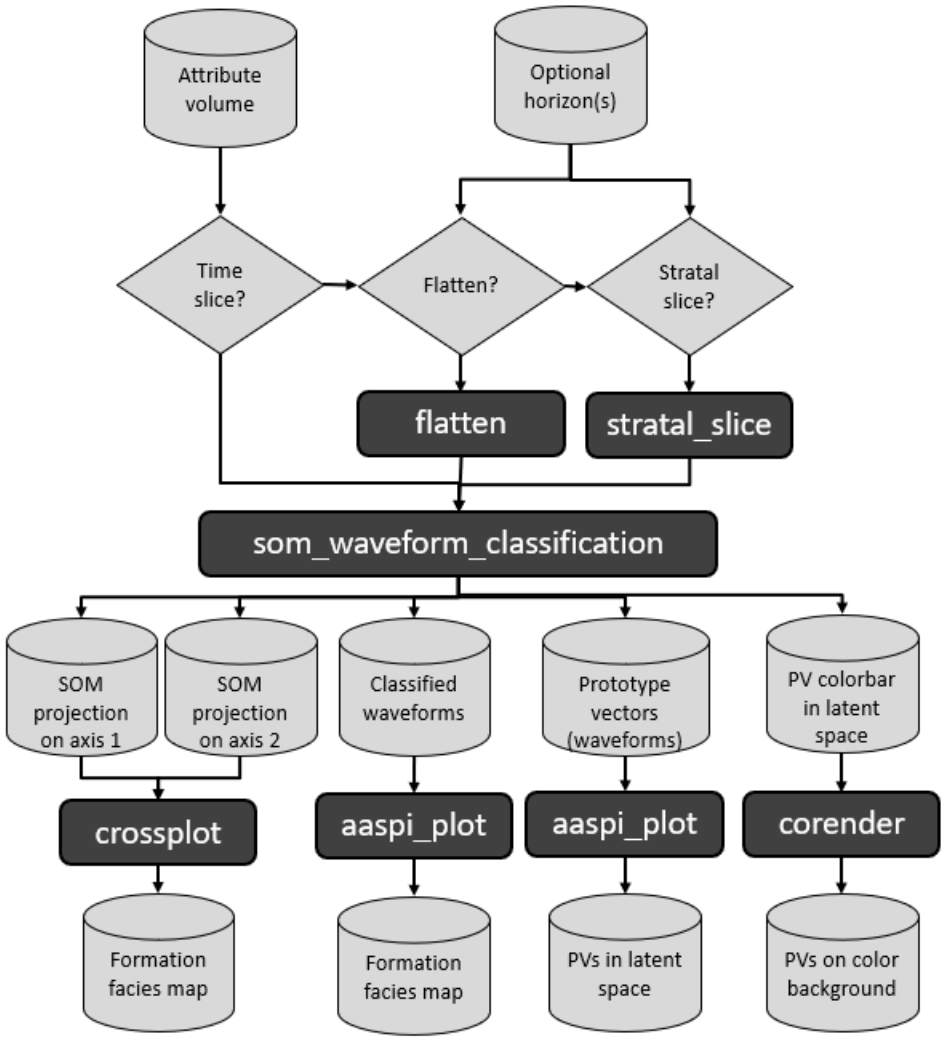
Formation Self-Organizing Maps for seismic facies analysis – PROGRAM som_waveform_classification

Contents

Computation flow chart 1
Output file naming convention 2
Theory 4
Running program som_waveform_classification and plotting the results 5
The *Primary parameters* tab 6
The *Temporal operation window* tab 7
Step 1: Execute program som_waveform_classification 8
Step 2: Plotting the classification results 10
Step 3. Plotting the prototype vectors against their color at the last iteration 11
Step 4. Plotting the location of the prototype vectors in the latent space at each iteration 13
A more flexible display option in interpretation workstations: Crossplotting the results 15
Visualization by crossplotting two SOM axes in Petrel 17
References 18

Computation flow chart

Program **som_waveform_classification** computes a 2D seismic facies map from a suite of seismic sample extracted about a time slice, horizon slice, or between two horizons using an unsupervised self-organizing mapping algorithm. The input can be seismic amplitude, impedance, Poisson's ratio or other volumes that exhibit lateral changes in waveform or geologic stacking patterns about the horizon. Each time, phantom horizon, or stratal slice represents an "attribute' in *N*-dimensional space. The centroids of the found classes are usually displayed as an *N*-dimensional vector, or wavelet, giving rise to the name "wavelet classification".. Below is the flowchart showing the workflow of 2D seismic facies analysis.



Output file naming convention

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

Output file description	File name syntax
Program log information	som_waveform_classification_*unique_project_name_suffix*.log
Program error/completion information	som_waveform_classification *unique_project_name_suffix*.err
Waveform eigenvectors	waveform_eigenvectors_*unique_project_name_suffix*.H
Waveform eigenvalues	waveform_eigenvalues_*unique_project_name_suffix*.H
Classified data	som_waveform_classification *unique_project_name_suffix*.H
Classified data projected on SOM axis 1	som_waveform_classification_axis1_ *unique_project_name_suffix*.H

Classified data projected on SOM axis 2	som_waveform_classification_axis2_ *unique_project_name_suffix*.H
Waveforms projected on latent space	som_waveforms_projected_on_latent_space_*unique_project_name_suffix*.H
Scaled prototype vectors	scaled_prototype_vector_waveform_*unique_project_name_suffix*.H
Unscaled prototype vectors	unscaled_prototype_vector_waveform_*unique_project_name_suffix*.H
Protype vector color matrix	prototype_vector_color_matrix_*unique_project_name_suffix.*H
Classification color bars	som_waveforms_colors_*unique_project_name_suffix*.alut

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.

SOM classification is initialized using the first two eigenvalues and eigenvectors, and in this application are identical to those generated by program **pca_waveform_classification**. This 2D plane (the simplest manifold in N-dimensional attribute space) is sampled by a suite of regularly spaced prototype vectors which are then projected onto the SOM latent space. At each iteration, the location of each prototype vector moves in the *N*-dimensional space to better represent the training data. These prototype vectors (some workers call them “neurons”) are then projected onto the 2D latent space at each iteration. Each sample in the input data represents a time slice, phantom horizon slice, or stratal slice. In order to classify, the input data are scaled using the mean and standard deviation for each slice. For this reason, there are two versions of the prototype vector waveforms – the one that is scaled and used internal to the program, and the one that is unscaled (in “world coordinates”) and is more useful to an interpreter. Both of these waveforms can be plotted against a color map called the *prototype vector color matrix*. The classified results are provided in two formats – as a labeled data volume (consisting of integer values stored as floating point numbers) that can be plotted against a corresponding classification color bar, or as the classes projected against SOM latent space axes 1 and 2, which can be plotted using aaspi_crossplot or crossplot tools available in commercial software. Most commercial software packages allow an interpreter to define polygons in the crossplot space, thereby providing more control in constructing seismic facies.

As with programs **rgb_cmy_plot**, **crossplot**, and **hlsplot**, the user can request the following optional colorbars for the more common interpretation software packages:

Output file description	File name syntax
Petrel classification color bars	som_waveforms_colors_*unique_project_name_suffix*.iesx
Landmark classification color bars	som_waveforms_colors_*unique_project_name_suffix*.cl2

Kingdom Suite classification color bars	som_waveforms_colors_*unique_project_name_suffix*.CLM
Seisware classification color bars	som_waveforms_colors_*unique_project_name_suffix*.xml
Voxelgeo classification color bars	som_waveforms_colors_*unique_project_name_suffix*.color
Geoprobe classification color bars	som_waveforms_colors_*unique_project_name_suffix*.gpc
Transform classification color bars	som_waveforms_colors_*unique_project_name_suffix*.cmp
Geomodeling classification color bars	som_waveforms_colors_*unique_project_name_suffix*.geomodeling
Seisware classification color bars	som_waveforms_colors_*unique_project_name_suffix*.CLM

Because the AASPI software uses the Petrel *.alut format files for its display; this file will always be generated.

Theory

Self-organizing mapping (SOM) is closely related to vector quantization methods (Haykin, 1999). Initially we assume that the J input data vectors are represented by smaller number of P prototype vectors (or "neurons") in an N-dimensional attribute space R_n, $\mathbf{x}_j = [\mathbf{x}_{j1}, \mathbf{x}_{j2}, \mathbf{x}_{j3} \ldots \mathbf{x}_{jN}]$ where N is the number of input attributes (or amplitude samples for "waveform" classification). Each of the $j=1,2,\ldots,J$ input data vectors are represented by a point in N-dimensional space. The seismic response of similar stratigraphy results in waveforms that are similar and points in N-dimensional space that "cluster" together. The objective of the SOM algorithm is to locate the centroids of these clusters and to organize them in a manner that similar clusters can be mapped to similar colors. In general, we do not know the number of distinct clusters. To address this issue, we over-define the number of possible clusters using a large number (typically 256) prototype vectors. Because of the organization in the latent space, prototype vectors that clump together will be represented by nearly identical colors. Using a crossplot tool, the interpreter can draw polygons around clumped clusters to construct a single seismic facies.

PVs are also called "SOM units". The PVs are initially distributed on a structured 2D hexagonal or rectangular grid defined by the first two eigenvectors of the input data. While the location of the prototype vectors are allowed to move within the 2D latent space, defining a 2D manifold in I-dimensional attribute space, the relative location of each PVs to its neighbors is preserved.

Let's consider a 2D SOM represented by P prototype vectors $\mathbf{m}_p = (m_{p1}, m_{p2}, \ldots, m_{pN})$, where $p=1, 2, \ldots, P$ that represent the N is the dimension of the input data (the number of samples in waveform classification). After initialization, the distance of each input vector $\mathbf{x}_j$ is computed to each of the P prototype vectors. The nearest prototype vector (the "best matching" PV) will be updated to better represent the location of $\mathbf{x}_j$ as part of SOM neighborhood training.

Given the previous background, Kohonen (2001) defines the SOM training algorithm using the following five steps:

Step 1: Consider input vector $\mathbf{x}_j$, which is randomly chosen from the set of input vectors.

Step 2: Compute the Euclidean distance between $\mathbf{x}_j$ and each PV $\mathbf{m}_p$, p=1, 2,…,P. The prototype vector, $\mathbf{m}_b$, that exhibits the minimum distance to the input vector $\mathbf{x}_j$ is called the best matching unit:

$$\left\|\mathbf{x}_j - \mathbf{m}_b\right\| = \underset{p}{\mathrm{MIN}}\left(\left\|\mathbf{x}_j - \mathbf{m}_p\right\|\right) \quad (1)$$

Step 3: At each iteration, t, update the best matching unit prototype vector and neighbors that fall within a radius $\sigma(t)$. The updating rule for the weight of the p^{th} PV inside and outside this neighborhood radius is given by

$$\mathbf{m}_p(t+1) = \begin{cases} \mathbf{m}_p(t) + \alpha(t)\, h_{bj}(t)\left[\mathbf{x}_j - \mathbf{m}_p(t)\right] & \text{if } \left\|\mathbf{r}_p - \mathbf{r}_b\right\| \le \sigma(t) \\ \mathbf{m}_p(t) & \text{if } \left\|\mathbf{r}_p - \mathbf{r}_b\right\| > \sigma(t) \end{cases} \quad (2)$$

where the neighborhood radius defined as $\sigma(t)$ is predefined for a problem and decreases with each iteration t. $\mathbf{r}_b$ and $\mathbf{r}_p$ are the position vectors of the best-matching unit PV $\mathbf{m}_b$ and the p^{th} PV $\mathbf{m}_p$. We define the "neighborhood function" $h_{bp}(t)$, the "exponential learning function" $\alpha(t)$, and the number of iterations or "length of training" T. $h_{bp}(t)$ and $\alpha(t)$ decrease with each iteration in the learning process as

$$h_{bj}(t) = \exp\left(-\frac{\left\|\mathbf{r}_p - \mathbf{r}_b\right\|^2}{2\sigma^2(t)}\right), \text{ and} \quad (3)$$

$$\alpha(t) = \alpha(0)\left(\frac{0.005}{\alpha(0)}\right)^{t/T}. \quad (4)$$

Step 4: Iterate through each learning step (steps 1-3) until the convergence criterion (which depends on the predefined lowest neighborhood radius and the minimum distance between the PVs in the latent space) is reached.

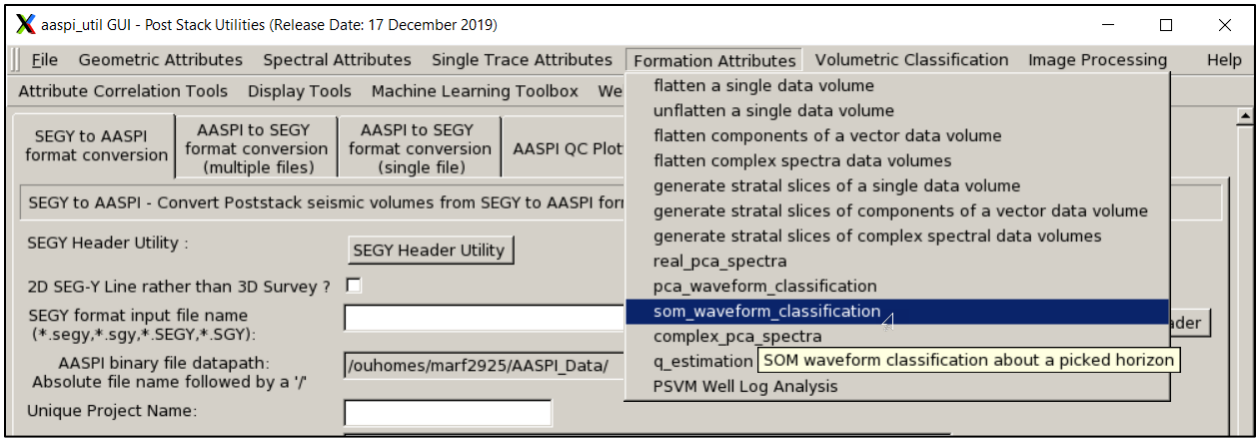
Step 5: Color-code the trained PVs as they are projected onto the 2D latent space (u_1, u_2) using a 2D color bar (Matos et al., 2009) defined by hue, H, and saturation, S:

$$H_p = \mathrm{ATAN}\left[\frac{u_{2p} - \underset{q}{\mathrm{MEAN}}\left(u_{2q}\right)}{u_{1p} - \underset{q}{\mathrm{MEAN}}\left(u_{1q}\right)}\right], \text{ and} \quad (5)$$

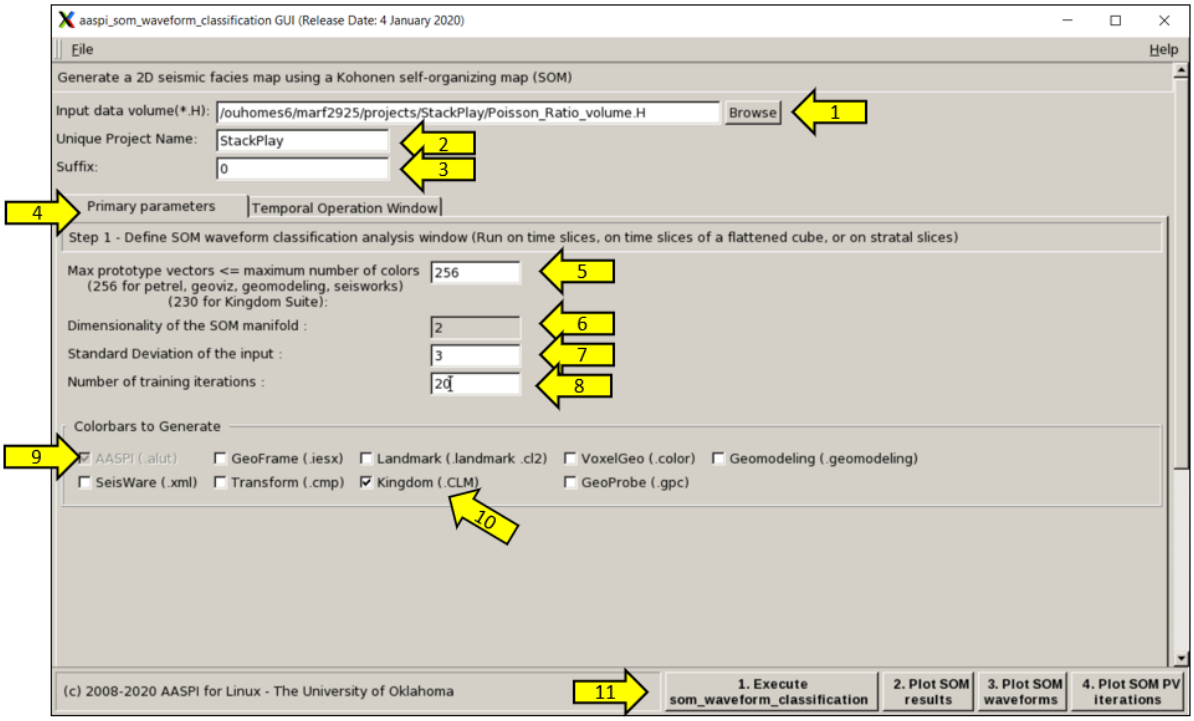
$$S_p = \left\{\left[u_{1p} - \underset{q}{\mathrm{MEAN}}\left(u_{1q}\right)\right]^2 + \left[u_{2p} - \underset{q}{\mathrm{MEAN}}\left(u_{2q}\right)\right]^2\right\}^{1/2}. \quad (6)$$

Running program som_waveform_classification and plotting the results

Program **som_waveform_classification** is launched from the *Formation Attributes* in the main **aaspi_util** GUI:



The following window will appear:



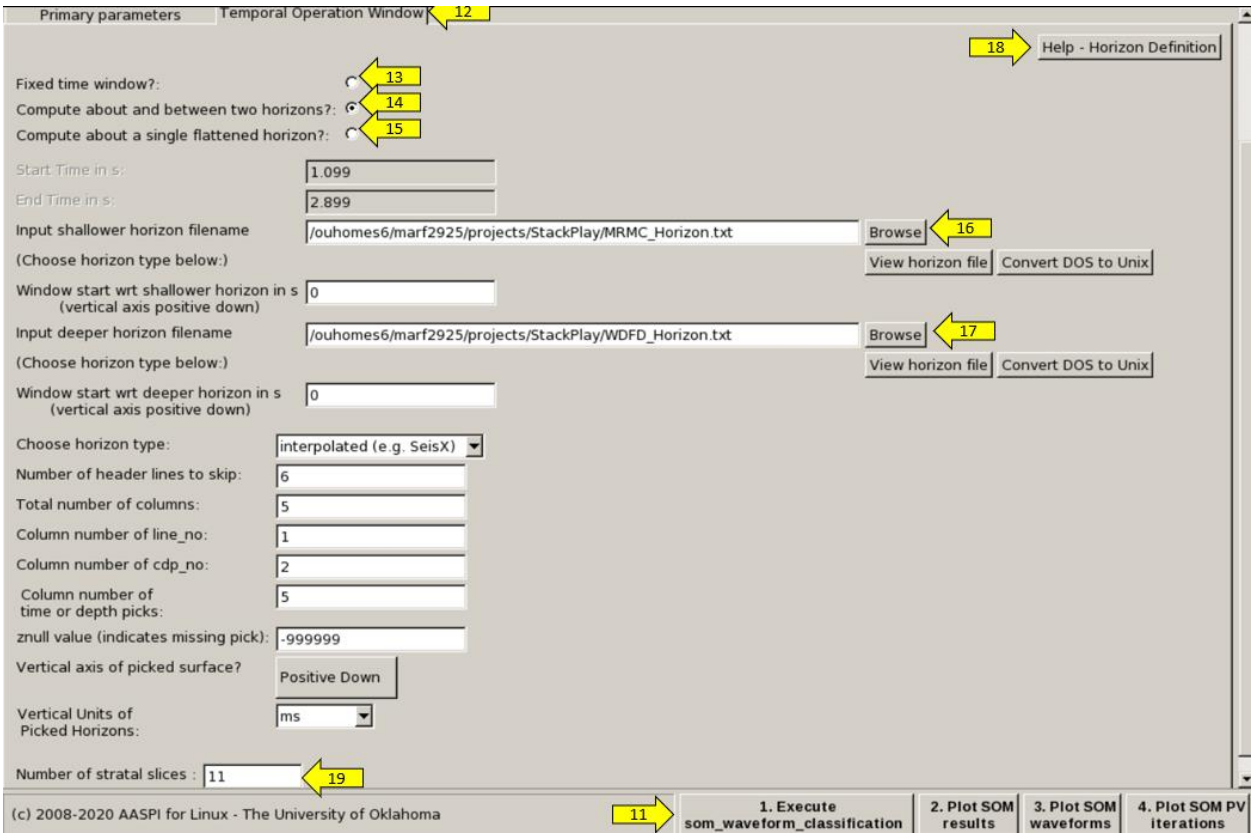
The *Primary parameters* tab

As with most AASPI programs, we enter (1) an input file name, (2) a unique project name, and (3) a suffix, where the latter option allows us to compare runs with different choices of parameters. There are two tabs, the first of which is (4) the *Primary parameters* tab. The maximum number of colors used in most workstations is 256, with Kingdom Suite only allowing 240. For this reason, the default number of prototype vectors is 256. Because the prototype vectors span the original planar manifold and 2D latent space at equal intervals, the program will use the maximum

number of prototype vectors that does not exceed this value. The (6) dimensionality of the manifold is hardcoded to be 2. In earlier work by Roy and Marfurt (2010) and Matos et al. (2009) we evaluated a 3D latent space plotted against an RGB color bar but saw little advantage in doing so. The input data are subjected to principal component analysis, where the (7) standard deviations along axes 1 and 2 are the square roots of the eigenvalues λ_1 and λ_2. Three standard deviations represent 99.7% of the data if the scaled input data can be represented by a normal distribution. The input data are decimated and then presented in a random order for each iteration in the training where in this example we have chosen (8) 20 to be the maximum number of iterations. We have adopted the Petrel color bar **.alut* format for display in the AASPI software; for this reason, this option (9) is always chosen. In this example, we have also (10) placed a checkmark in front of the Kingdom Suite option, thereby generating a **.CLM* file that we can load for display in that interpretation package. Before going to the second tab, note the (11) four steps in the computation. The first step performs the classification. The second step generates the corresponding colorbar for the last iteration and is required to properly display the results in either the AASPI or commercial software. The third and fourth steps are optional and provide some insight into how the SOM algorithm performs.

The *Temporal operation window* tab

Program **som_waveform_classification** provides a formation by formation classification where the *N* attributes are the *N* samples of the seismic trace extracted with the target area. There are three options on defining the operation window which are found under the (12) *Temporal operation window* tab:



By default the samples are (13) extracted within a fixed time window. For waveform classification,

The second option is to (14) analyze the waveforms between two picked horizons. This option will invoke program **stratal_slice** in a subsequent python script, proportionally constructing a suite of slices between the two horizons. In this example the (16) upper horizon is the Meramec, and (17) the lower horizon the Woodford. Details on the defining horizons can be found by clicking the (18) *Help – Horizon Definition* tab. If the two horizon option is chosen, the user will be prompted to (19) define the number of stratal slices used. This option is the method of choice when looking for patterns in Poisson's ratio, $\lambda\rho$, $\mu\rho$, Z_P, Z_S, or other geomechanical parameters. Finally, we may choose to (15) the analysis window about a single picked horizon. When this option is chosen the subsequent python script will invoke program **flatten** before classifying the waveforms. Because the seismic waveform is a function of the seismic source wavelet as well as of the reflectivity pattern, this option should be chosen if we wish to classify a formation using a seismic amplitude volume as input. Note that such an analysis may not produce the desired results for formations that are not approximately constant thickness.

Step 1: Execute program som_waveform_classification

With these parameters chosen, we can return to (11) Step 1, and *Execute som_waveform_classification*. In my example the first few lines that appear on my screen (the

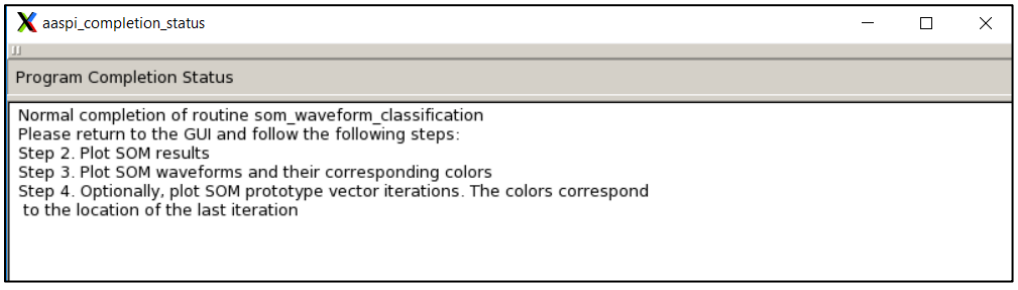
xterm window in Linux, the black AASPI window in Windows) shows the execution of program **stratal_slice**:

```
[stratal_slice]
[cdp_col=2]
[horizon_type=interpolated]
[input_fn=/ouhomes6/marf2925/projects/StackPlay/Poisson_Ratio_volume.H]
[line_col=1]
[lower_horizon_fn=/ouhomes6/marf2925/projects/StackPlay/WDFD_Horizon.txt]
[ncol=5]
[nskip=6]
[relative_t_end=0]
[relative_t_start=0]
[stratal_slice_fn=stratal_slice_StackPlay_0.H]
[suffix=0]
[time_col=5]
[unique_project_name=StackPlay]
[upper_horizon_fn=/ouhomes6/marf2925/projects/StackPlay/MRMC_Horizon.txt]
[vertical_axis_down=y]
[vertical_horizon_units_scale_factor=0.001]
[znull=-999999]



 ========================================================================
 | PROGRAM: stratal_slice                                               |
 | COPYRIGHT 2020 ATTRIBUTE-ASSISTED SEISMIC PROCESSING AND INTERPRETATION |
 | THE UNIVERSITY OF OKLAHOMA                                           |
 | ROYALTY FREE USE TO AASPI SPONSORS AND COINVESTIGATORS               |
 | SOFTWARE DISTRIBUTION TO OTHERS PROHIBITED WITHOUT COMMERCIALIZATION  |
 | AGREEMENT                                                            |
 ========================================================================
software release date = 17 December 2019

run time date = 20200106
                     month      year
       current date      1      2020
    expiration date      3      2099

   key name   key value
       trid           5
      cdp_x          16
      cdp_y          17
     scalco           8
     cdp_no          19
    line_no          18
       mute          13
       muts          12
ns_key         25  n_keys_in         24  n_keys_out         25
live_key          5
                nt       901
             nt_fft       945
             nslice        11
                dt     0.002 %
     t_first_sample     1.099 %
      t_last_sample     2.899 %
geometry successfully read in
```

followed by the program som_waveform_classification:

```
[som_waveform_classification]
[eigenvalue_fn=waveform_eigenvalue_StackPlay_0.H]
[eigenvector_fn=waveform_eigenvector_StackPlay_0.H]
[error_fn=som_waveform_classification_StackPlay_0.err]
[geomodeling_out=n]
[geoprobe_out=n]
[iesx_out=n]
[input_fn=stratal_slice_StackPlay_0.H]
[landmark_out=n]
[n_eigen=2]
[niteration=20]
[np_max=256]
[nstd=3]
[prototype_vector_color_matrix_fn=prototype_vector_color_matrix_StackPlay_0.H]
[scaled_prototype_vector_waveform_matrix_fn=scaled_prototype_vector_waveform_StackPlay_0.H]
[seisware_out=n]
[smt_out=y]
[som_color_table_fn=som_waveform_classification_colors_StackPlay_0]
[som_colorbar_fn=som_waveform_classification_colors_StackPlay_0.alut]
[som_projection_axis_1_fn=som_waveform_classification_axis1_StackPlay_0.H]
[som_projection_axis_2_fn=som_waveform_classification_axis2_StackPlay_0.H]
[som_waveform_classification_fn=som_waveform_classification_StackPlay_0.H]
[som_waveforms_projected_on_latent_space_fn=som_waveforms_projected_on_latent_space_StackPlay_0.H]
[suffix=0]
[t_end=0.01]
[t_start=0]
[transform_out=n]
[unique_project_name=StackPlay]
[unscaled_prototype_vector_waveform_matrix_fn=unscaled_prototype_vector_waveform_StackPlay_0.H]
[voxelgeo_out=n]



 ========================================================================
 | PROGRAM: som_waveform_classification                                 |
 | COPYRIGHT 2020 ATTRIBUTE-ASSISTED SEISMIC PROCESSING AND INTERPRETATION |
 | THE UNIVERSITY OF OKLAHOMA                                           |
 | ROYALTY FREE USE TO AASPI SPONSORS AND COINVESTIGATORS               |
 | SOFTWARE DISTRIBUTION TO OTHERS PROHIBITED WITHOUT COMMERCIALIZATION  |
 ! AGREEMENT                                                            |
 ========================================================================
 software release date = 17 December 2019

 run time date = 20200106
                       month      year
       current date        1      2020
    expiration date        3      2099

 first_cdp,last_cdp,n_cdp_in         1031       2653       1623
 first_line,last_line,n_line_in         2537       4106       1570
             t_start              0.000
               t_end            100.000
               t_min              0.000
               t_max            100.000
                  d1             10.000
```

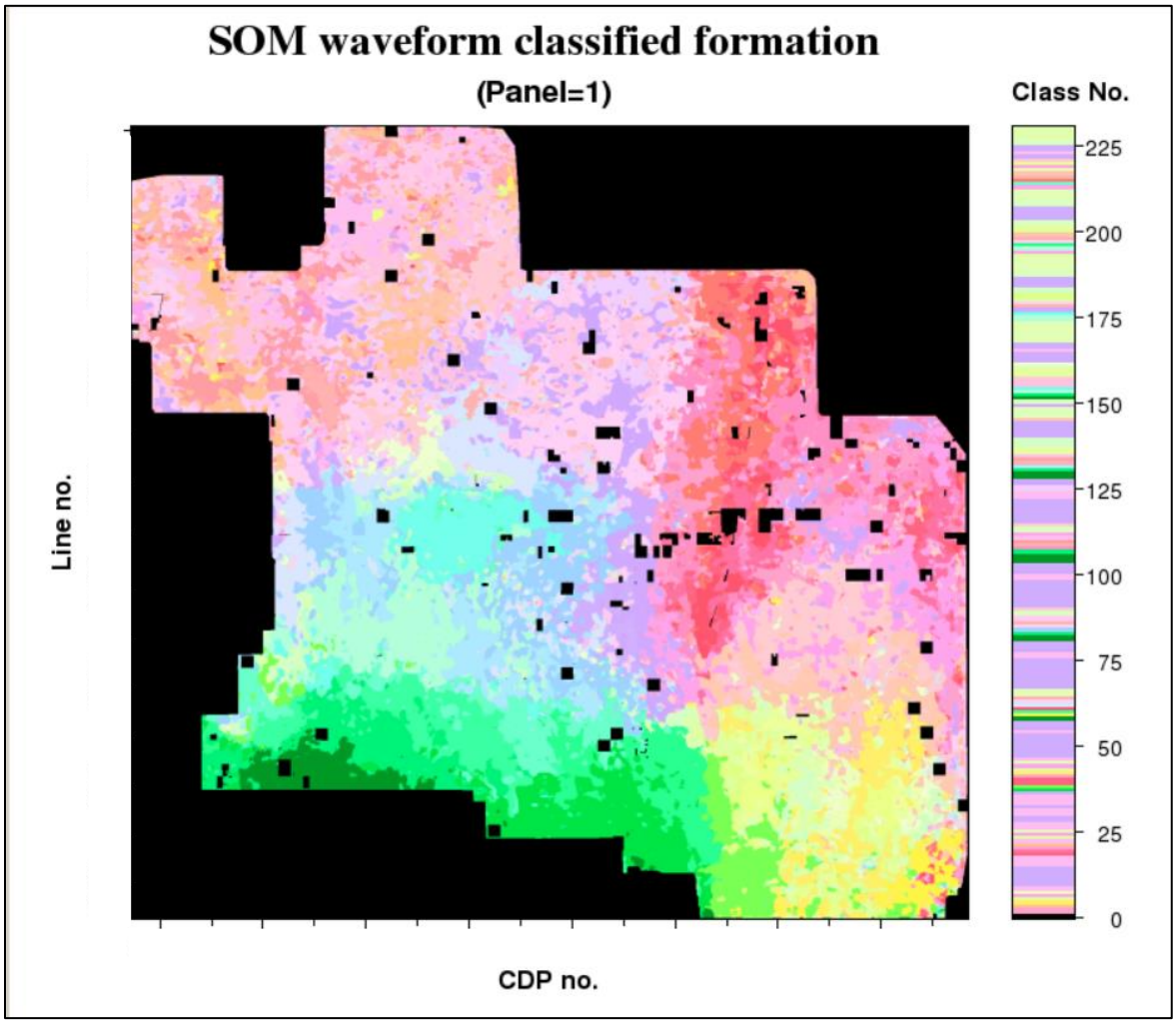
When complete, a pop-up window will appear reporting the status of the program. In the case or normal completion, the panel will look like this:



Step 2: Plotting the classification results

Returning to the GUI, we invoke (11) *Step 2. Plot SOM results*. Here, the GUI invokes the python script *aaspi_aaspiviewer_poststack.py* that we commonly use to quality control most AASPI results. The multiplexed 2D colorbar generated in the previous step maps the distribution of the prototype vectors as they appear in the latent space at the final iteration. Because there is only one value for each trace in the windowed formation, the data file

som_waveform_classification_unique_project_name_suffix.H is only one sample thick. The python script "slices" and transposes this file prior to plotting the results:

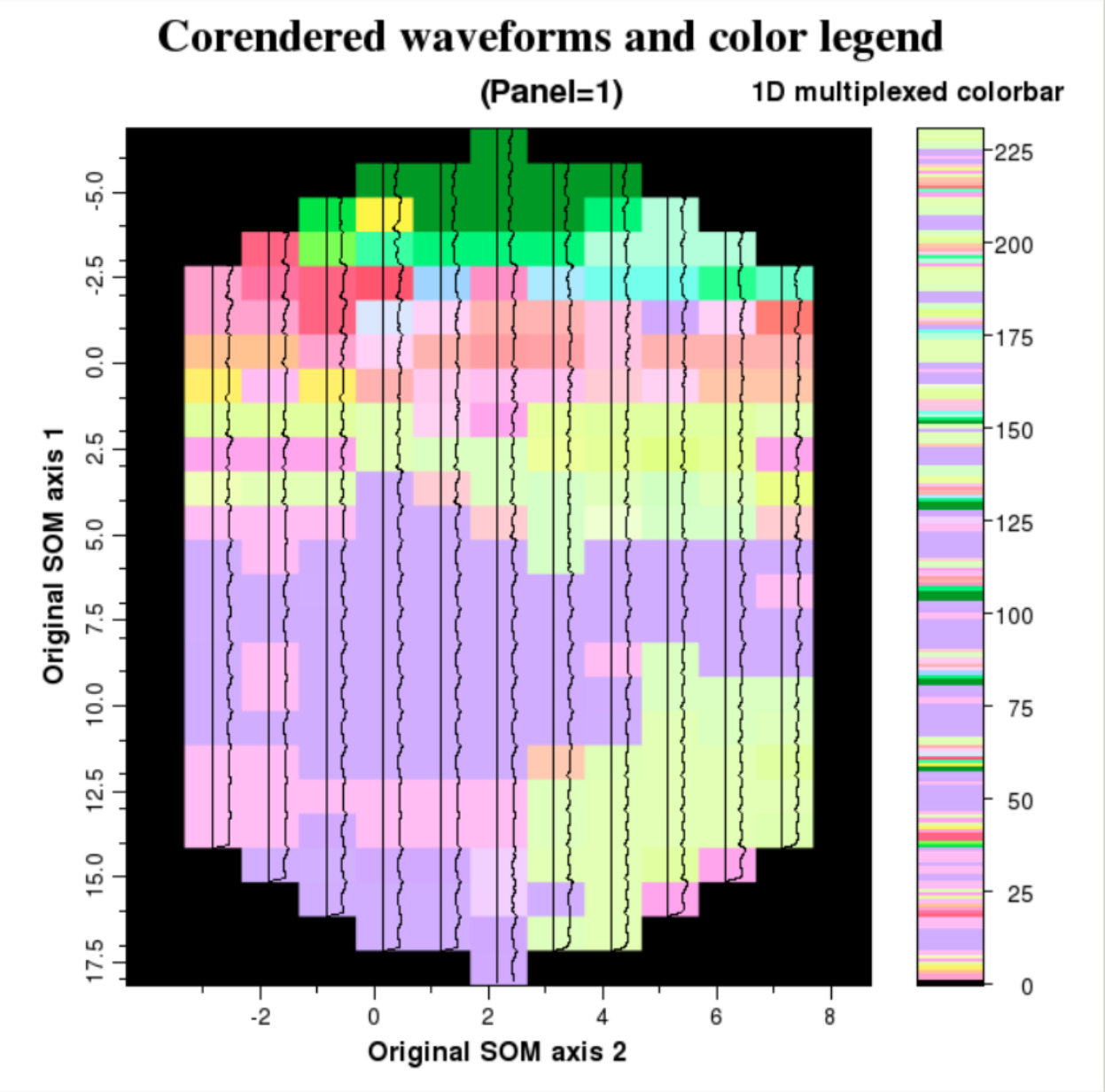


where I have whited-out the actual numbers and CDP numbers for reasons of data confidentiality. Although a total of 231 classes were used, most interpreters may see only ten or so distinct colors, indicating that most of the clusters have clumped together into a smaller subset.

Step 3. Plotting the prototype vectors against their color at the last iteration

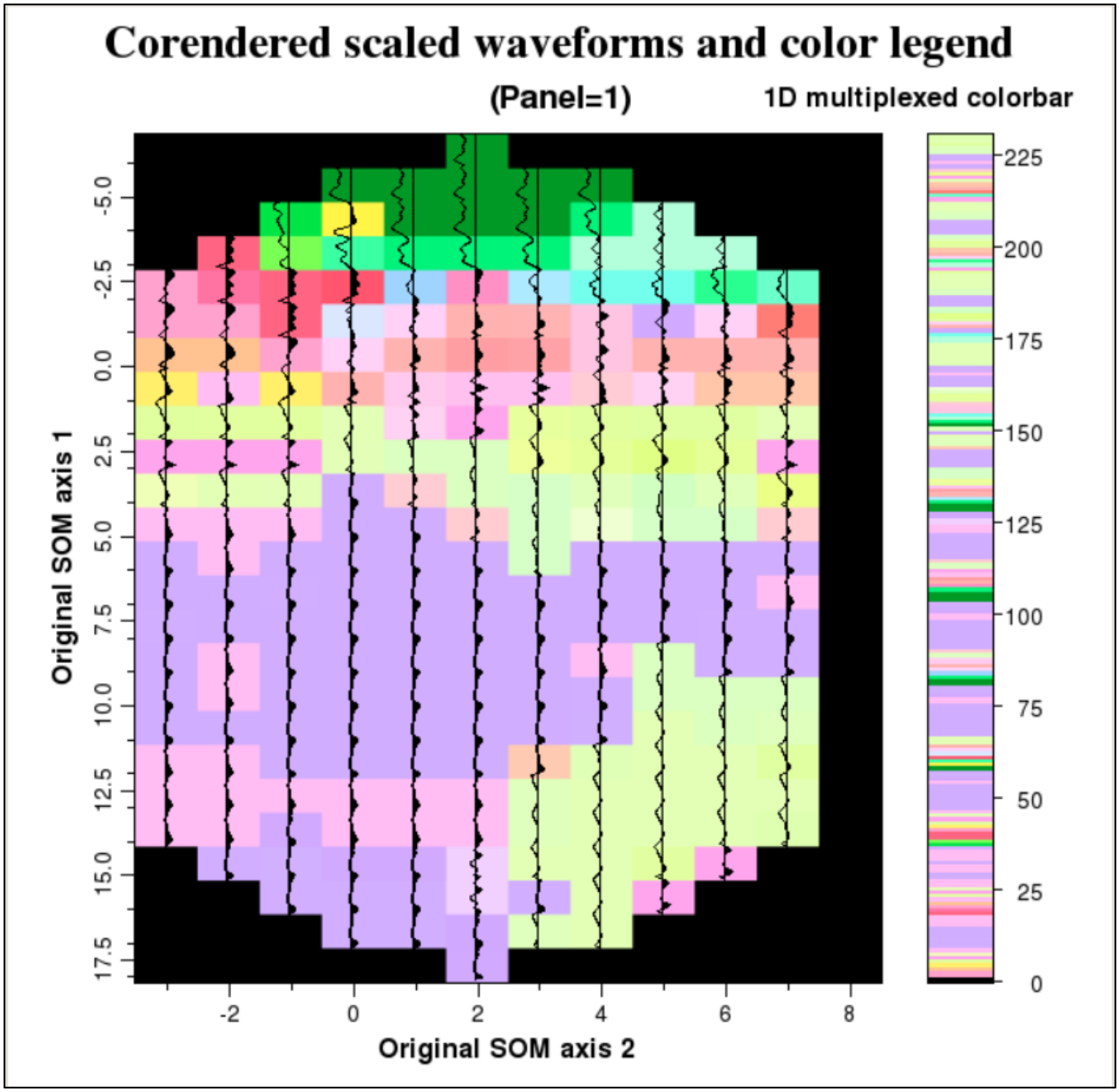
The **som_waveform_classification** program also outputs the prototype vectors which can be corendered with the 2D colorbar, giving a visualization of the relation between prototype vectors and facies colors. The two files are named as *prototype_vector_color_matrix_unique_project_name_suffix.H* and

prototype_vector_waveforms_unique_project_name_suffix.H. Clicking (11) *Step 3. Plot waveforms* in the GUI invokes the python script aaspi_corender.py to corender the two files:



where in this example the unscaled Poisson's ratio wavelet is plotted against a rectangular color background. Note that there are no longer 231 distinct colors.

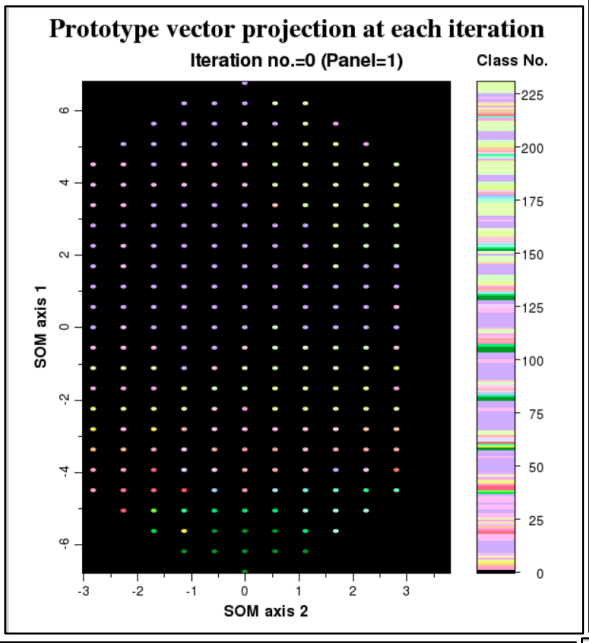
Internal to the program, the classification is actually applied to the scaled data, which therefore generate scaled waveforms that even though they are for Poisson's ratio, now have both positive and negative values:

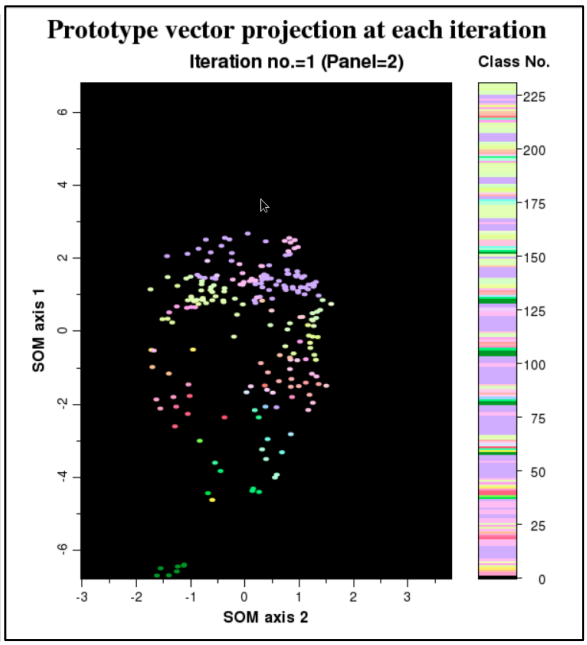


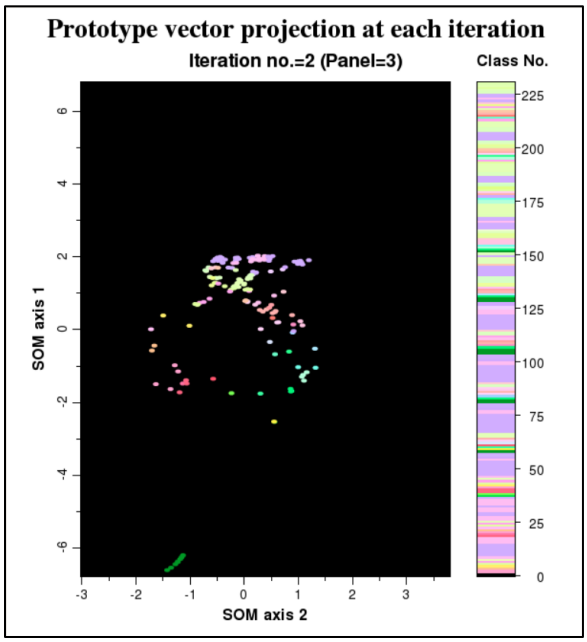
Step 4. Plotting the location of the prototype vectors in the latent space at each iteration

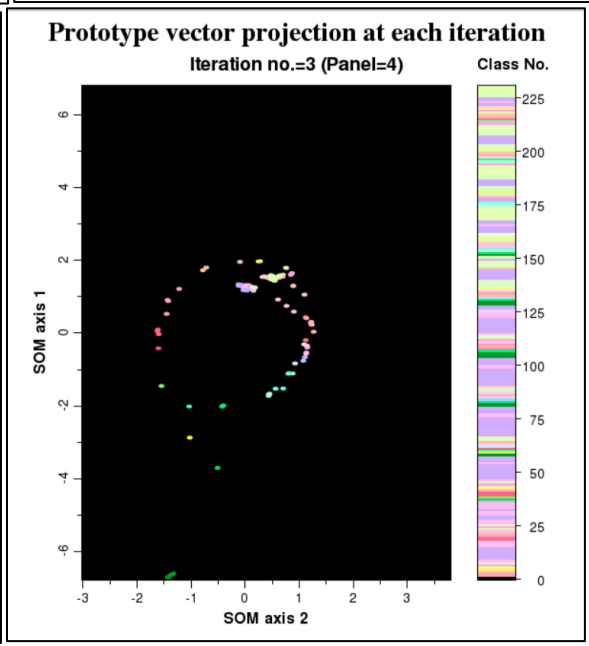
To gain some insight into the inner workings of the **som_waveform_classification** program we can plot the location of the prototype vectors projected onto the latent space against SOM axes 1 and 2 for each iteration. The zeroth iteration (the program initialization) consists of equally spaced prototype vectors distributed on an ellipse whose axes are the first two eigenvectors and whose ranges, ±3σ, were defined as an input parameter. Returning to the GUI, we (11) click *Step*

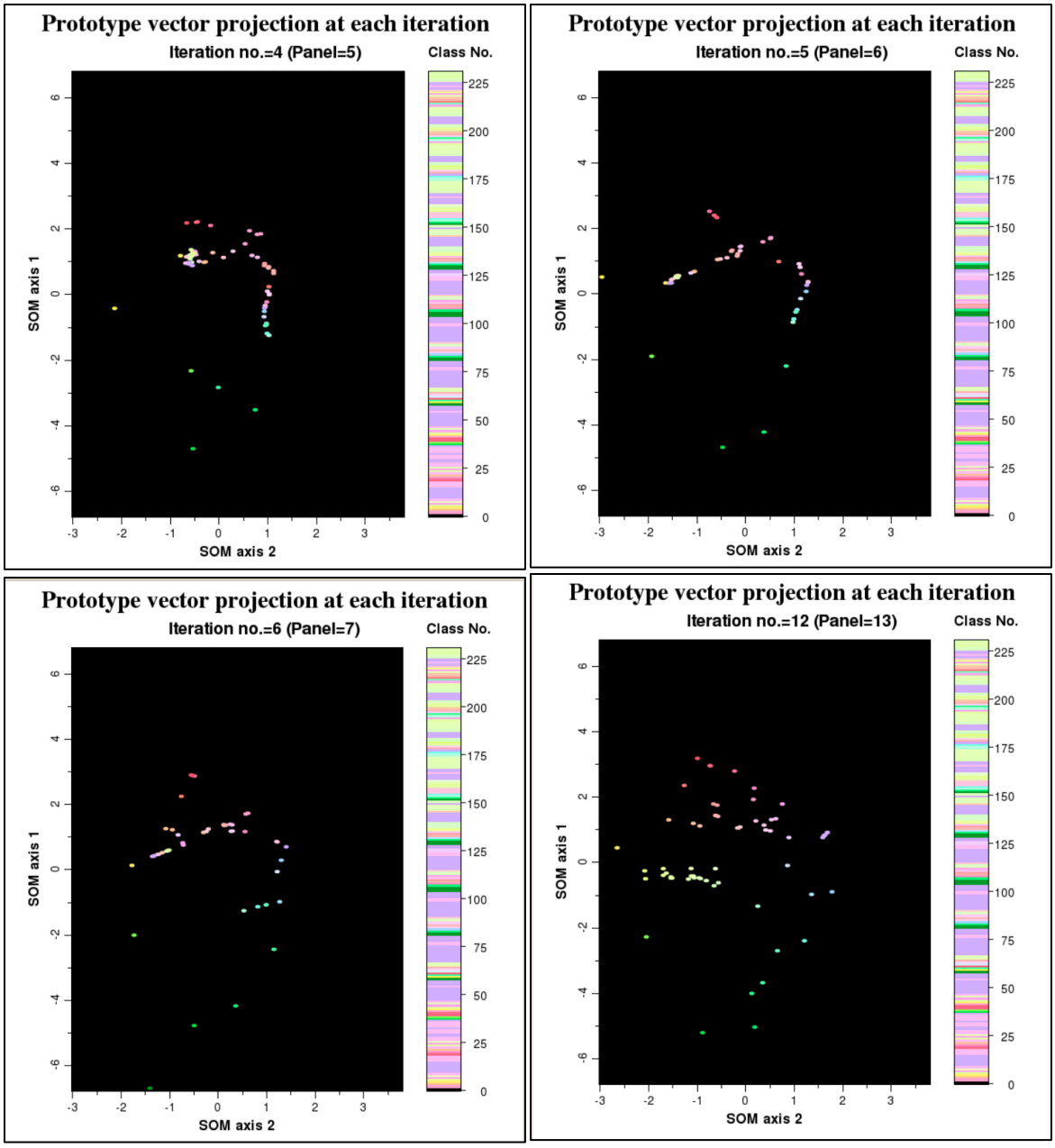
4. Plot SOM PV iterations after which the GUI plots the file som_waveforms_projected_on_latent_space_*unique_project_name_suffix*.H by invoking the python script *aaspi_aaspiviewer_poststack.py*. A subset of the images looks like this:







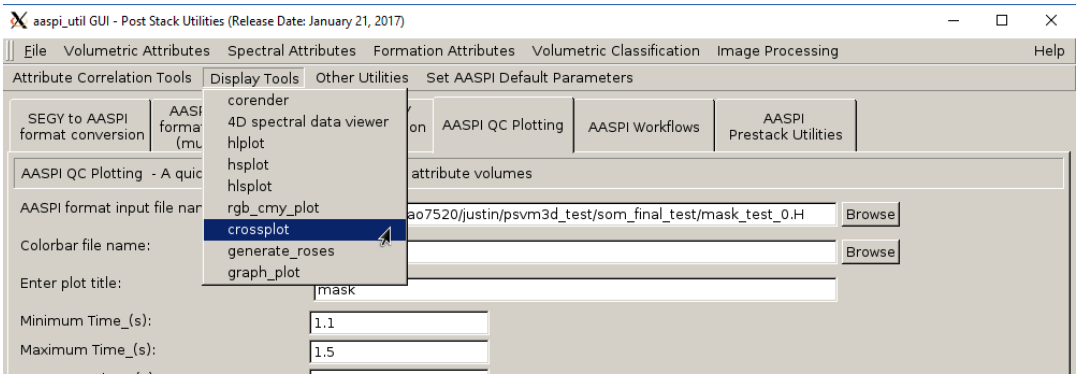




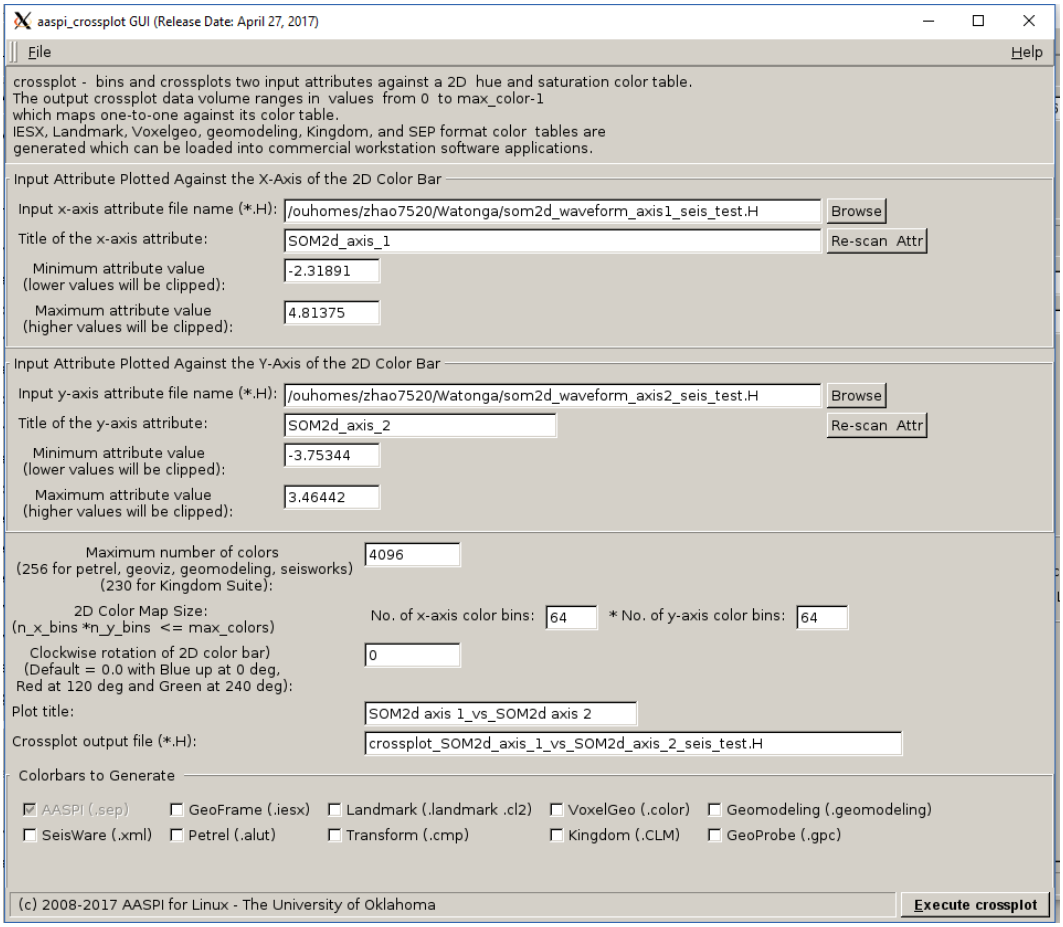
Note that at iteration 0, the prototype vectors are equally distributed across an ellipse. A great deal of reorganization takes place in the first two or three iterations. By iteration 12 there are no more changes. Ideally, each iteration should have its own colorbar, but this would require multiple files that would be more difficult to animate. Instead, the colorbar used in this display for each prototype vector correspond to their final location at iteration 12.

A more flexible display option in interpretation workstations: Crossplotting the results

The user can use **crossplot** module in the **aaspi_util** to crossplot two SOM axes in order to generate the SOM facies map with a 2D color map. The **crossplot** module can be found under the *Display Tools* tab in the **aaspi_util** GUI:

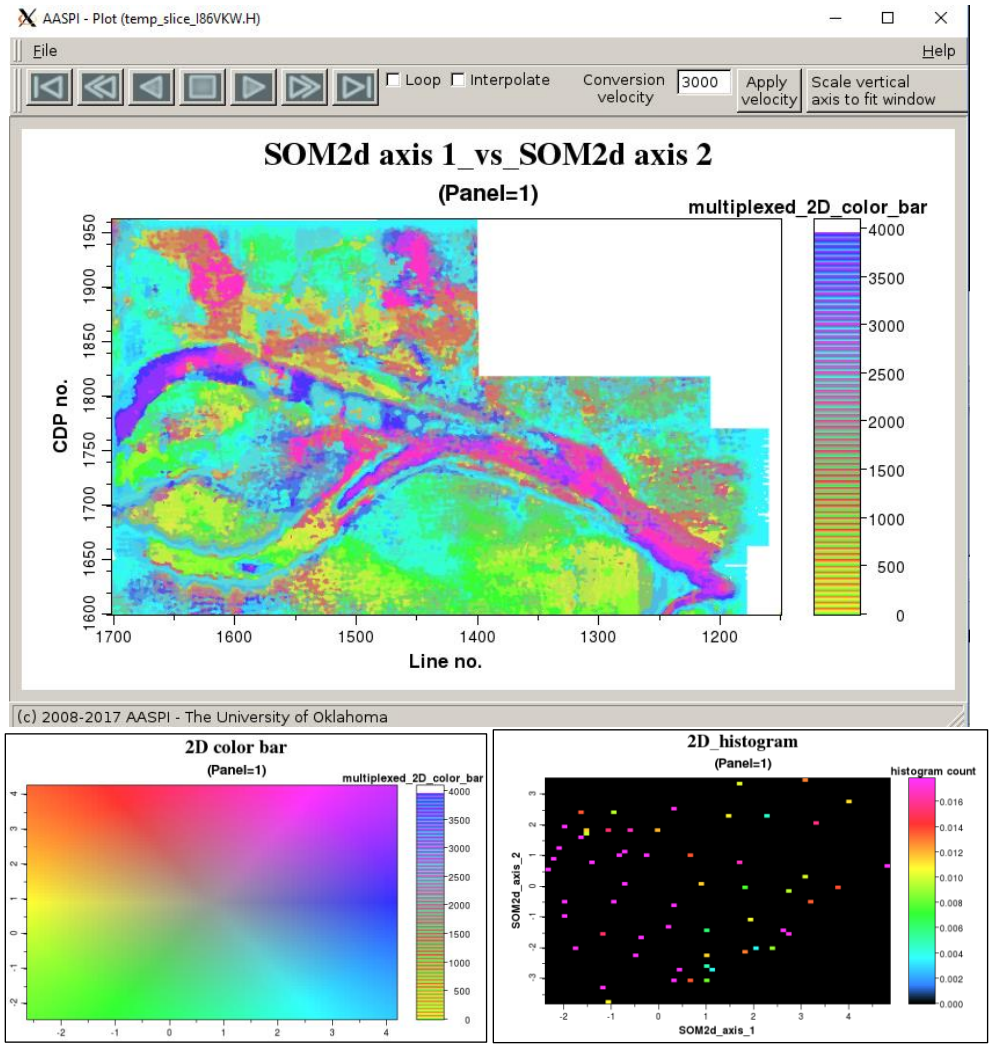


The crossplot GUI is shown below:



The input for **crossplot** are som_waveform_classification_axis1_ *unique_project_name_suffix*.H and som_waveform_classification_axis2_ *unique_project_name_suffix*.H. SOM axes 1 and 2 are taken as inputs for x and y axes in the crossplot. To ensure a smooth color transition, 4096 colors

are used for the 2D colorbar to be generated (64 by 64 colors). The result is shown below (computed from flattened seismic amplitude data about the top Red Fork formation):



Program **crossplot** generated a crossplotted volume, a 2D color map, and a 2D histogram of the crossplotted volume. The 2D histogram shows clusters of facies, where these clusters are color-coded by the color at the corresponding position in the 2D color map. In this example, we observe the different stages of the channels, as well as the flood plain deposits.

Visualization by crossplotting two SOM axes in Petrel

Please refer to the documentation of **som3d for** using the crossplotting functionalities in Petrel for visualizing the **som_waveform_classification** facies map.

An example using seismic amplitude phantom horizon slices about the top Red Fork formation, Oklahoma.

References

Coléou, T., M. Poupon, and K. Azbel, 2003, Unsupervised seismic facies classification: A review and comparison of techniques and implementation: The Leading Edge, v. 22, p. 942-953.

Gao, D., 2007, Application of three-dimensional seismic texture analysis with special reference to deep-marine facies discrimination and interpretation: An example from offshore Angola, West Africa: AAPG Bulletin, v**.** 91**,** p. 1665-1683.

Kohonen, T. ,1982 Self-organized formation of topologically correct feature maps: Biological Cybernetics, v. 43 p. 59-69.

Kohonen, T., 2001, Self-organizing Maps, 3rd ed.: Springer- Verlag.

Matos, M. C., K. J. Marfurt., and P. R. S. Johann, 2009, Seismic color Self-Organizing Maps: 11th International Congress of the Brazilian Geophysical Society, Expanded Abstracts.

Matos, M. C., P. L. Osorio, and P. Johann, 2007, Unsupervised seismic facies analysis using wavelet transform and self-organizing maps: Geophysics, v. 72, p. P9-P21.

Roy, A., and K. J. Marfurt, 2010, Applying self-organizing maps of multiattributes, an example from the Red-Fork Formation, Anadarko Basin: 81st Annual International Meeting Society of Exploration Geophysicists, Expanded Abstracts, p. 1591-1595.

Strecker, U., and R. Uden, 2002, Data mining of 3D poststack attribute volumes using Kohonen self-organizing maps: The Leading Edge, v. 21, p. 1032-1037.

Wallet, C. B., M. C. Matos, , and J. T. Kwiatkowski, , 2009, Latent space modeling of seismic data: An overview, The Leading Edge, v. 28, p. 1454-1459.