvector, which best represents the lateral amplitude pattern in both the original data, d_j and its Hilbert transform d_j^H.

Because the eigenvector by definition is a unit vector, $|\mathbf{v}^{(1)}| \equiv 1$, we can normalize the Laplacian by the coherent energy, E_{coh}, of the analysis window, which is the first eigenvector, λ_1,

$$L_{\text{energy-weigthed}} = \lambda_1 \left(\frac{\partial^2 v^{(1)}}{\partial x^2} + \frac{\partial^2 v^{(1)}}{\partial y^2} \right), \tag{21}$$

Or by the coherent RMS amplitude of the analysis window

$$L_{\text{RMS-amplitude-weigthed}} = \sqrt{\lambda_1} \left(\frac{\partial^2 v^{(1)}}{\partial x^2} + \frac{\partial^2 v^{(1)}}{\partial y^2} \right). \tag{22}$$

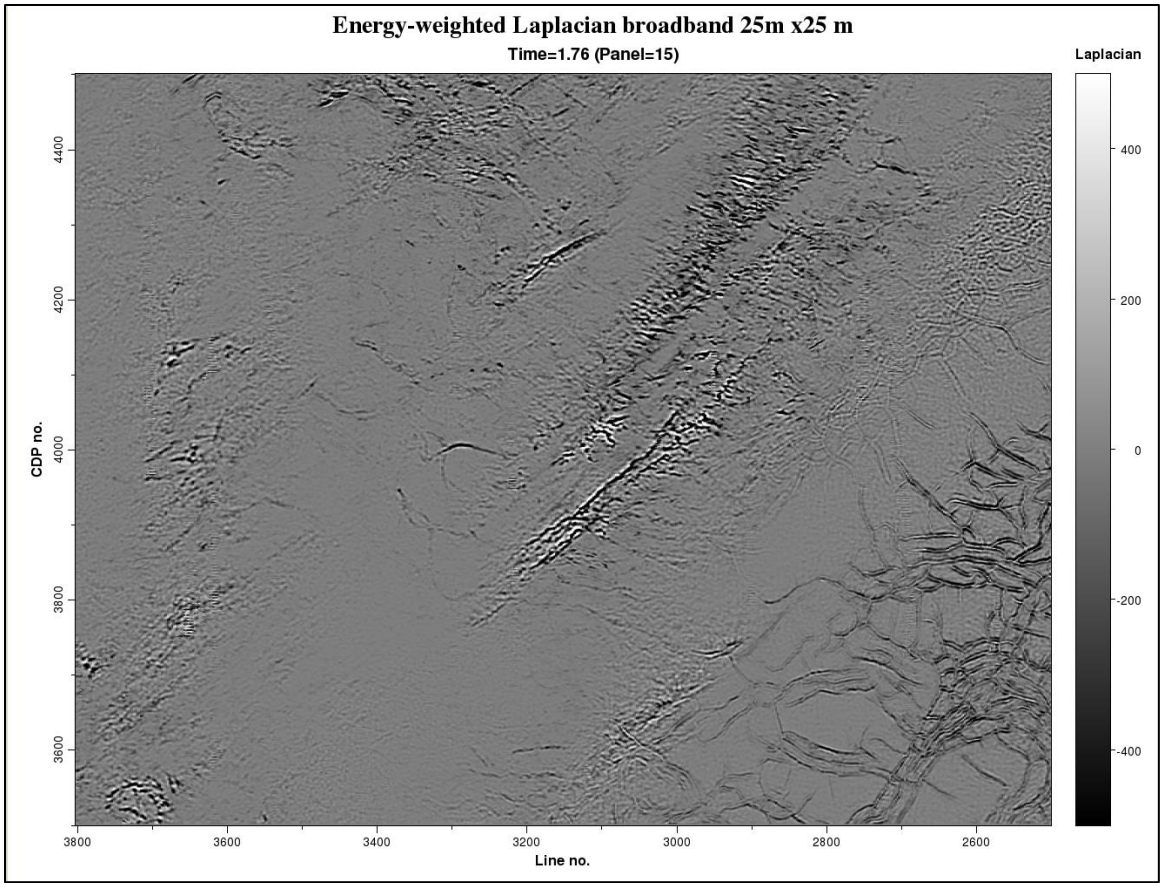


Figure 17. Time slice at *t*=1.76 s through the energy-weighted Laplacian showing faulting in the lower right and channels cutting through the slope in the middle. This image provides less insight into the geology than those using the energy-weighted gradient and energy ratio coherence.

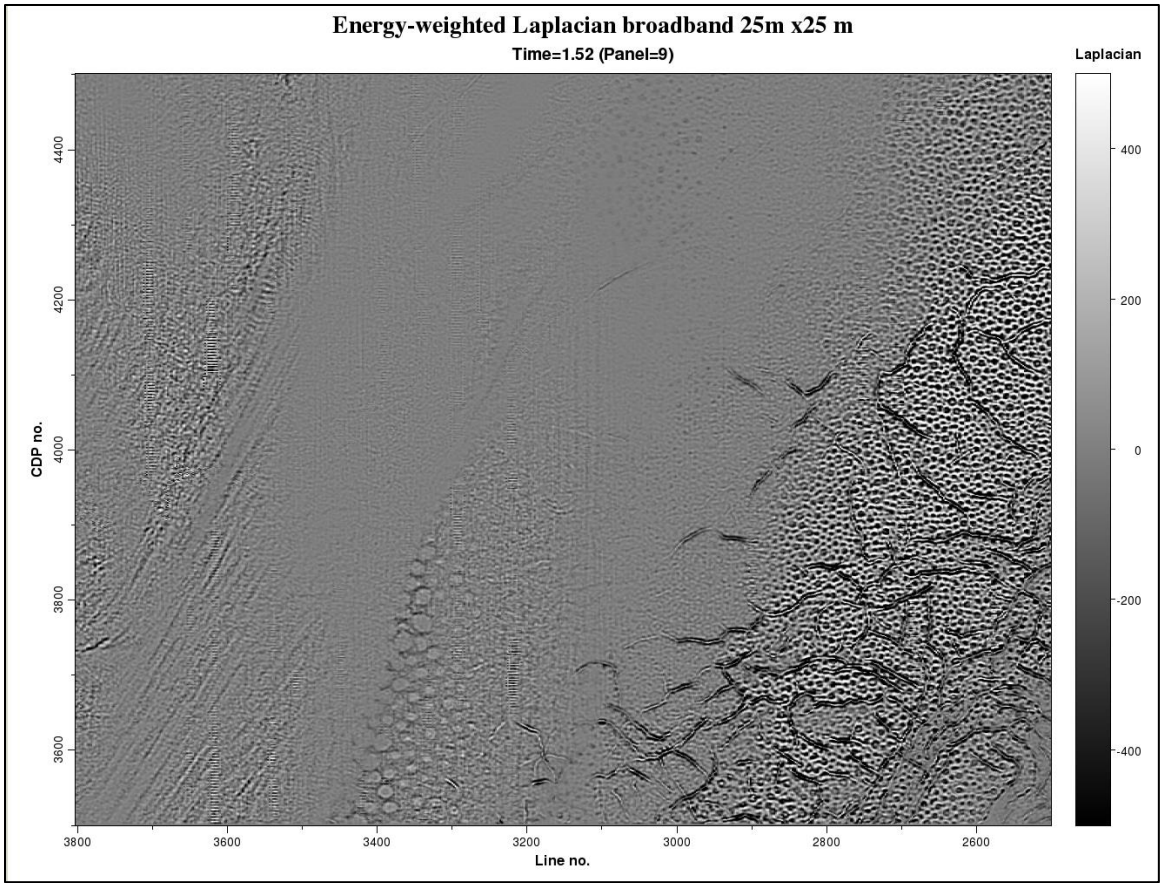


Figure 18. In contrast, the energy-weighted Laplacian shows very nice delineation of the hexagonal syneresis features at *t*=1.52 s where we see large features in the center and smaller features on the right, which are likely controlled by the lithology. The linear features on the left correlate to downlapping reflectors on the slope.

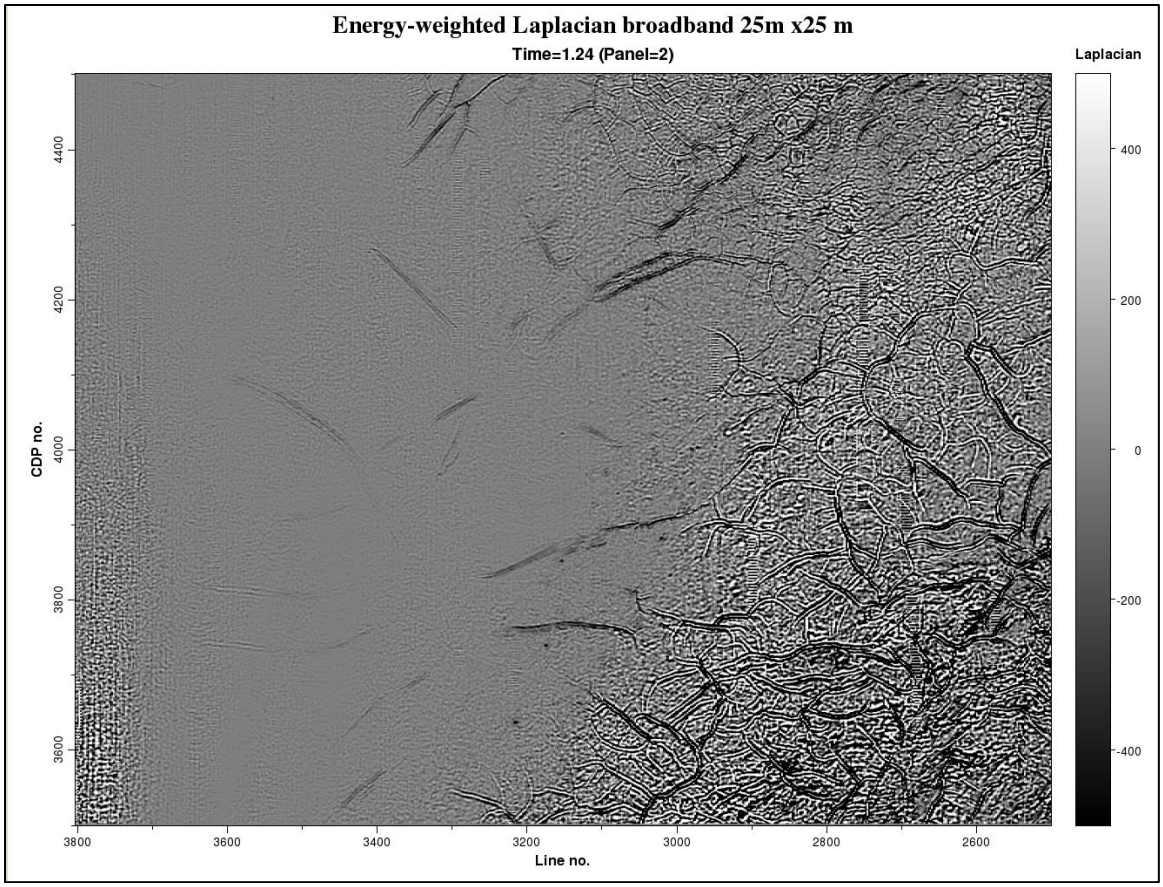


Figure 19. The time slice at *t*=1.24 s appears to show many of the smaller faults that were note well imaged by coherence. We have seen similar detail when computing amplitude curvature using AASPI program **curvature3d**. Recall that the Laplacian is basically the mean amplitude curvature.

Multispectral Coherence, Gradients, and Laplacians

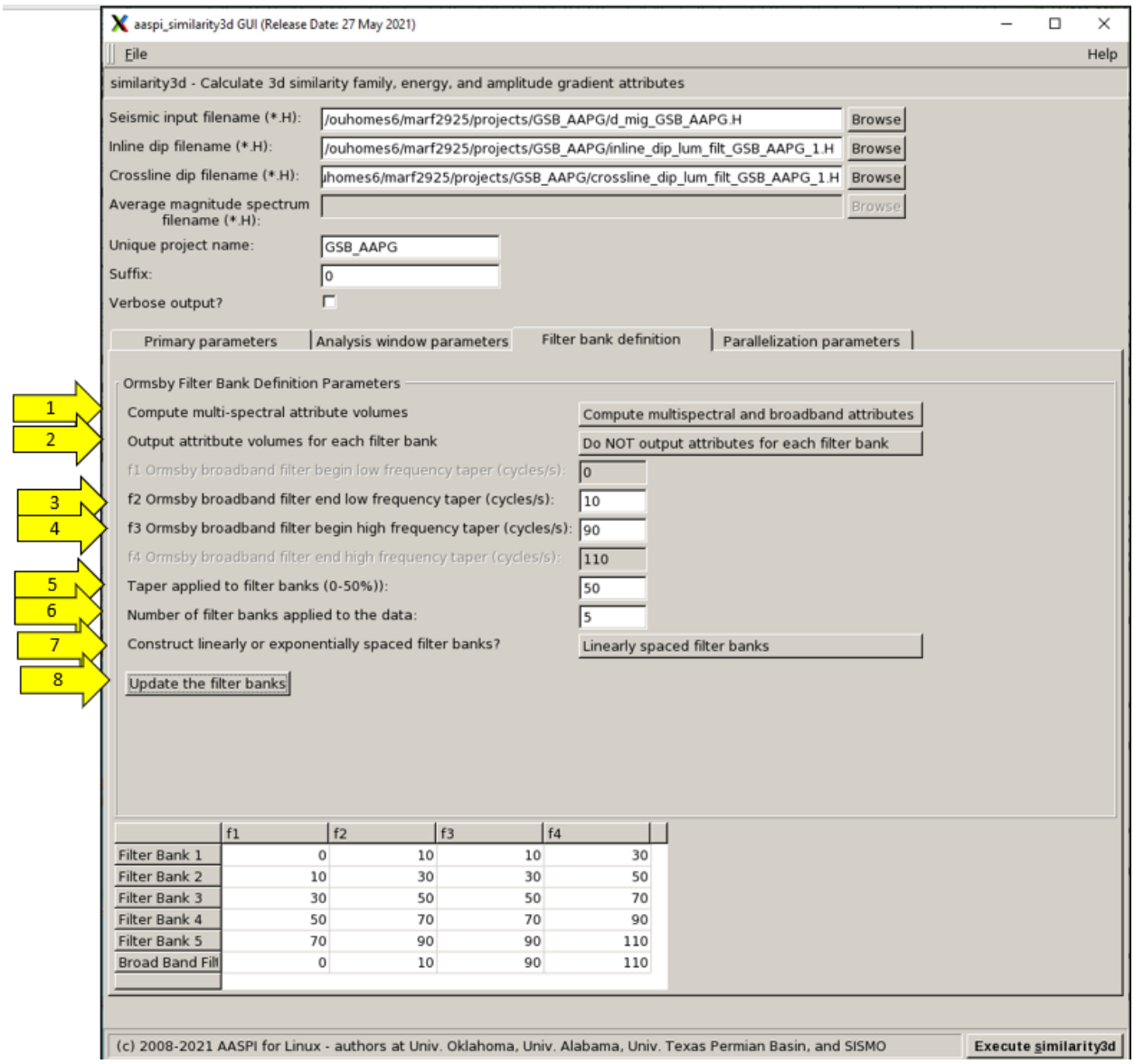
Filter banks and spectral decomposition

Hardage (2009) recognized that because of the variable signal-to-noise ratio at different frequencies, that faults were more easily identified in his data on the low frequency components that were less contaminated by strong interbed multiples. Gao (2013) showed how different components of narrow band spectral probes highlighted different edges at different frequencies. Li and Lu (2014) and Honorio et al. (2016) computed coherence from a suite of spectral components and combined them using RGB colour blending, resulting in not only improved discontinuity images, but in addition, an estimate at which spectral bands such discontinuities occurred. The main limitation of this approach is that only three spectral components can be co-rendered at any one time.

Geometric Attributes: Program **similarity3d**

Using the spectral voices computed from programs **spec_cwt**, and **spec_cmp**, and the cross-correlation components from **spectral_probe** as input to **similarity3d**, one can compute the coherence response for a suite of filter banks. (Note that AASPI program **spec_max_entropy** provides suboptimum input to **similarity3d** since it favors discrete rather than continuous spectral components, resulting in "holes" in the output spectral voices). However, this workflow requires running program **similarity3d** multiple times, resulting in significant intermediate output that may not be used. For this reason, in January 2017 we released a multispectral option in program **similarity3d**, which can be found on the *Filter bank definition* tab:

The Filter Bank Definition Tab (Defining Multispectral Filter Banks)



To invoke a multispectral computation, toggle (1) *Compute multi-spectral attribute volumes.* If you wish to examine the attributes computed from each of the filter banks, toggle (2) Output attribute volumes for each filter bank. The variables (3) *f2* and (4) *f3* define the range of the filter banks, while (6) the number of filter banks defines how many band-limited versions of the input data will be analyzed. The filter banks will have the form of an Ormsby filter defined by corner frequencies f_1, f_2, f_3, and f_4. The tapers between f_1 and f_2 and between f_3 and f_4 are defined as half of raised cosines that are (5) a percentage of the width of each filter bank. If this taper is 0%, no taper is applied; if the taper is 50%, $f_2=f_3$ for each filter bank and the filter will have the form a raised cosine (the default). In June 2019 we added more flexibility in the design of the filter banks, which can now be:

- Linearly spaced, constant size filter banks,
- Exponentially spaced, exponentially increasing size filter banks (where the spectrum is sampled by octaves), and
- Arbitrarily defined, where you the interpreter, can simply type in a suite of four-point Ormsby filters.

In the example above, we choose five linearly spaced filter banks ranging between 0 and 90 Hz each with a 50% taper. Note that now the previous default broadband filter is now redefined to encompass the 5 smaller filter banks. Using the *Wiggle* option in program **aaspi_plot** provides the following image of the file *filter_banks_similarity3d_GSB_AAPG_0.H*, where the last filter bank (number 6) is the broad band response:

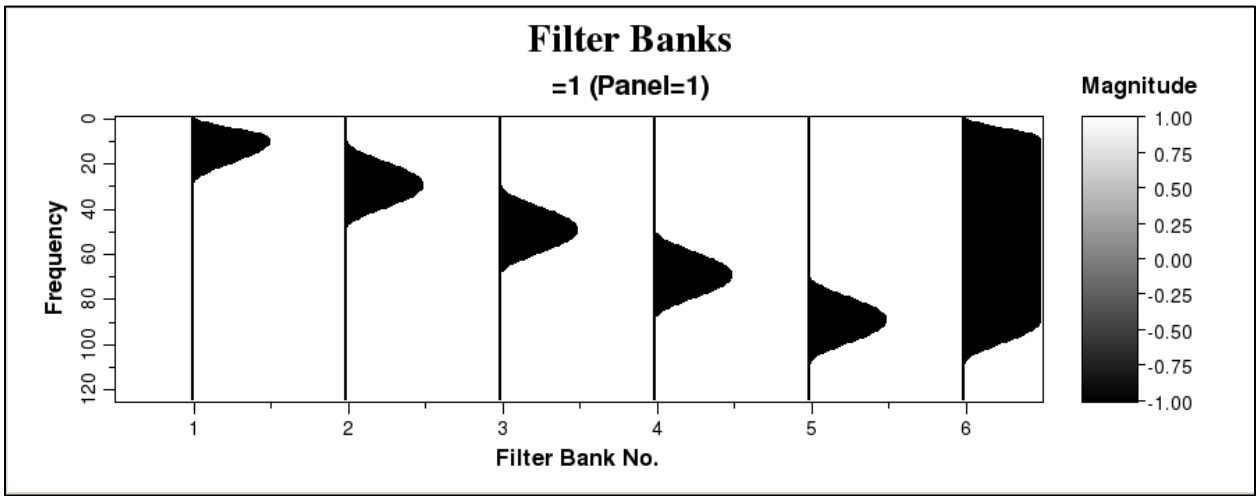


Figure 20.

Setting the *Taper* applied to the filter banks to be 25% provides the following numerical table of filter banks:

Primary parameters | Analysis window parameters | Filter bank definition | Parallelization parameters

Ormsby Filter Bank Definition Parameters

Compute multi-spectral attribute volumes	Compute multispectral and broadband attributes
Output attritbute volumes for each filter bank	Do NOT output attributes for each filter bank
f1 Ormsby broadband filter begin low frequency taper (cycles/s):	0
f2 Ormsby broadband filter end low frequency taper (cycles/s):	5
f3 Ormsby broadband filter begin high frequency taper (cycles/s):	95
f4 Ormsby broadband filter end high frequency taper (cycles/s):	105
Taper applied to filter banks (0-50%)):	25
Number of filter banks applied to the data:	5
Construct linearly or exponentially spaced filter banks?	Linearly spaced filter banks

Update the filter banks

	f1	f2	f3	f4
Filter Bank 1	0	5	15	25
Filter Bank 2	15	25	35	45
Filter Bank 3	35	45	55	65
Filter Bank 4	55	65	75	85
Filter Bank 5	75	85	95	105
Broad Band Filt	0	5	95	105

and a corresponding plot, where we note the band-pass filters flatten out between the tapers:

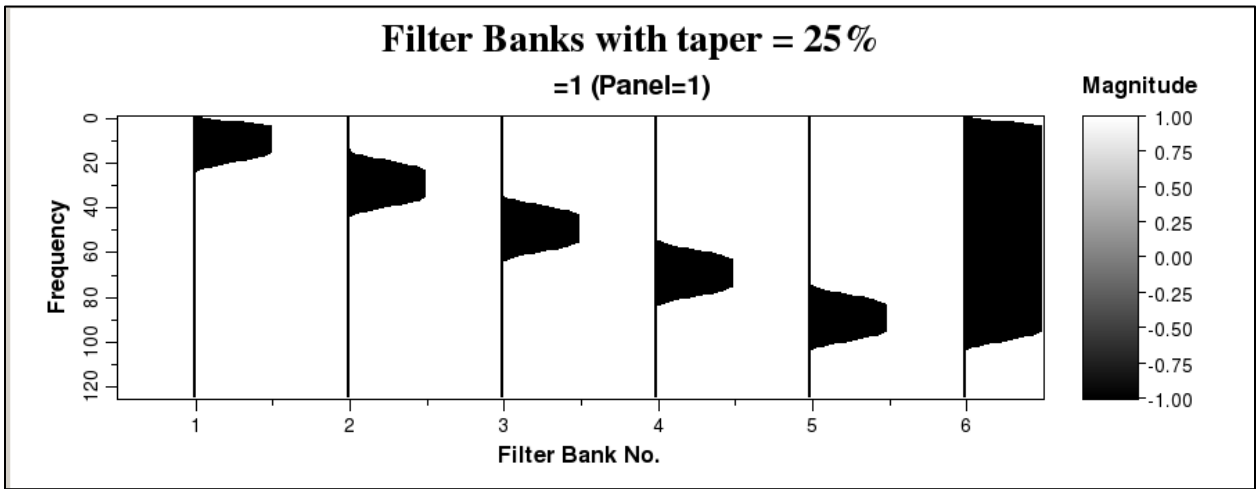


Figure 21.

If we choose to define the filter banks to be exponentially (rather than linearly) spaced, the table appears like this:

Primary parameters | Analysis window parameters | Filter bank definition | Parallelization parameters

Ormsby Filter Bank Definition Parameters

Compute multi-spectral attribute volumes	Compute multispectral and broadband attributes
Output attritbute volumes for each filter bank	Do NOT output attributes for each filter bank
f1 Ormsby broadband filter begin low frequency taper (cycles/s):	1.05736
f2 Ormsby broadband filter end low frequency taper (cycles/s):	1.42541
f3 Ormsby broadband filter begin high frequency taper (cycles/s):	4.7379
f4 Ormsby broadband filter end high frequency taper (cycles/s):	5.10596
Taper applied to filter banks (0-50%)):	25
Number of filter banks applied to the data:	5
Construct linearly or exponentially spaced filter banks?	Exponentially spaced filter banks

Update the filter banks

	f1	f2	f3	f4
Filter Bank 1	2.87875	4.15956	6.01024	8.68433
Filter Bank 2	6.01024	8.68433	12.5482	18.1311
Filter Bank 3	12.5482	18.1311	26.198	37.8541
Filter Bank 4	26.198	37.8541	54.6962	79.0317
Filter Bank 5	54.6962	79.0317	114.195	125
Broad Band Filt	2.87875	4.15956	114.195	125

The separation between filter banks as well as their width and the length of the tapers are all computed in the log(f) domain, such that filter banks centered around higher frequencies appear wider in the linear f domain. When summed, the response of the tapered filter banks equals that of the requested broad band filtered data. The resulting image looks like this (see following page):

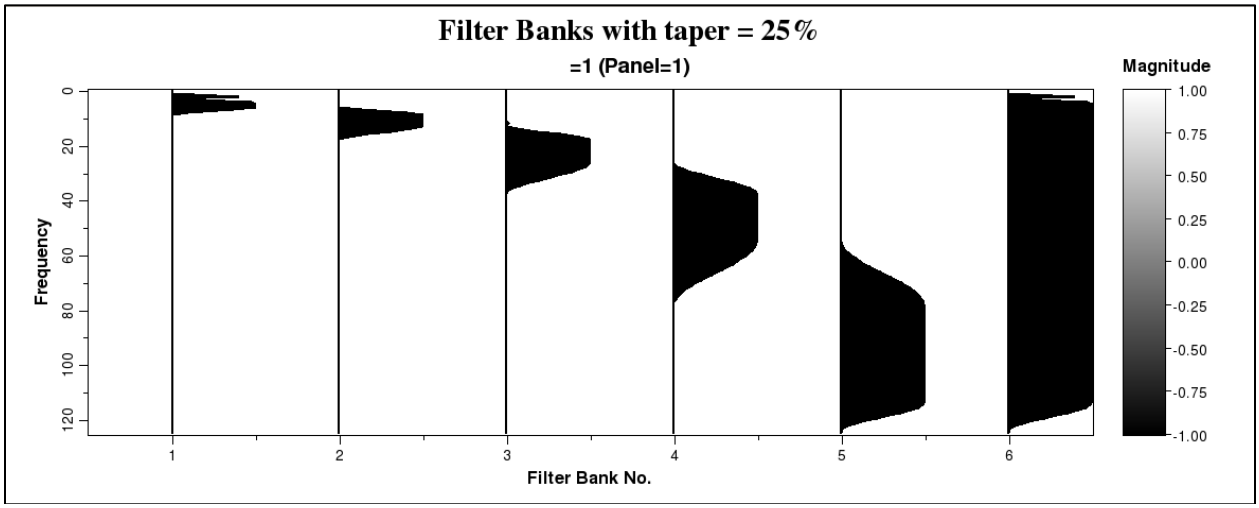


Figure 22.

Filter Design Tip #1: When manually entering changes to the values of *f2_broadband*, *f3_broadband*, the *Taper applied to the filter banks*, and the *Number of filter banks applied to the data*, you will need to click the *Update the filter banks button*.

Filter Design Tip #2: To manually define a suite of filter banks, first select the *Number of filter banks applied to the data* then click the *Update the filter banks button* to obtain the desired number of rows to define the filter banks. Then simply overtype the filter bank corner points directly in the table. Do not click the *Update the filter banks* button or they will revert to the default linearly or exponentially spaced filter banks.

Theory: Multispectral Coherence, Amplitude Gradients, and Laplacians

Dewett and Hensa (2015) combined multiple coherence attribute images using self-organizing maps. Each energy ratio coherence volume was computed along structure from spectral voices, **u(f):**

$$u(f_l, t_k, x_m, y_m) = a(f_l, t_k, x_m, y_m)\exp[i\varphi(f_l, t_k, x_m, y_m)] \quad (23)$$

constructed using the AASPI spectral decomposition algorithm, **spec_cwt**, where **a** is the spectral magnitude and **ϕ** the spectral phase of each component, *l*. These images were subsequently skeletonized using a commercial swarm intelligence algorithm to provide significantly improved fault images in an Eagle Ford Shale survey.

Sui et al. (2015) also noted the value of multispectral coherence and 3-component limitations of RGB display, and computed coherence based on spectral magnitudes, $a(f_l,t_k,x_m,y_m)$, using the covariance matrix

$$C_{mn} = \sum_{l=1}^{L}\sum_{k=-K}^{K}\left[a(t_k, f_l, x_m, y_m)a(t_k, f_l, x_n, y_n)\right], \quad (24)$$

where *L* is the number of spectral components. By not using the phase component, the covariance matrix is less sensitive to dip, allowing the use of a simpler, non-structure-oriented computation.

Program **similarity3d** builds on the above work but rather than using the spectral magnitude computed along time slices used in equation 24, it uses the spectral voices, **u**, and their Hilbert transforms, $\mathbf{u^H}$, computed along structure described above to construct the covariance matrix:

$$C_{mn} = \sum_{l=1}^{L}\sum_{k=-K}^{K}\left[u(t_k, f_l, x_m, y_m)u(t_k, f_l, x_n, y_n) + u^H(t_k, f_l, x_m, y_m)u^H(t_k, f_l, x_n, y_n)\right]. \quad (25)$$

This new covariance matrix is then used to compute "multispectral" energy ratio similarity, outer-product similarity, energy-weighted or RMS amplitude weighted gradient and energy-weighted or RMS amplitude weighted Laplacian.

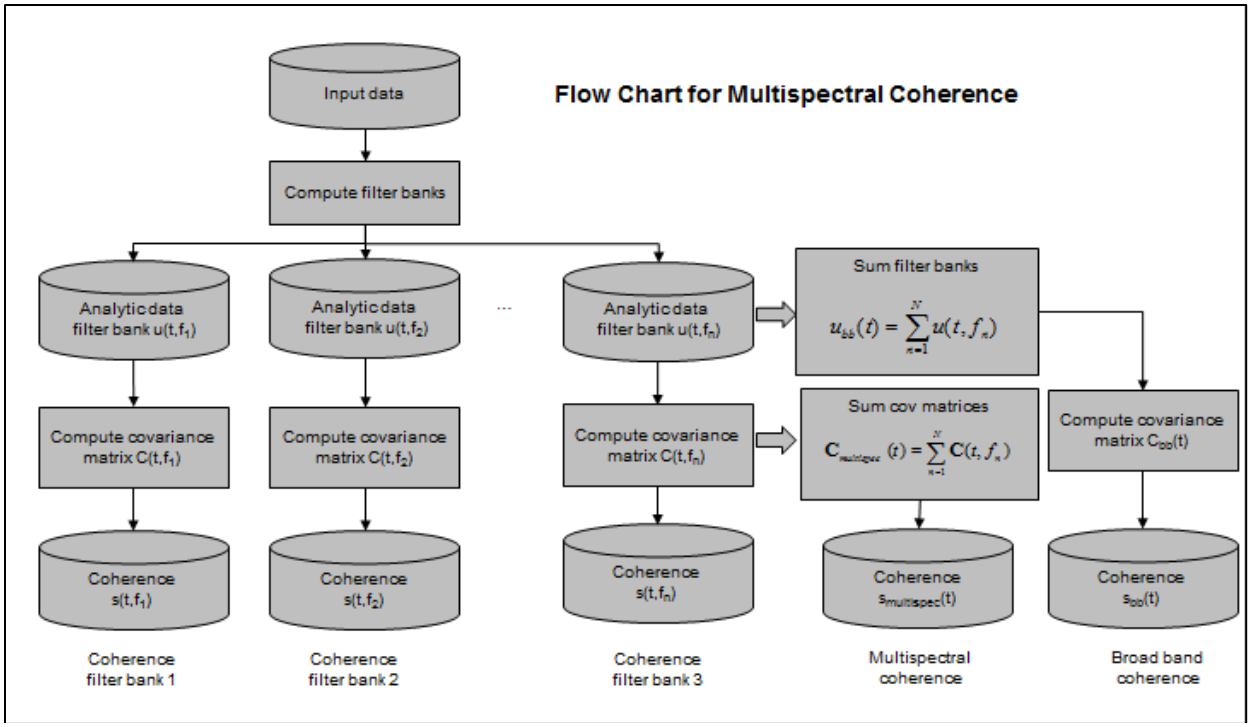


Figure 23.

Theory: Multispectral Sobel Filter Similarity

The Sobel filter similarity for a given frequency component, f_m, is defined as,

$$c_{Sobel}(f_m) = \frac{\sum_{k=-K}^{+K} \left\{ \left[\sum_{j=1}^{J} D_j^{(x)} d_j(f_m) \right]^2 + \left[\sum_{j=1}^{J} D_j^{(y)} d_j(f_m) \right]^2 + \left[\sum_{j=1}^{J} D_j^{(x)} d_j^H(f_m) \right]^2 + \left[\sum_{j=1}^{J} D_j^{(y)} d_j^H(f_m) \right]^2 \right\}^{1/2}}{\sum_{k=-K}^{+K} \left\{ \left[\sum_{j=1}^{J} \left| D_j^{(x)} \right| \left| d_j(f_m) \right| \right]^2 + \left[\sum_{j=1}^{J} \left| D_j^{(y)} \right| \left| d_j(f_m) \right| \right]^2 + \left[\sum_{j=1}^{J} \left| D_j^{(x)} \right| \left| d_j^H(f_m) \right| \right]^2 + \left[\sum_{j=1}^{J} \left| D_j^{(y)} \right| \left| d_j^H(f_m) \right| \right]^2 \right\}^{1/2}}, \quad (26)$$

where the convolution operators $\mathbf{D}^{(x)}$ and $\mathbf{D}^{(y)}$ were defined in a previous box. For M components, we simply sum the numerators and denominators for each component prior to taking the square root:

$$c_{Sobel}^{multispectral} = \frac{\sum_{k=-K}^{+K} \left\{ \sum_{m=1}^{M} \left(\left[\sum_{j=1}^{J} D_j^{(x)} d_j(f_m) \right]^2 + \left[\sum_{j=1}^{J} D_j^{(y)} d_j(f_m) \right]^2 + \left[\sum_{j=1}^{J} D_j^{(x)} d_j^H(f_m) \right]^2 + \left[\sum_{j=1}^{J} D_j^{(y)} d_j^H(f_m) \right]^2 \right) \right\}^{1/2}}{\sum_{k=-K}^{+K} \left\{ \sum_{m=1}^{M} \left(\left[\sum_{j=1}^{J} \left| D_j^{(x)} \right| \left| d_j(f_m) \right| \right]^2 + \left[\sum_{j=1}^{J} \left| D_j^{(y)} \right| \left| d_j(f_m) \right| \right]^2 + \left[\sum_{j=1}^{J} \left| D_j^{(x)} \right| \left| d_j^H(f_m) \right| \right]^2 + \left[\sum_{j=1}^{J} \left| D_j^{(y)} \right| \left| d_j^H(f_m) \right| \right]^2 \right) \right\}^{1/2}}. \quad (27)$$

As with the multispectral covariance matrix, the sum of the ratios of each numerator and denominator is not equal to the ratio of the sums of the numerators and denominators, thereby giving a different answer that is weighed by the strength of each spectral component.

Examples

Using the GSB data volume with eight filter banks and f_width=10 Hz, one obtains the following images through energy ratio coherence:

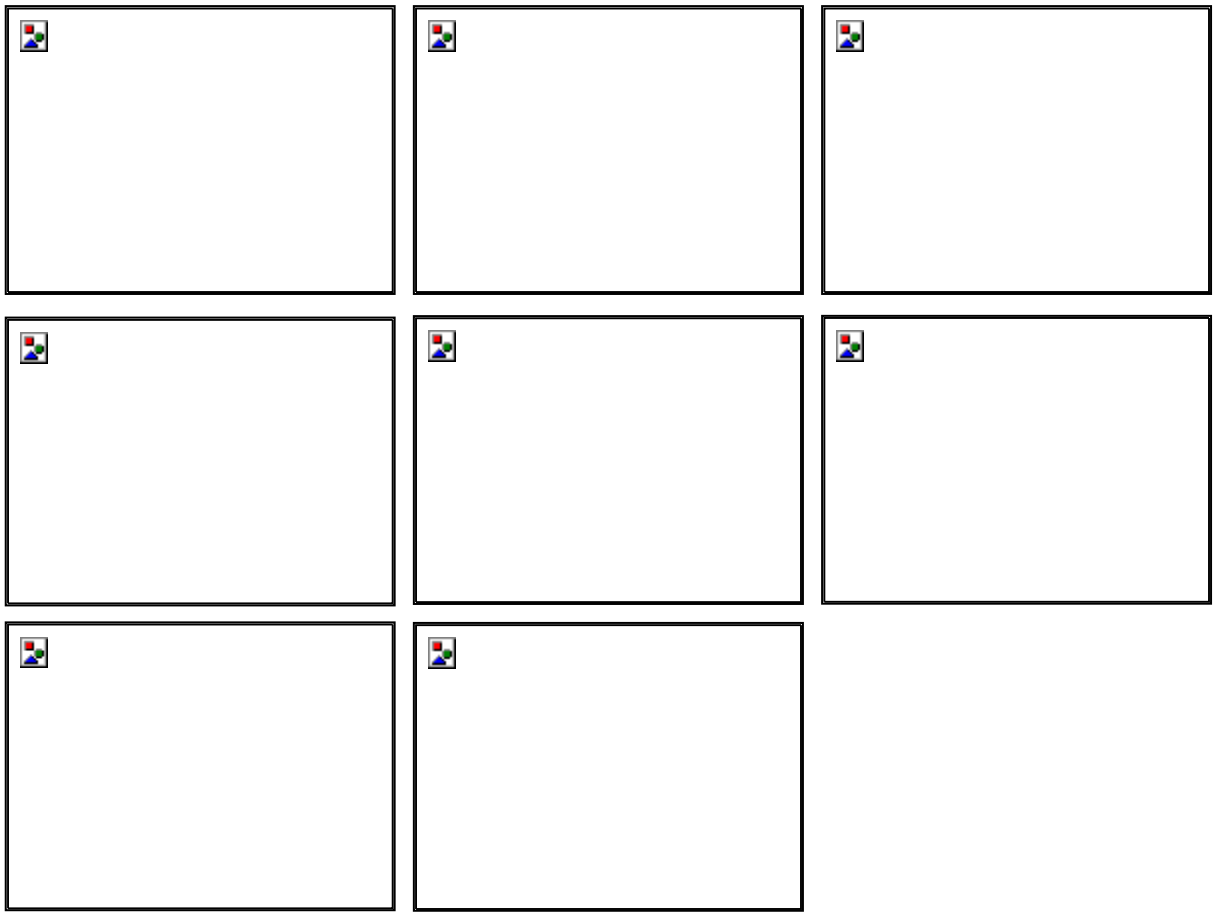


Figure 24.

Co-rendering three coherence images against RGB is somewhat counterintuitive, since high coherence for each component will result in R=1.0, G=1.0, and B=1.0, or white. However, if the highest frequency component is lower, we subtract a little blue from white and end up with yellow. I summarize this arithmetic in the following image on the next page:

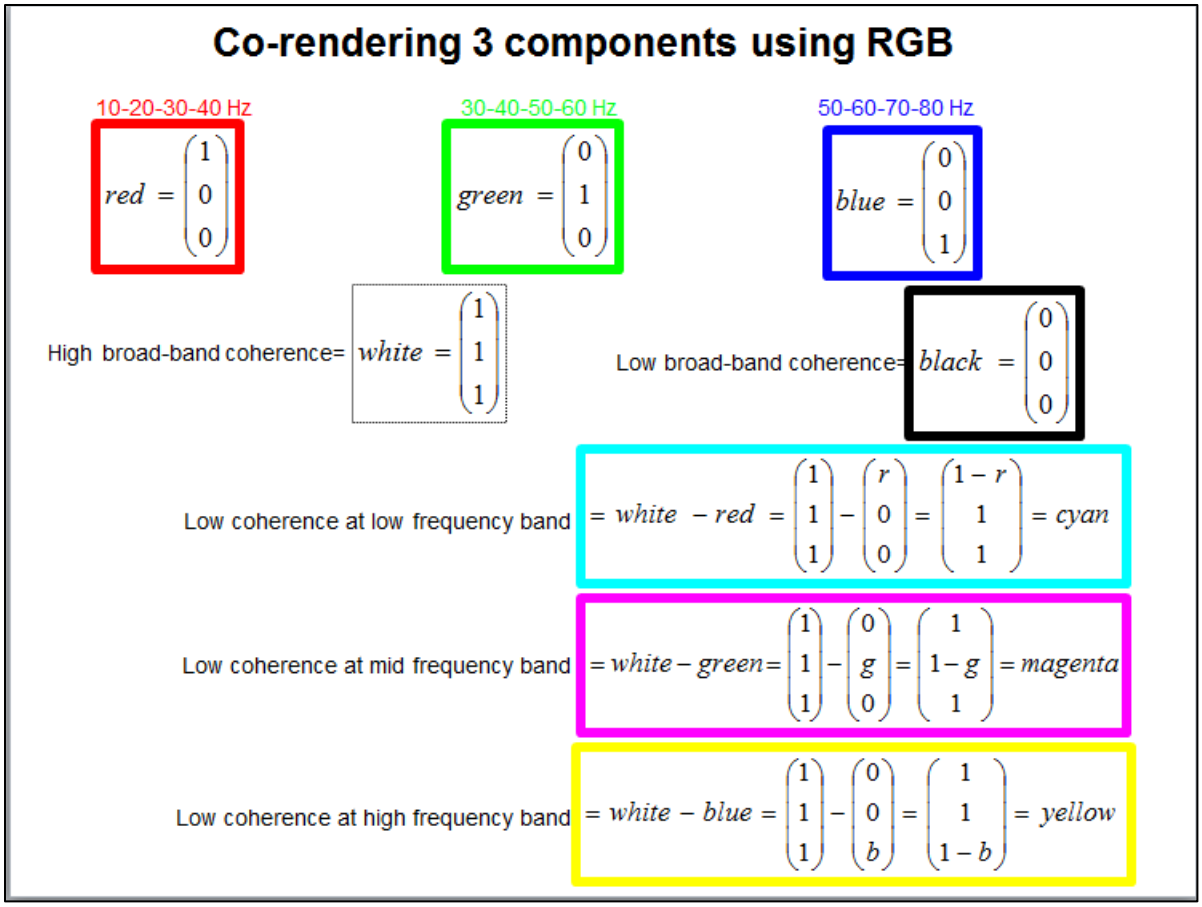

Co-rendering 3 components using RGB
10-20-30-40 Hz
30-40-50-60 Hz
50-60-70-80 Hz
red = (1 0 0)
green = (0 1 0)
blue = (0 0 1)
High broad-band coherence= white = (1 1 1)
Low broad-band coherence= black = (0 0 0)
Low coherence at low frequency band = white − red = (1 1 1) − (r 0 0) = (1−r 1 1) = cyan
Low coherence at mid frequency band = white − green = (1 1 1) − (0 g 0) = (1 1−g 1) = magenta
Low coherence at high frequency band = white − blue = (1 1 1) − (0 0 b) = (1 1 1−b) = yellow

Figure 25.

Co-rendering three of the spectral components using AASPI program **corender** against RGB, one obtains the following image:

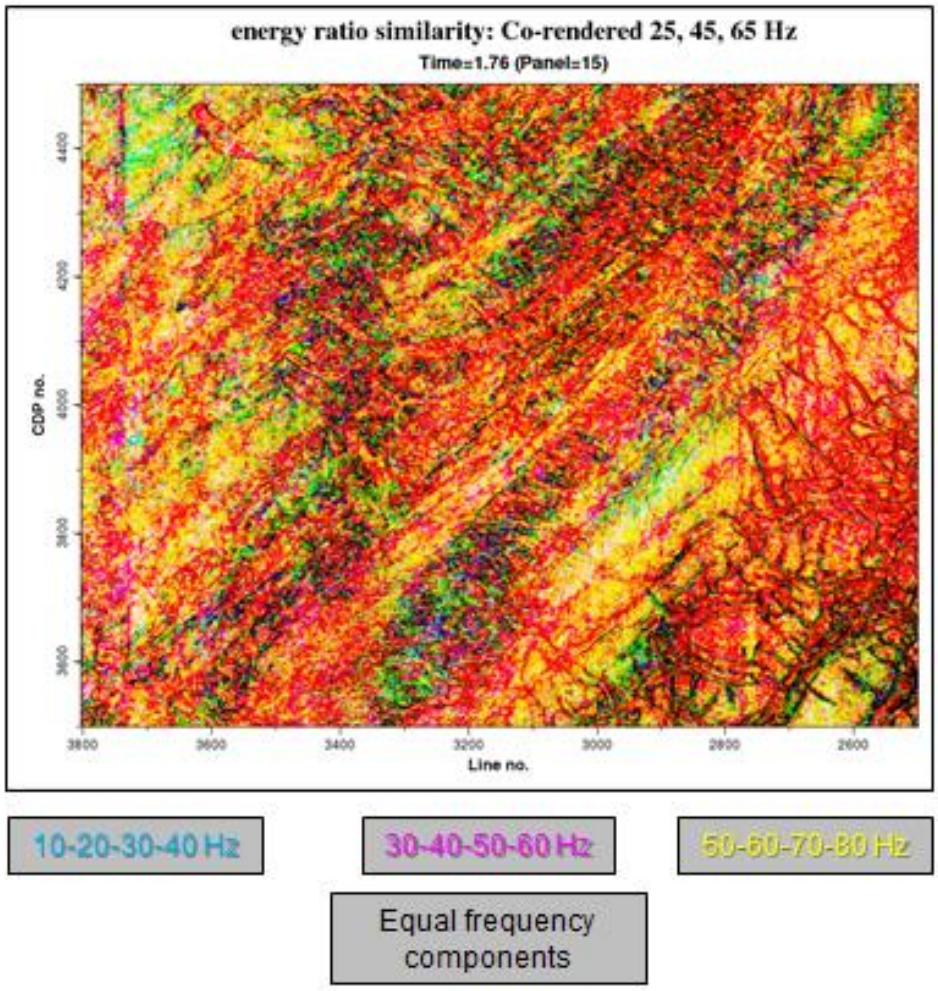


Figure 26.

Areas that appear to be yellow indicate that the higher frequencies are less coherent. Areas that that appear to be red indicate that both the mid and the high frequencies are less coherent (leaving the red component). Areas that appear to be green indicate that both the low and the high frequencies are less coherent (leaving the green component). Finally, areas that appear to be black are less coherent for all three frequency filter banks. Next, compare the previous image to multispectral coherence:

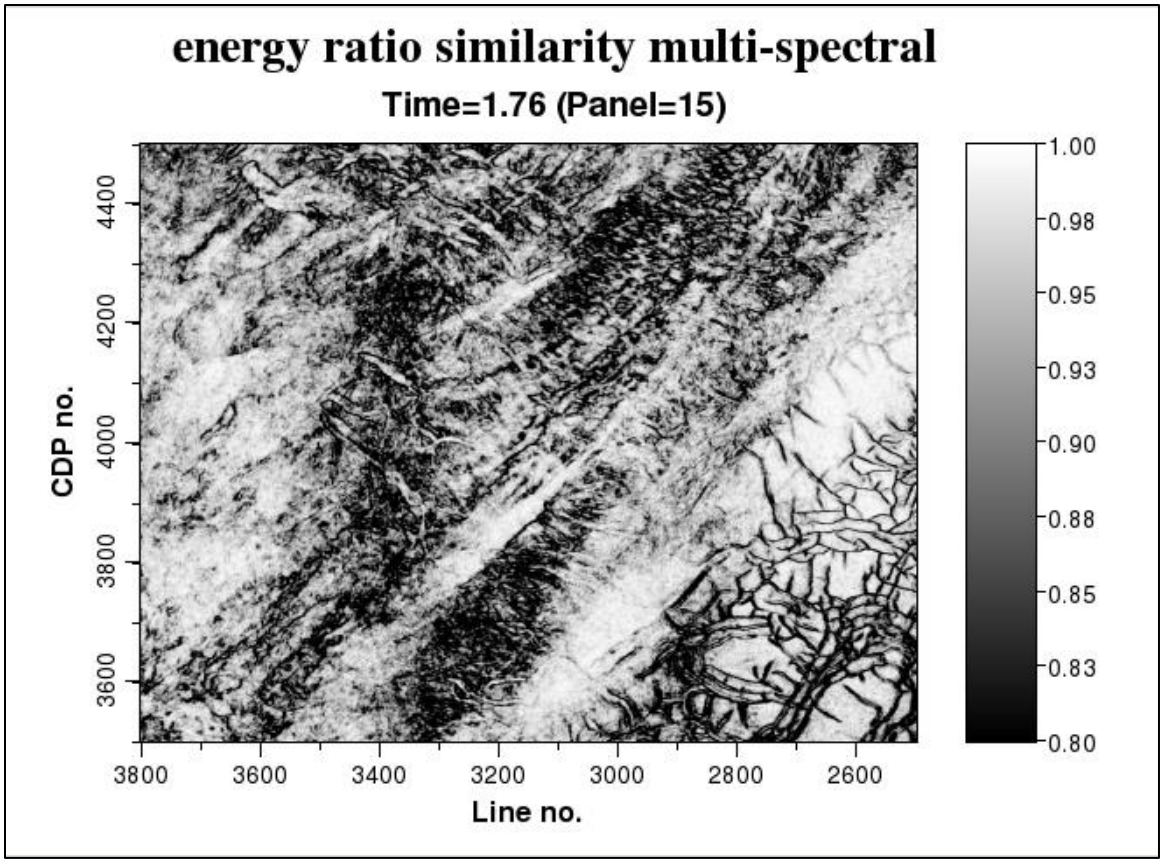


Figure 27.

At this point, I see a few more (smaller?) faults in the combined RGB image, but the channels show up better here. This appearance may be due to scaling, so I have changed my color bar range to be from 0.80 to 1.00. Let us compare this image to the previously computed broad-band (conventional) coherence image, but now also scaled to range between 0.80 and 1.00:

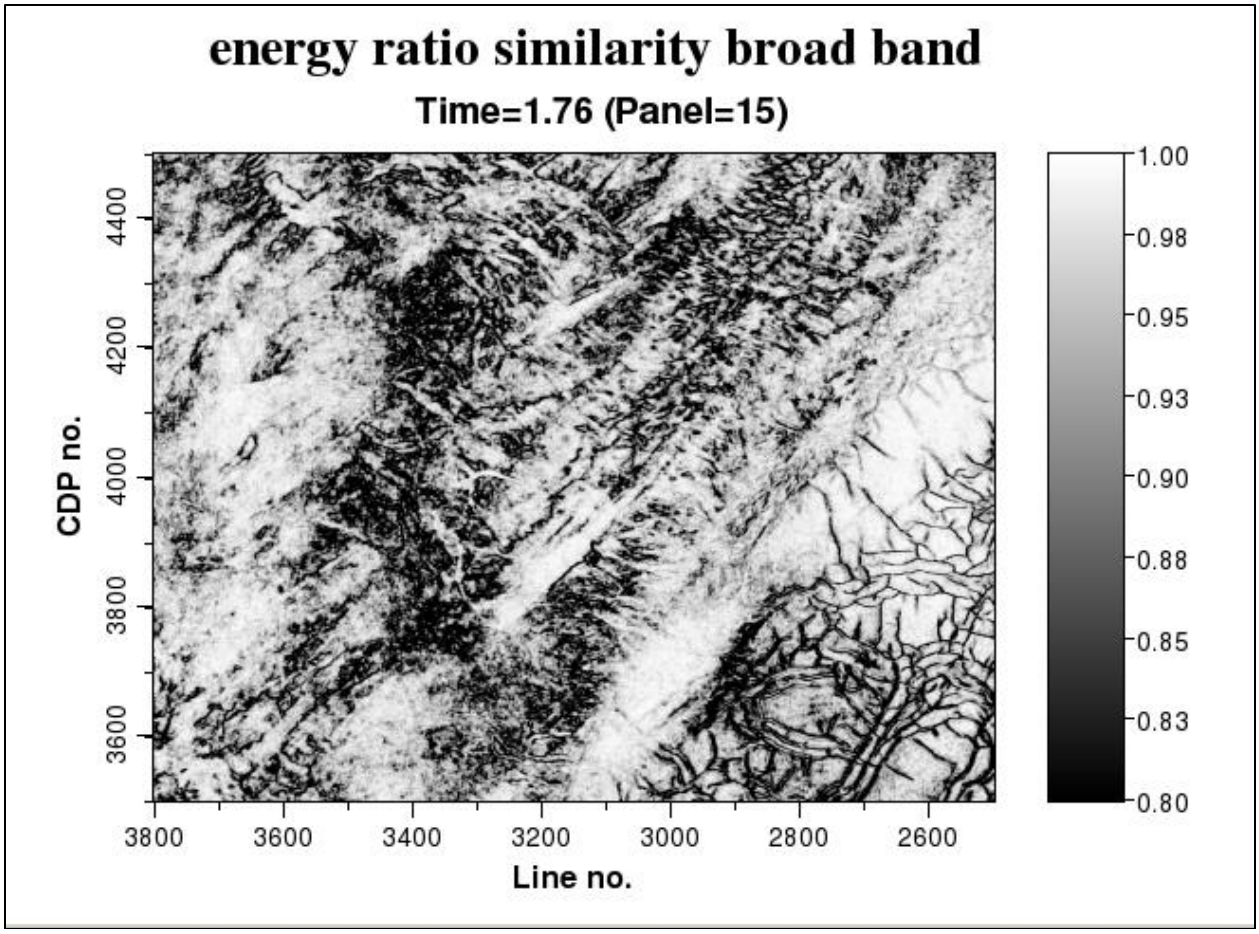
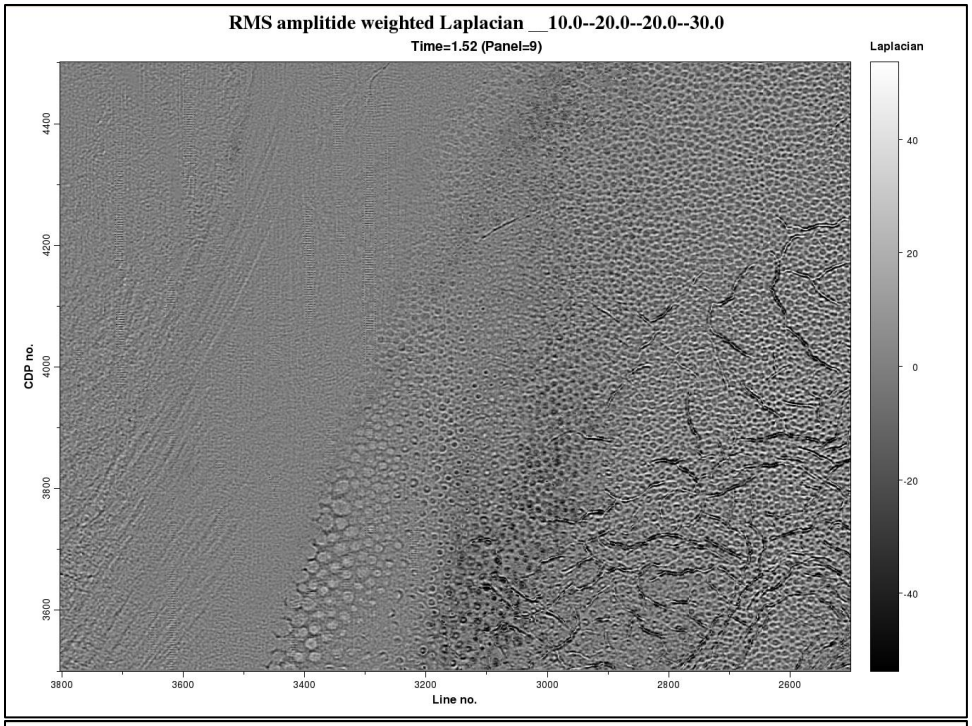
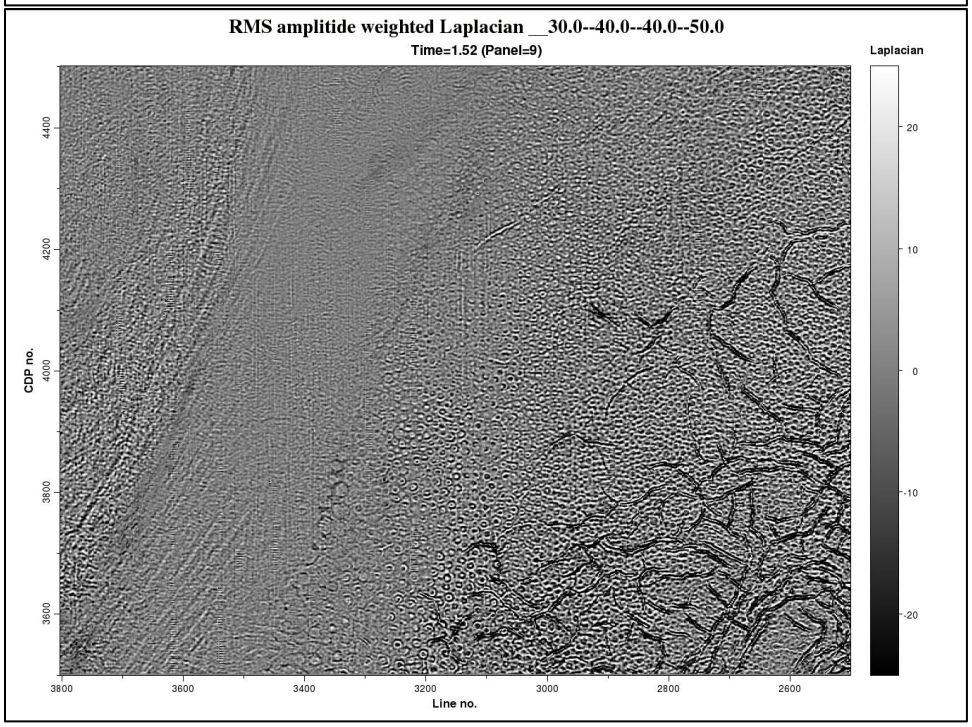


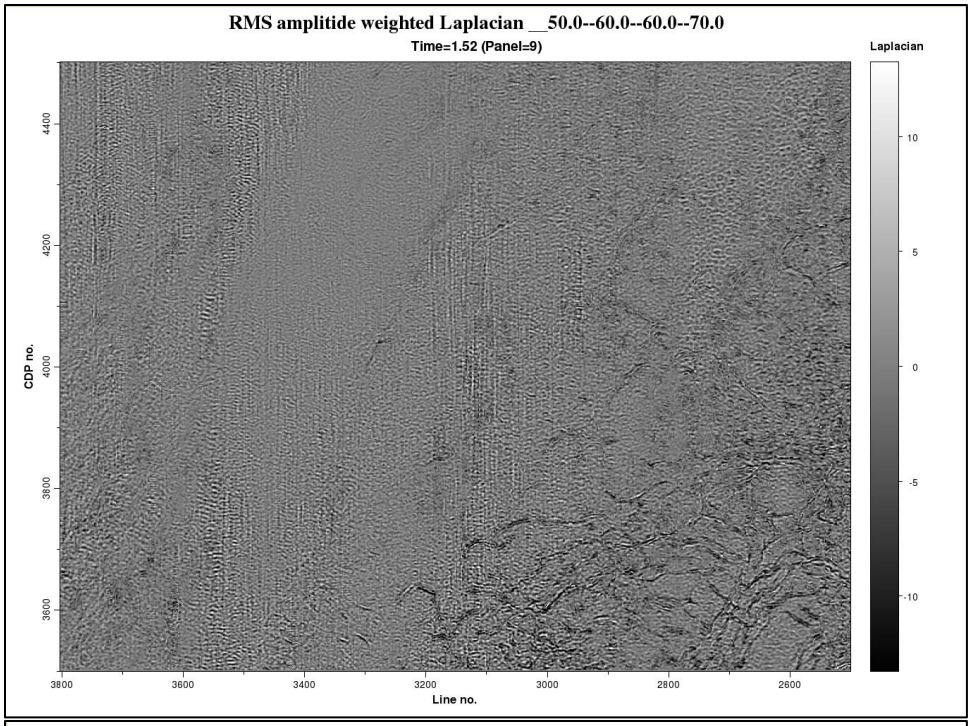
Figure 28.

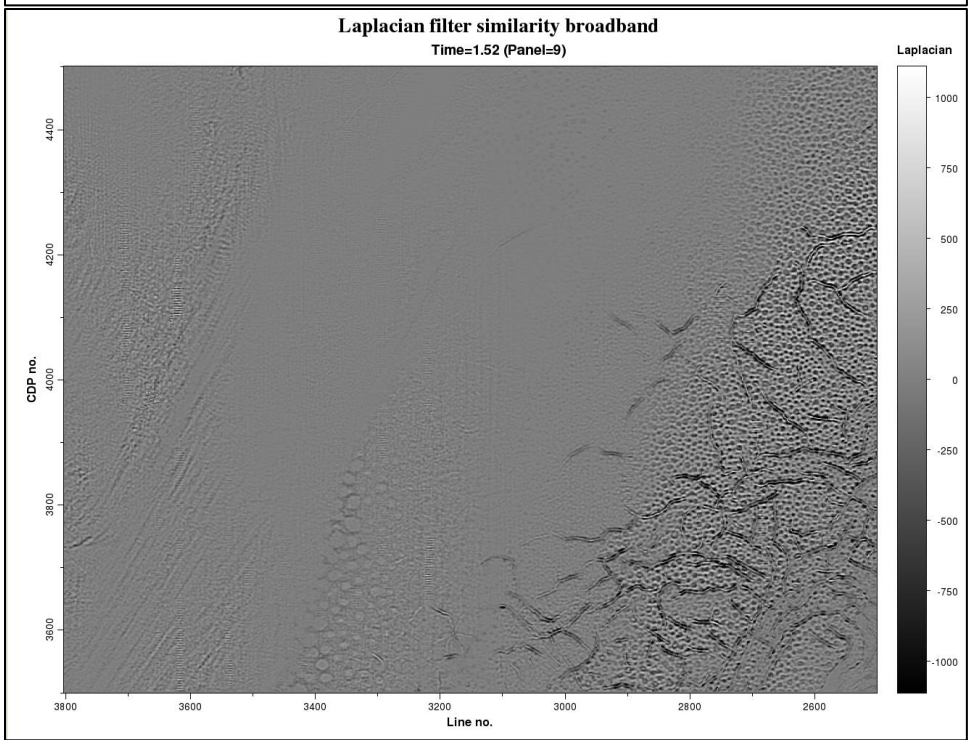
In general, background noise has been reduced and many of the channel features appear sharper.

Next, let us examine the Laplacian at different frequency components:


RMS amplitide weighted Laplacian __10.0--20.0--20.0--30.0
Time=1.52 (Panel=9)
Laplacian
40
20
0
-20
-40
CDP no.
4400
4200
4000
3800
3600
Line no.
3800
3600
3400
3200
3000
2800
2600

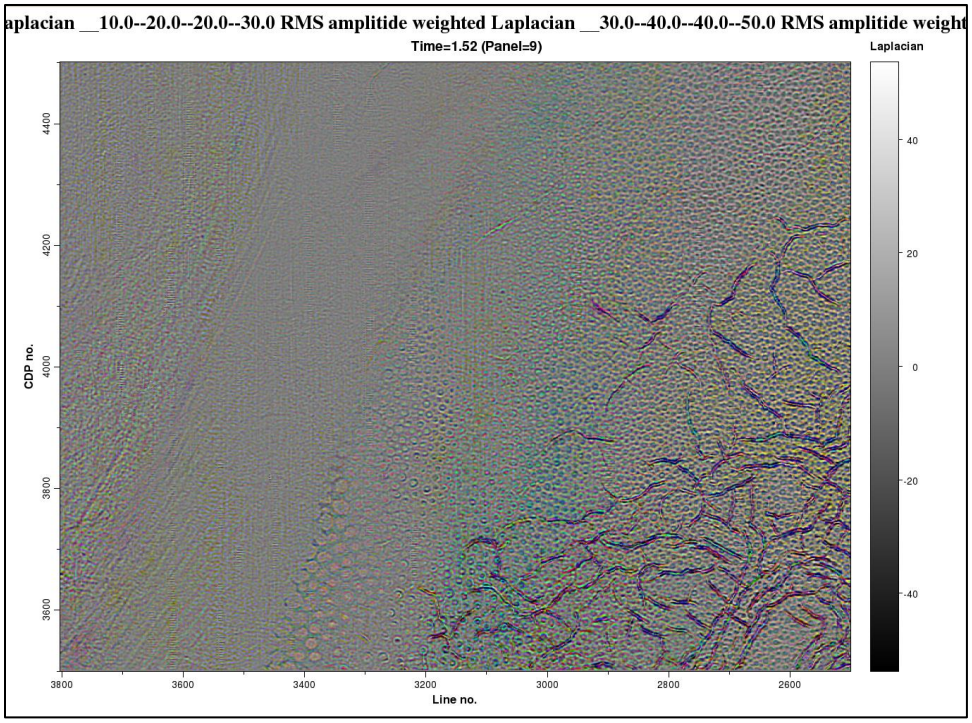

RMS amplitide weighted Laplacian __30.0--40.0--40.0--50.0
Time=1.52 (Panel=9)
Laplacian
20
10
0
-10
-20
CDP no.
4400
4200
4000
3800
3600
Line no.
3800
3600
3400
3200
3000
2800
2600



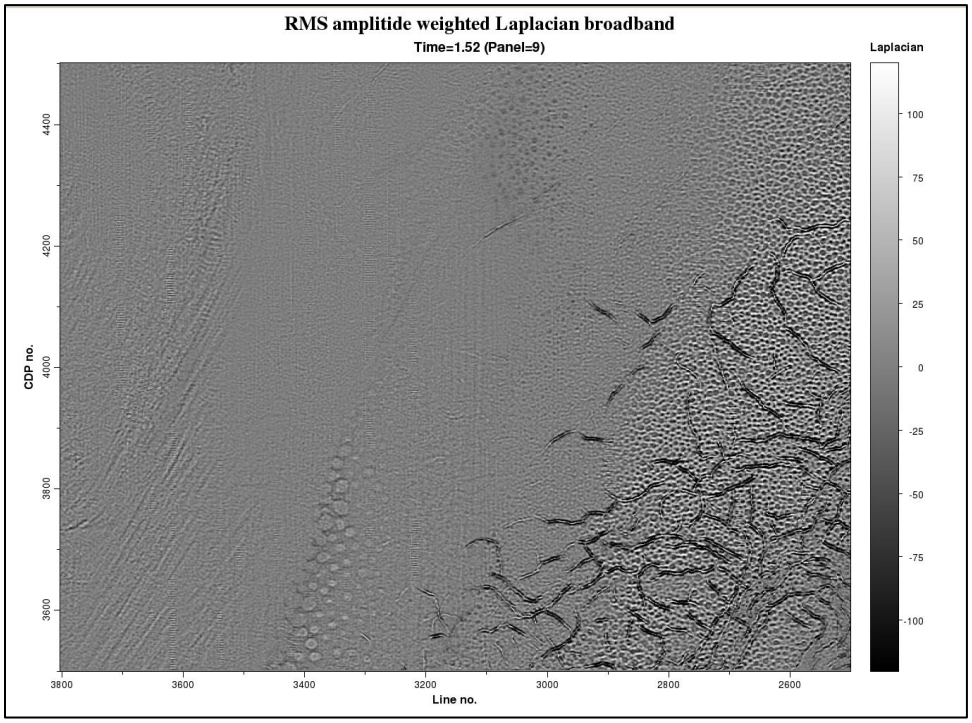


Corendering these three images using RGB gives

Geometric Attributes: Program **similarity3d**

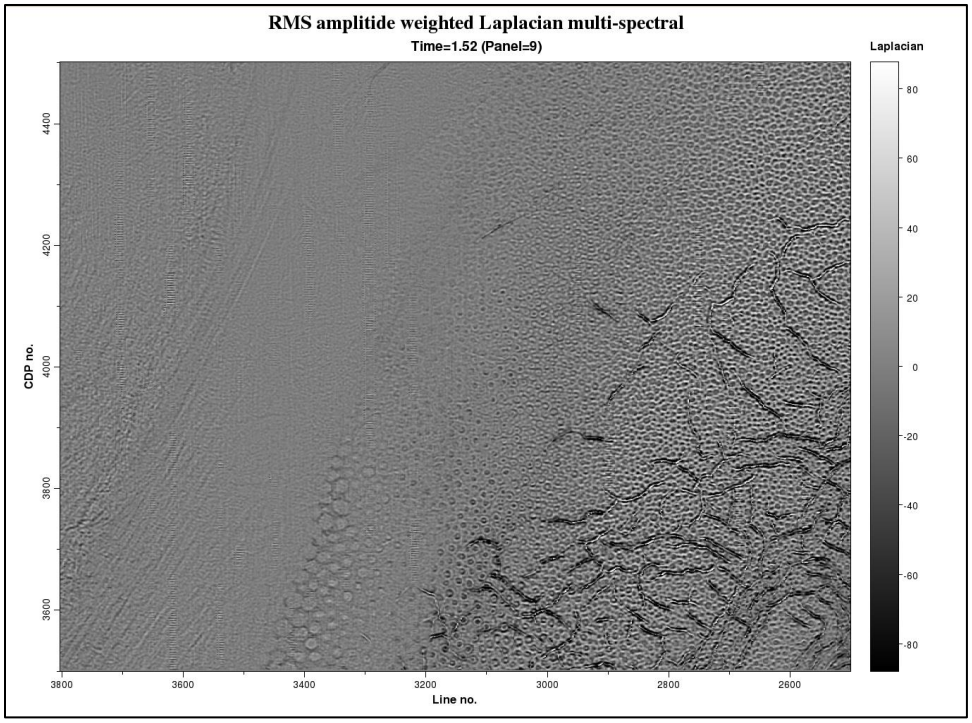


The broadband Laplacian looks like this:

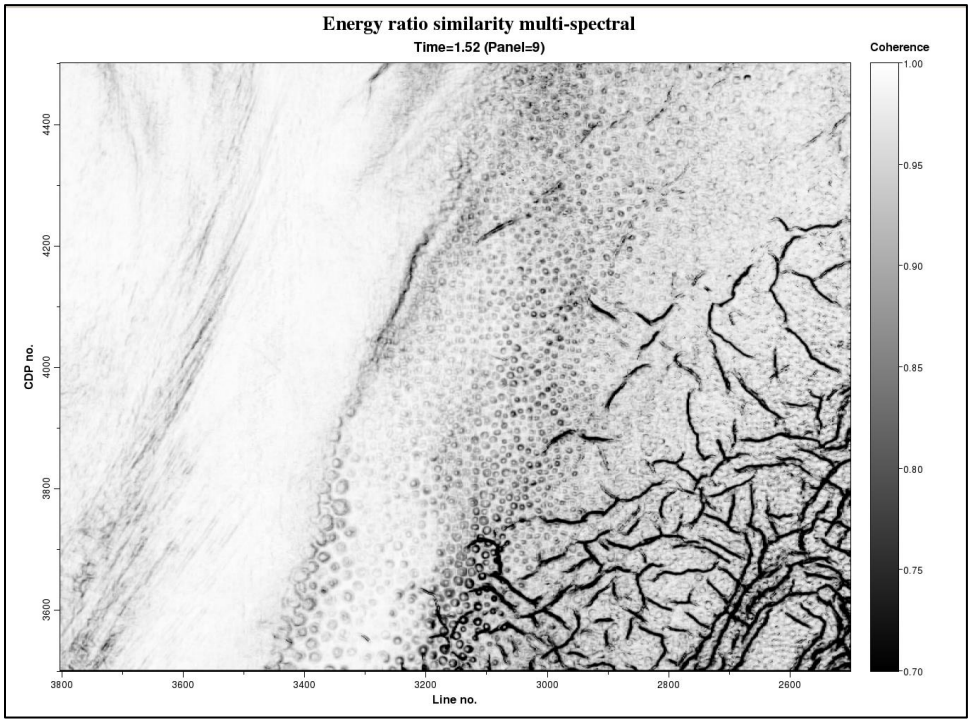


Finally, the multispectral Laplacian gives:

Geometric Attributes: Program **similarity3d**



where we next plot the multispectral coherence image for comparison:



Pitfalls and Limitations

Stair Step Artifacts

The stair step artifacts seen on vertical slices through coherence are “annoying”, resulting in fault anomalies picked on coherence time slices being laterally shifted from those picked on vertical slices through the seismic amplitude volume. The following images show a suite of vertical slices through a seismic amplitude volume co-rendered with coherence computing from the GSB survey using a 5-trace by ±40 ms. The sample increment = 4 ms, and the bin size =12.5 m x 25 m. Note the stair-step artifacts indicated by the red circles. Folk wisdom attributes the stair step anomalies to the size of the vertical analysis window, which while oriented along structural dip, typically consists of vertical trace segments forming oblique cylinders. We have attempted to address this problem by rotating windows, designing data-adaptive windows, and other programming-intensive endeavors, all to no avail. Closer analysis shows that the hypothesis of the stair steps is an effect of the vertical analysis window size and orientation to be incorrect, with even windows of height 0.0 s (one sample thick) exhibiting the artifact. Instead, stair steps turn out to be due to the way the seismic wavelet is imaged (see the subsequent theory box). If the reflectors are perpendicular to the fault plane, there are no stair step artifacts. In contrast, if the faults cut at angle to the reflectors, the stair steps orient themselves perpendicular to the reflector, wavelet by wavelet.

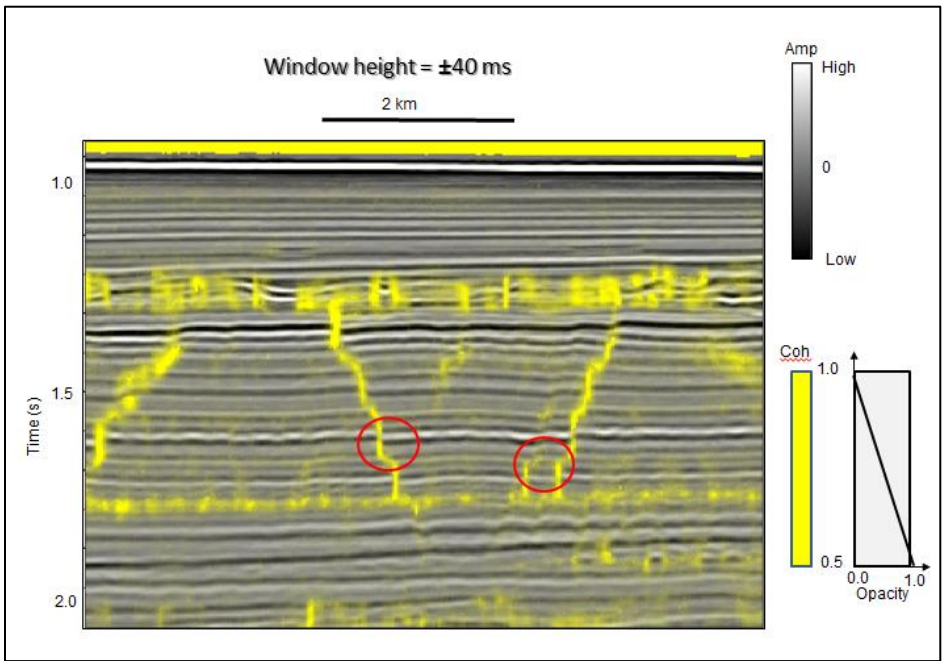


Figure 29.

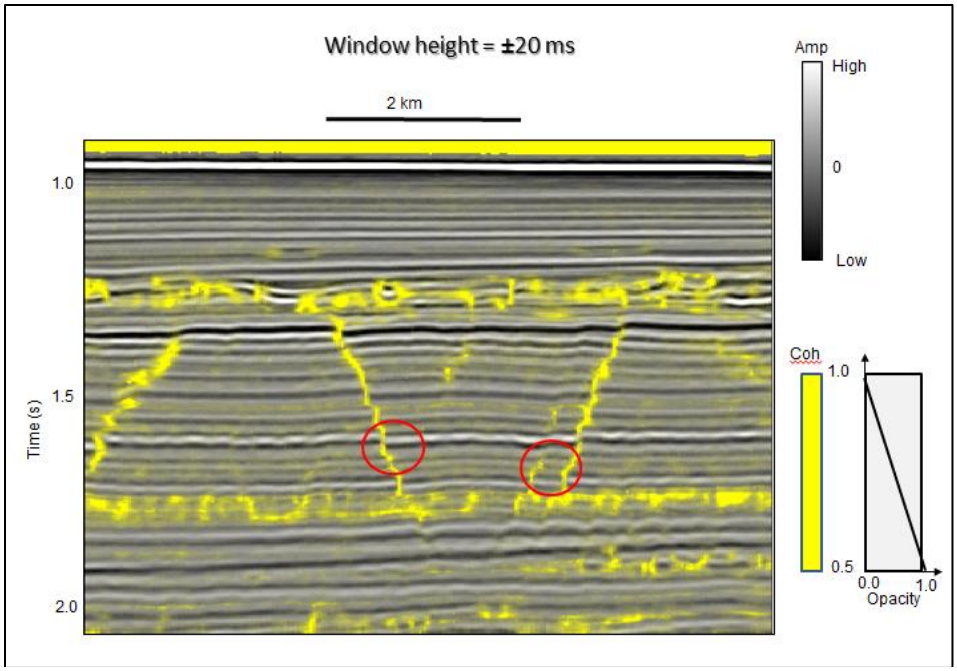


Figure 30.

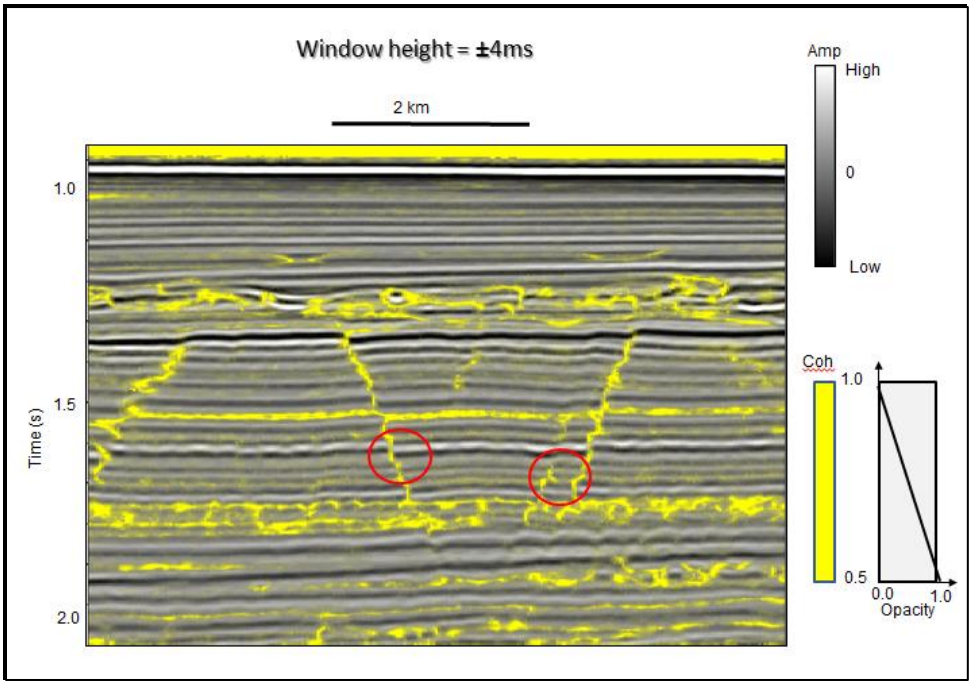


Figure 31.

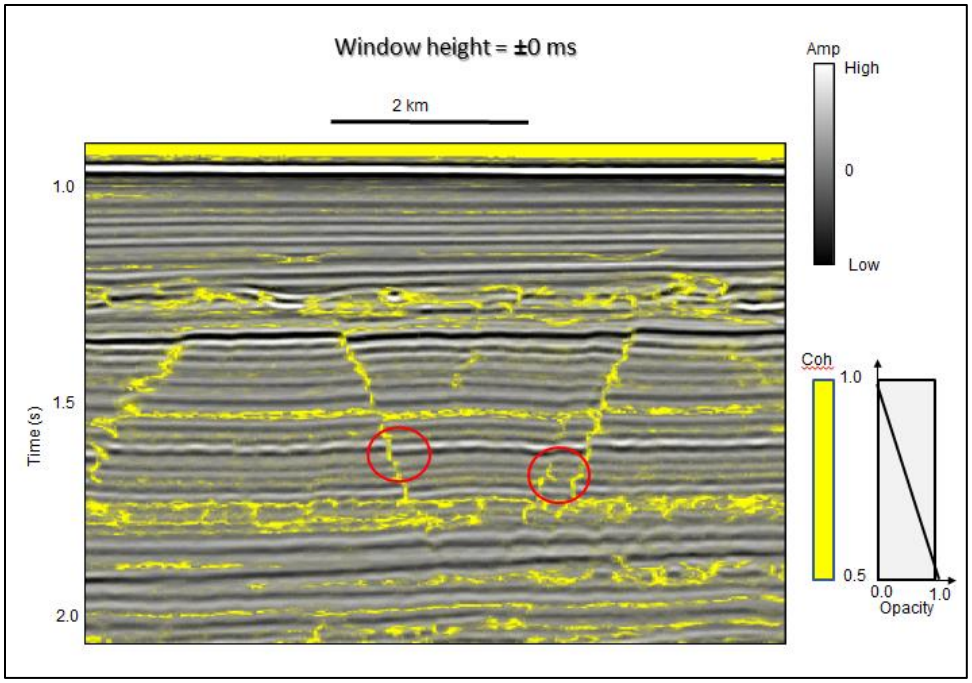
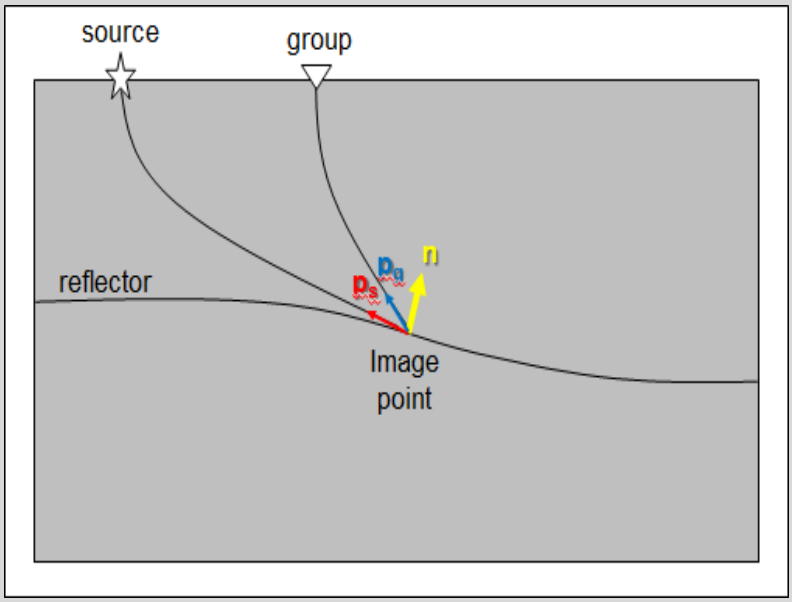

Window height = ±0 ms
2 km
Amp
High
0
Low
Coh
1.0
0.5
0.0
1.0
Opacity
Time (s)
1.0
1.5
2.0

Figure 32.

Theory: Seismic Imaging of Reflectors

The image bellow shows the geometry of seismic migration, using the notation of the diffraction imaging community. **n** defines the normal to the hypothesized reflector at the image point. If no hypothesis is made, most algorithms assume **n** to be vertical, while some eliminate the obliquity factor completely. $\mathbf{p}_s$ and $\mathbf{p}_g$ define ray parameter unit vectors between the source and the geophone group to the image point. The obliquity factor is the cosine of the angle between the yellow vector and the average of the blue and red vectors.



Diffraction imaging tells us that the seismic wavelet will be oriented perpendicular to the reflector. Thus, if a fault is not perpendicular to an interface, the wavelet will give rise to a stair-step response at each reflector event! Nothing can change this except to increase vertical resolution.

Redundant Attributes

The examples show here, that in general, energy ratio similarity, outer product similarity, and Sobel filter similarity provide very similar results. Barnes (2007) inspired discussion through his paper on redundant and useless attributes. The problem of "useless" attributes arises with improvements in technology, with early attempts to map seismic features having been supplanted by more quantitative and higher resolution with further development, such as the progression from three-trace cross-correlation to multi-trace semblance and eigenstructure coherence. A cursory inspection of the three major coherence attributes described here – outer product similarity, energy ratio similarity, and Sobel filter similarity, shows that they are somewhat redundant. All provide good images of faults, channel edges, and syneresis (shale dewatering) features. Mathematically, these three attributes are slightly different; however, they are closely coupled through the underlying geology. Energy ratio similarity measures lateral changes in waveform but is independent of amplitude. Semblance and Sobel filter

similarity measure are sensitive to changes in amplitude. However, a change in the waveform gives rise to changes in amplitude as well.

To determine which attribute works "best", one needs to think of end members. Common end members are thin channel systems that fall below the tuning thickness. Internal to the channel, the waveform will be identical with only the amplitude changing. In this case, edges of architectural elements internal to the channel will be mapped by semblance and Sobel filter similarity, but not by energy ratio similarity. Other end members are faults where the amplitude fades in and out, perhaps associated with the Fresnel zone. Here, the energy ratio similarity will show sharper edges while the semblance and Sobel filter similarity appear somewhat blurred.

References

Bahorich, M. S., and S. L. Farmer, 1995, 3D seismic coherency for faults and stratigraphic features: The Leading Edge, **14**, 1053-1058.

Barnes, 2007, Redundant and useless attributes: Geophysics, **72**, P33-P38.

Barnes, A. E., 2011, Displaying Seismic Data to Look Like Geology: GCSSEPM 31st Annual Bob. F. Perkins Research Conference on Seismic attributes – New views on seismic imaging: Their use in exploration and production, 100-119.

Chopra, S., and K. J. Marfurt, 2019, Multispectral, multiazimuth, and multioffset coherence attribute applications: Interpretation, **7**,

Chopra, S., and K. J. Marfurt, 2007, Seismic attributes for prospect identification and reservoir characterization: SEG Geophysical Development Series, 11.

Davogustto, O., and K. J. Marfurt, 2011, Footprint suppression applied to legacy seismic data volumes: to appear in the GCSSEPM 31st Annual Bob. F. Perkins Research Conference on Seismic attributes – New views on seismic imaging: Their use in exploration and production, 1-34.

Gao, D. [2013] Wavelet spectral probe for seismic structure interpretation and fracture characterization: A workflow with case studies. Geophysics, 78, O56-O67.

Gersztenkorn, A., and K. J. Marfurt, 1999, Eigenstructure-based coherence computation as an aid to 3-D structural and stratigraphic mapping, Geophysics, 64, no. 5, 1468-1479.

Gersztenkorn, A., and Marfurt, K. J., 1995, Eigenstructure-based coherence computation as an aid to 3-D structural and stratigraphic mapping. Geophysics, 64, 1468-1479.

Hardage, B., 2009, Frequencies are fault finding factors: Looking low aids data interpretation: AAPG Explorer, **30**, no 9, 34.

Hemon, C. H., and D. Mace, 1978, use of the Karhunen-Loeve transformation in seismic data processing: Geophysical Prospecting, **26**, 600-626.

Kirlin, R. L., and W. J. Done, 1999, Covariance analysis for seismic signal processing: Geophysical Developments Series, SEG, 355p**.**

Li, F., and W. Lu, 2014. Coherence attribute at different spectral scales: Interpretation, **2**, 1-8.

Li, F., J. Qi, B. Lyu, and K. J. Marfurt, 2018, Multispectral coherence: Interpretation, **6**, T61-T69.

Lin, T., 2016, Seismic attribute analysis of depth-migrated data: Ph.D. dissertation, The University of Oklahoma.

Lin, T., B. Zhang, and K. J. Marfurt, 2019, Geometric seismic attribute estimation using data-adaptive windows: Interpretation, **7**, https://library.seg.org/doi/abs/10.1190/int-2018-0114.1

Lin, T., T. Ha, K. J. Marfurt, and K. L. Deal, 2016, Quantifying the significance of coherence anomalies, Interpretation, **4**, T205-T213.

Luo, Y., S. al-Dossary, M. Marhoon, and M. Alfaraj, 2003, Generalized Hilbert transform and its application in geophysics: The Leading Edge, 22, 198-202.

Luo, Y., W. G. Higgs, and W. S. Kowalik, 1996, Edge detection and stratigraphic analysis using 3-D seismic data: 66th Annual International Meeting, SEG, Expanded Abstracts, 324–327.

Lyu, B., J. Qi, F. Li, Y. Hu, T. Zhao, S. Verma, and K. J. Marfurt, 2020, Multispectral coherence – Which decomposition should we use?: Interpretation, 8, T115-T129.

Lyu, B., J. Qi, S. Sinha, J. Li, and K. J. Marfurt, 2020, Improving fault delineation using maximum entropy multispectral coherence, Interpretation, **8**, T835-T850.

Marfurt, K. J., 2006: Robust estimates of reflector dip and azimuth: Geophysics, **71**, P29-P40.

Marfurt, K. J., R. L. Kirlin, S. L. Farmer, and M. S. Bahorich, 1998, 3-D seismic attributes using a semblance-based coherency algorithm: Geophysics, **63**, 1150-1165.

Sui, J.-K., X. D. Zheng, and Y.-D. Li, 2015, A seismic coherency method using spectral attributes: Applied Geophysics, **12**, no. 3, 353-361.