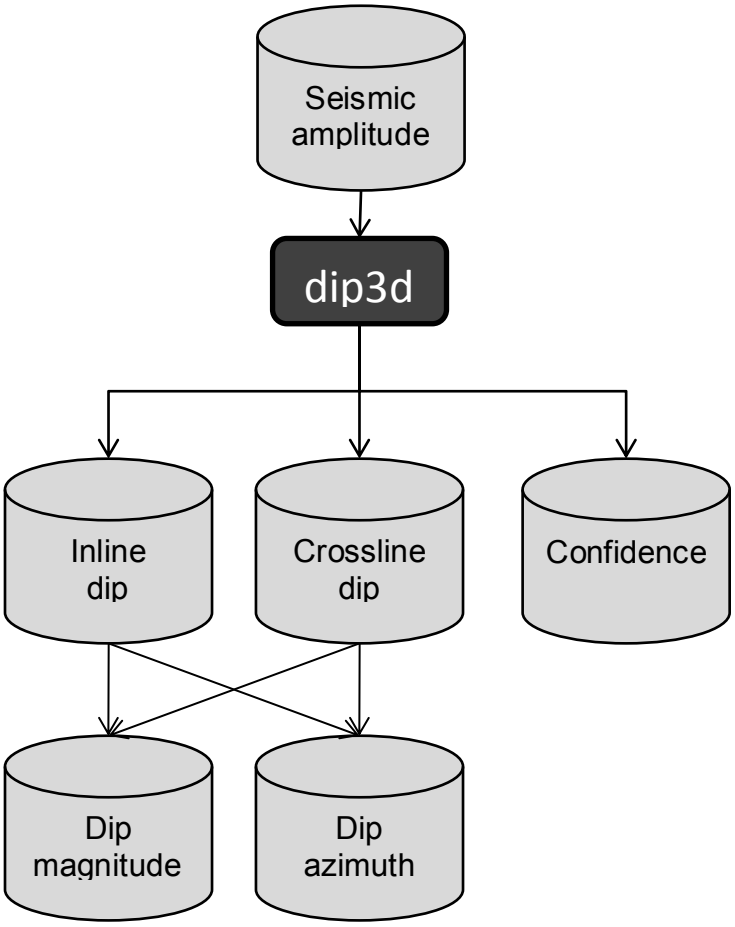
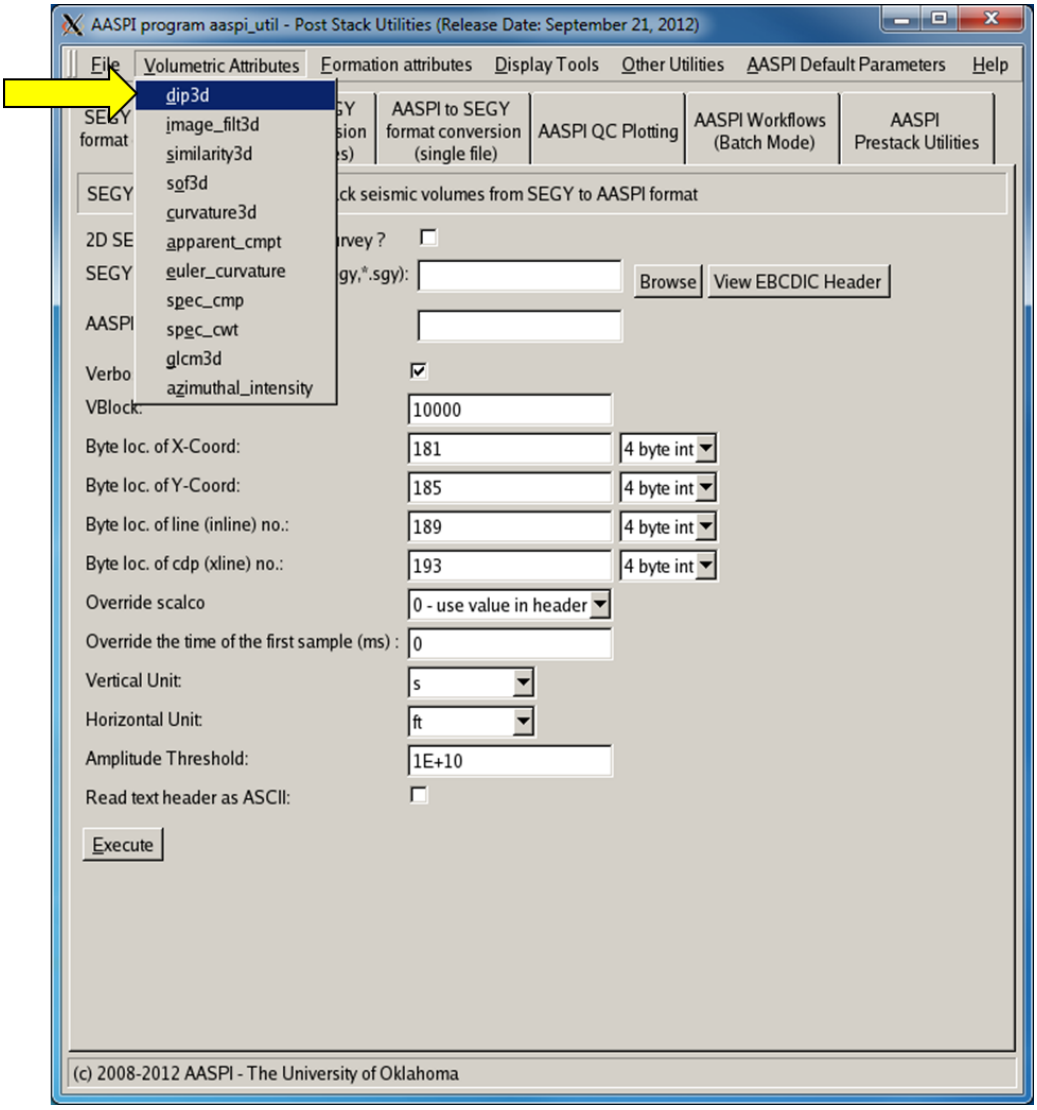
Computing Structural Dip – Program dip3d

Computation flow chart

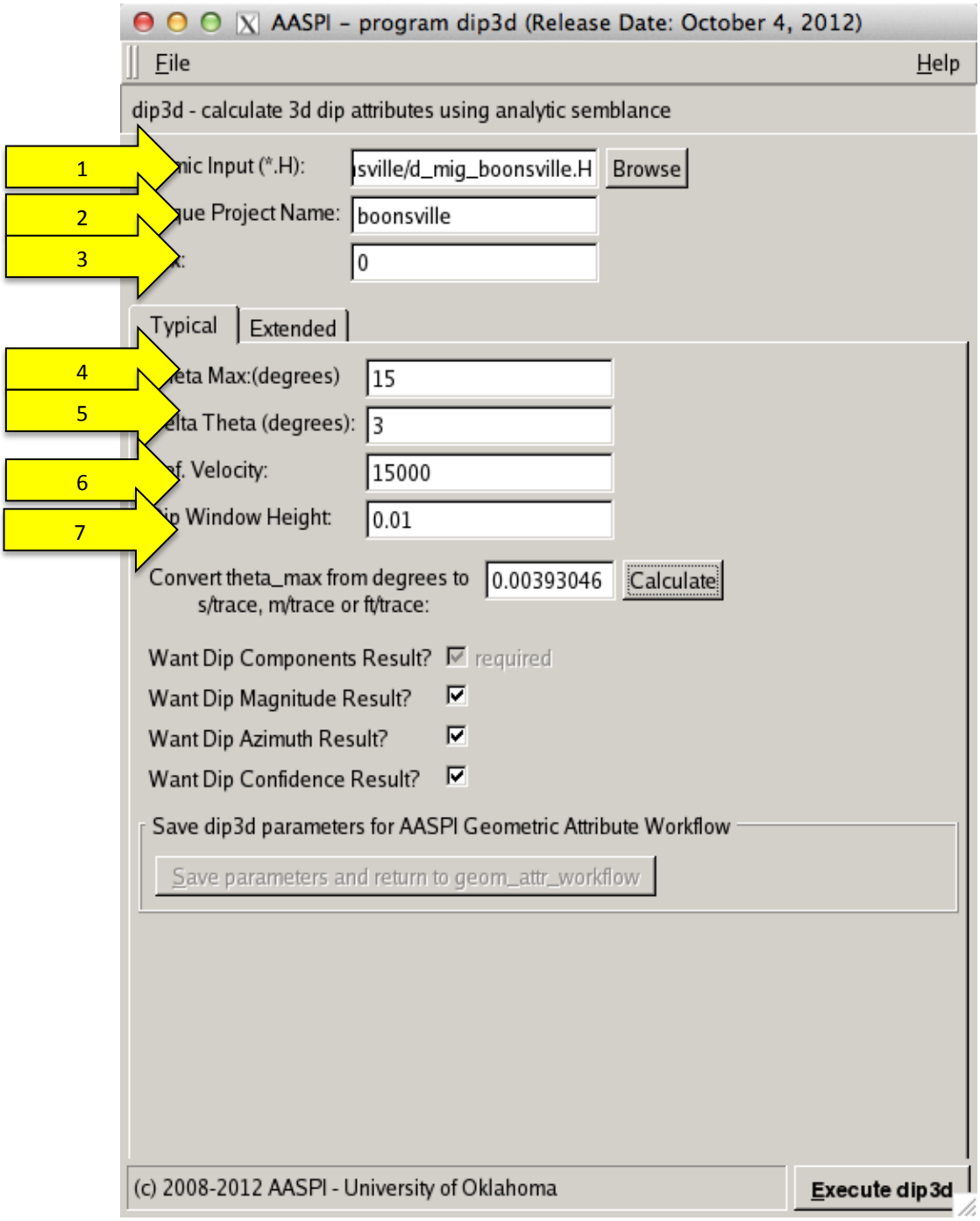
Program **dip3d** uses a seismic amplitude volume as its input and generates estimates of inline dip, crossline dip, dip magnitude, dip azimuth, and a confidence measure of these estimates. The inline and crossline dip components are critical for almost all subsequent AASPI computations. The dip magnitude and dip azimuth are useful if you wish to display dip azimuth modulated by dip magnitude using the multiattribute display program **hlplot.** The confidence volume is used to remove artifacts in the dip volumes using program **image_filt3d.** Initially, these input seismic amplitude volume will be either your time- or depth-migrated amplitude or acoustic impedance. However, as we progress through the AASPI software, we shall wish to recompute the dip components from the data that have been subjected to structure-oriented filtering using the program **sof3d**, or have been spectrally balanced using the program **spec_cmp**.



Parameter description



We are now ready to begin computing attributes. At the top of the **aaspi_util** GUI, click the 'Volumetric Attributes' tab, located to the right of the 'File' tab. A downdrop menu will appear containing the major AASPI programs – **dip3d, image_filt3d, similarity3d, sof3d, curvature3d, apparent_cmpt, euler_curvature, spec_cmp, glcm3d**, and **azimuthal_intensity**. In general, we will proceed from top to bottom, with the output of **dip3d** being required input for **similarity3d, sof3d**, **curvature3d,** and **glcm3d**. AASPI has "AASPI Workflows (Batch Mode)", with which you can run set of programs in sequence with one click, which saves time and effort (ChapterXX). The output of **sof3d** can be used for either **dip3d** or **similarity3d**, providing iterative structure-oriented filtering capabilities.

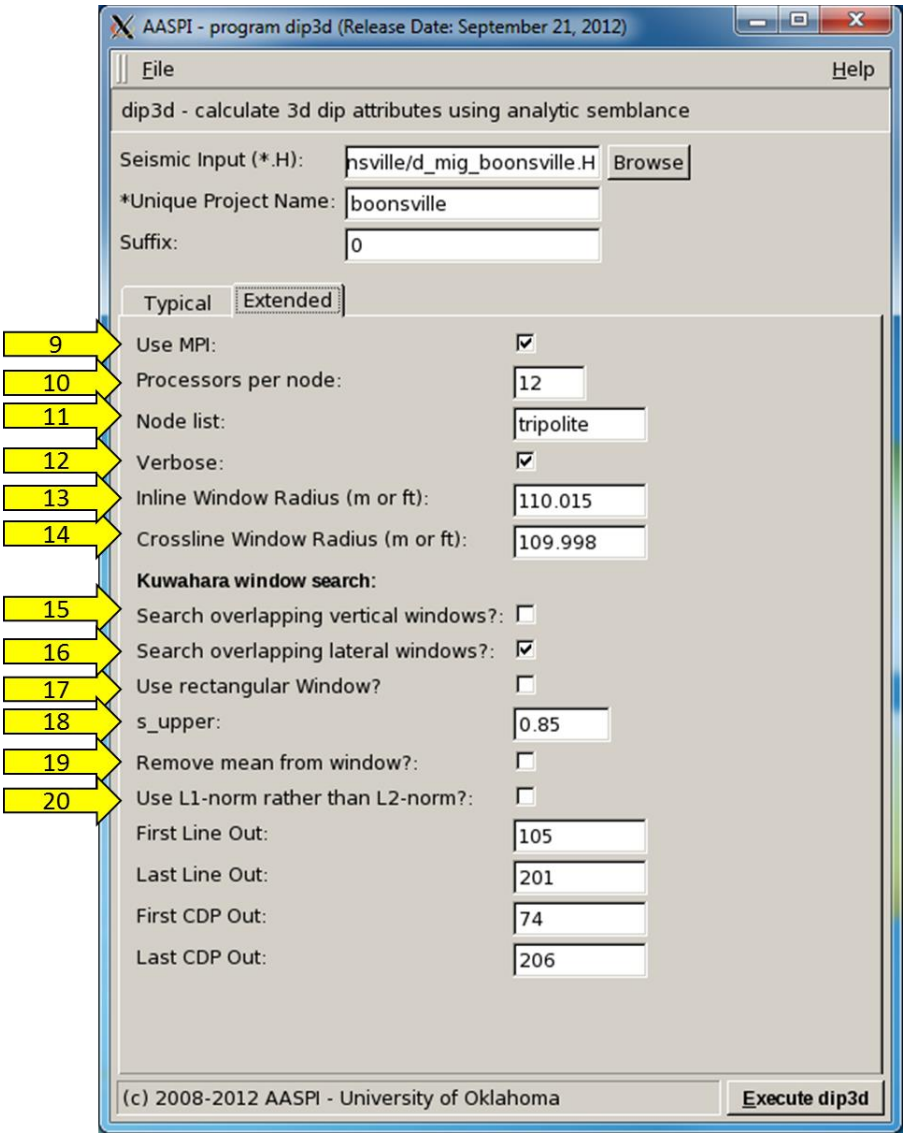


Click (1) *Browse,* and select *d_mig_boonsville.H* as your input seismic data set. You also need to (2) choose a unique project name, which will be tacked on to an attribute descriptor to organize the output. We purposely used my project name in the conversion from segy to SEP format, but this is not necessary. Type in 'boonsville'.Then (3) select a *Suffix*. If this is the original migrated data and you wish to distinguish between it and a data volume that has gone through two passes of structure-oriented filtering, you might give this run a *Suffix = 0* or *Suffix = orig*, and the 2nd run a *Suffix = pc_2*.

The default dip search will be from -25^0 to $+25^0$ at increments of 5^0. The reflectors in the Boonsville survey are very flat, rarely exceeding 5^0. Therefore, I set the value of (4) *Theta_max* = 15, and (5) *dTheta* = 3. The discrete searches will occur over inline dip components, *p*, and crossline dip components, *q*, forming a rectangular grid of discrete dips. Thus, for the values of 15 and 3 above, there will be (5+1+5)*(5+1+5) =121 discrete semblance evaluations along candidate dips. Doubling the dip search range (*Theta Max*) but retaining the same search increment (*dTheta*) would quadruple the computational effort. Note that this search is more exhaustive than that provided in commercial algorithms which limit the search to inline and crossline ((5+1+5)+(5+5)=21 discrete semblance evaluations. The more exhaustive search provides superior results for aliased data and very steep dips.

Alternatively, you might wish to compare the effect of values of (5) *dtheta*, or of window size of 20 ms and set your suffix to 'dtheta_2.5' or 'window_20ms'. Remember that you will need to use characters that can form a valid Linux file name (no = plus signs, slashes, blanks, etc.) Also, remember that the binary files are being stored in a directory defined in file ~/.datapath .Even if you run programs from different directories, files with the same name will overwrite each other in the directory defined by .*datapath*. If you wish, you can copy and modify the contents of .*datapath* in your local project directory. This local version of *datapath* will take precedence over that in your home directory.

The *reference velocity* (6) is read in from the file *aaspi_default_parameters*, which is currently set to be 15,000 ft/sec. For the Boonsville data, we may wish to use a slower velocity of about 10,000 ft/sec. The *dip window height* (7) is defaulted to be 5**dt*, which in this case is 0.002, or the sample rate (2 millisecond data). We will use this setting for our first run. Clicking the *calculate* button will tell us the time each dip calculation will take so we can estimate how long the process will run. Finally, lets put a check in the boxes for *'Want dip magnitude result?' , 'Want dip azimuth result?' , and 'Want dip confidence result?'.*



If you click the *Extended* option in the middle of the menu, you will obtain the window above. We will wish to (9) use MPI so place a checkmark in front of it. I am logged onto a machine called '*tripolite.ou.edu*' and wish to run locally so I (11) type *tripolite* as my node list. I wish to use (10) all 12 processors on *tripolite*. Nodes typically will have 1, 2, 4, 8, 16, 32, or even 64 processors in a single box. If you are running on your local computer, you simply type in *localhost*. If you were running on three machines with names of 'jade','hematite', and 'tripolite', you would type in 'jade hematite tripolite' in this line. Note the blank space delimiters! If each of these nodes has 8 processors, you would use a total of 24 processors. In principal, the maximum number of processors is the number of cdps in a given seismic line, but improvement in performance levels off for distributed processors when the computation time on any given processor approaches the network communication time to copy input data into the processor and results back to the master node. If you are fortunate to own a shared-memory, 64-processor computer, MPI simply copies the data from one memory bank to another such that the speed increases almost linearly with the number of processors.

In an earlier section on defining your computational environment, I discussed copying the following file from the AASPI installation location to your home directory by typing:

cp ~${AASPIHOME}/par/aaspi_default_parameters .

Don't forget the dot '.' at the end. My aaspi_default_parameters file on my cluster account looks like this:

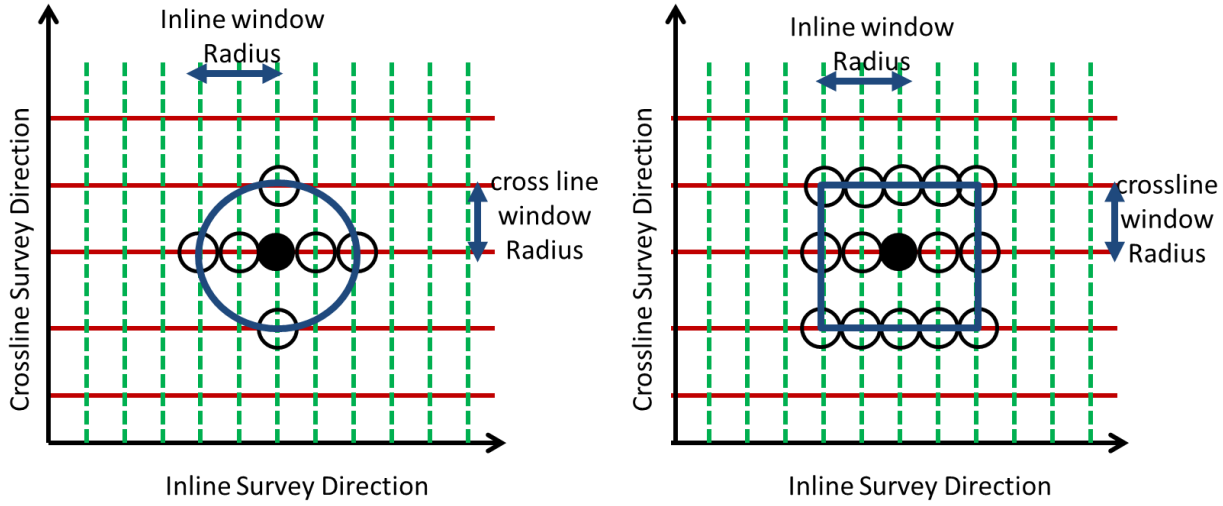
```
#-------------------------------------------------------------------------
# the following is a list of default parameters read by several of the GUIs
#-------------------------------------------------------------------------
node_list="d009 d010 d011 d012 d013 d014"
processors_per_node=8

input_line_no=189
input_cdp_no=193
input_cdp_x=181
input_cdp_y=185

output_line_no=189
output_cdp_no=193
output_cdp_x=181
output_cdp_y=185
```

where d009 , d010, …, d014 are each 8-processor nodes. The (12) *verbose* option does just that – it turns a great deal more information as the program runs allowing one to track any previous problems.

Analysis window description in AASPI software

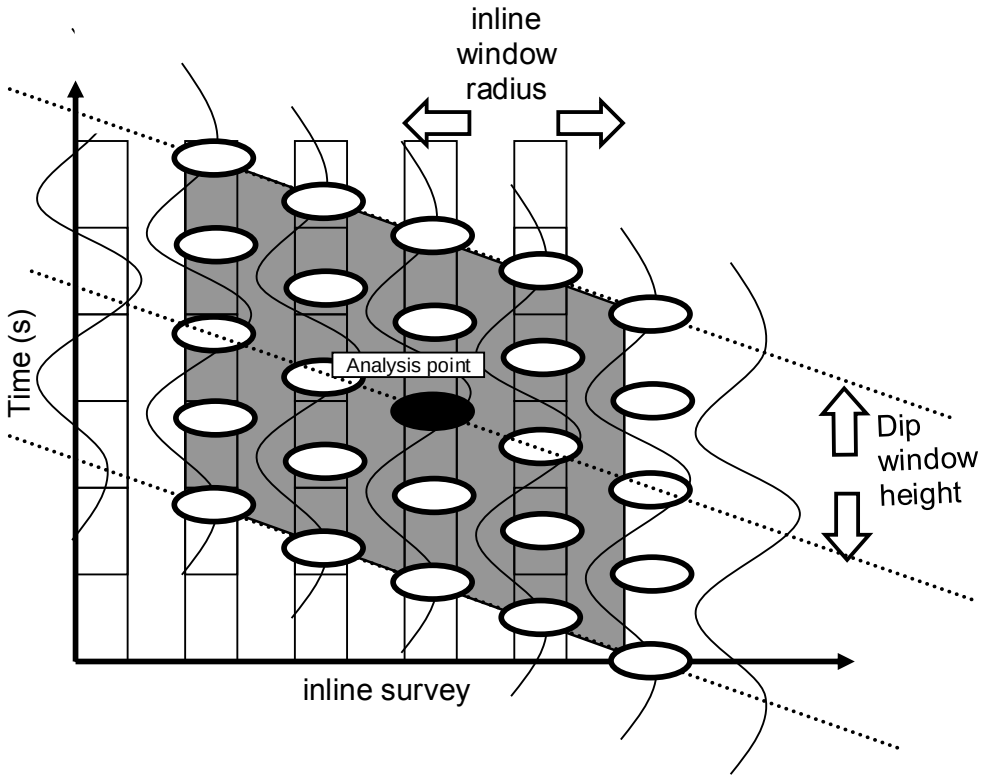
Most, but not all, of the AASPI algorithms work within a running analysis window. Typically these windows are defined by their length and width in m (or ft) and height is s (or km, kft, m, ft), and are oriented along structural dip and azimuth. The figure below shows a typical rectangular analysis window. Let's assume the trace separation is *dcdp=12.5 m* in the inline direction and *dline=25 m* in the crossline direction. If you wanted to have the same dip resolution in both directions you may choose an *inline_window_radius=2*dcdp*=25 m and *crossline_window_radius=dline*=25 m. The program default is to set the window radii to be the inline and crossline trace spacing (bin size) resulting in a 5-trace analysis circular window. Larger analysis windows result in longer run times, increased angular resolution, and decreased spatial resolution (smearing). If the data are noisy, a circular window radius equal to 2 bins is a good place to start, resulting in a 13-trace analysis window if *dcdp=dline.*

The window above corresponds to a survey with *dcdp*=12.5 m and *dline*=25 m. A circular window with inline and crossline radii of 25 m therefore contains 7 traces.



If we place a check mark in the (17) *Use rectangular window?* option in the Extended tab, we will obtain the window above which contains 15 traces.

While the vertical_window_height parameter defines the half-height of the analysis window, the window itself will always be centered along dip, as in the following image:



A good rule of thumb is to select a vertical window about the size of the dominant frequency in your data. Thus, if your dominant frequency is 20 Hz, the dominant period is 0.050 s, suggesting a half-window height of 0.025 s. If your dominant frequency is about 50 Hz, giving a period of 0.020 s, you may wish to use a half-window height would be 0.010 s. The default is to set the half window height to be 5**dt*, where dt is the seismic sample rate in *unit1* (e.g. ms, s, ft, m, kft, km). In contrast to the window radii, the window height does not significantly impact run times, but larger windows can result in vertical smearing. If your data are particularly noisy, you will want to use larger vertical analysis windows, at least until you have the opportunity to run structure-oriented filtering.

For the first run, I will choose an (13) inline_window_radius=110 ft, a (14) crossline_window_radius=110 ft, and on the previous tab (7) a vertical analysis half window of 0.010 s.

Kuwahara Windows

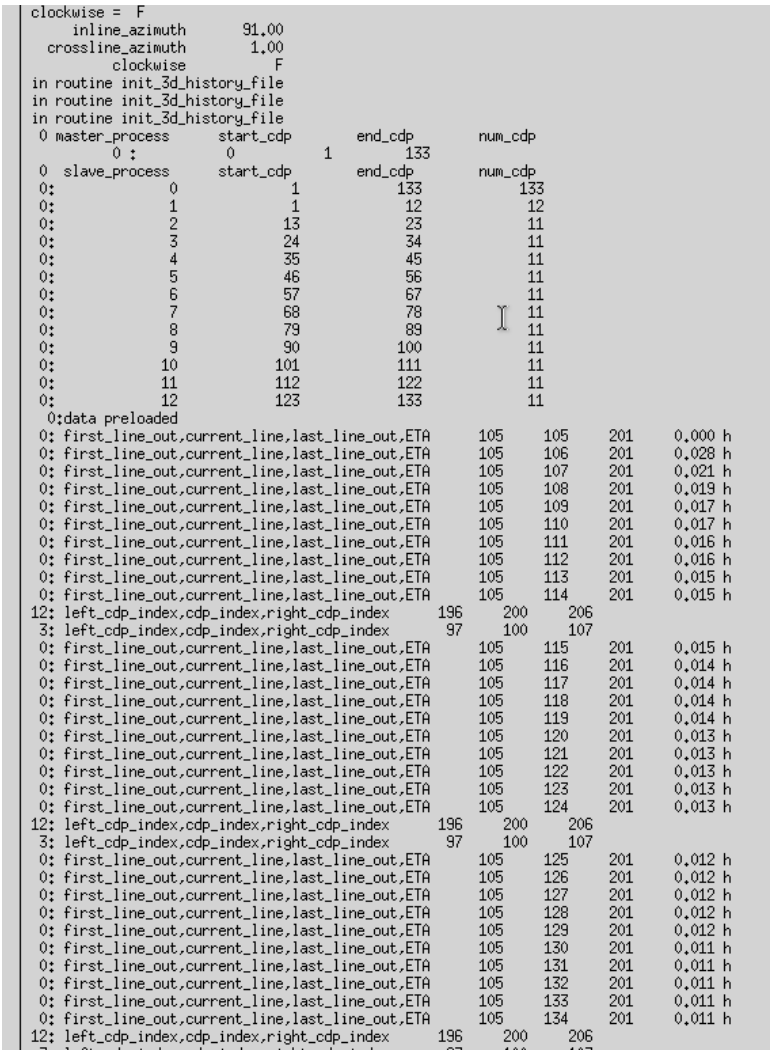
Programs **dip3d, sof3d**, and **sof_prestack** all use a modification of overlapping window parameter estimates introduced by Kuwahara et al. (1976) in medical imaging. The original idea is fairly simple. If an analysis window contains five traces, then there are total of five windows (a centered window and four adjacent, offset windows) that contain the analysis point. In Kuwahara et al.'s (1976) original work and Luo et al.'s (2002) edge-preserving smoothing algorithm, one calculates the mean and standard deviation of each window. That window which has the smallest standard deviation is hypothesized to be less noise-contaminated. The mean of this window is then used as the output for the analysis point. Marfurt (2006) modified this approach for volumetric dip calculations where he used 3D rather than 2D overlapping windows. Also, the semblance of the analytic signal is used rather than the standard deviation to determine which window is least contaminated by noise. The inline and crossline components of reflector dip of this 'best' window are then assigned to be the output at the analysis point.

For reasonably good quality seismic data, place a checkmark to (16) *search lateral windows.* If the data are very noisy, using Kuwahara windows can give rise to a 'patchy' appearance on the dip components. If this occurs, remove the checkmark in front *of search lateral windows.* For very good quality seismic data, place a checkmark in front of (15) *search vertical windows.* This option will avoid smearing angular unconformities, onlap, toplap, and other configurations important to seismic stratigraphy interpretation.

To minimize the 'patchy' appearance that can occur when using Kuwahara windows, (18) set a threshold value of the analytic semblance, *s*=0.85. If the semblance of the centered window is greater than this value of *s*, its value of dip and azimuth will be used. If the semblance is less than this value of *s*, the Kuwahara window concept will be implemented, with the dip components of the (centered, laterally offset, or vertically offset) window having the highest value of semblance assigned to the analysis point.

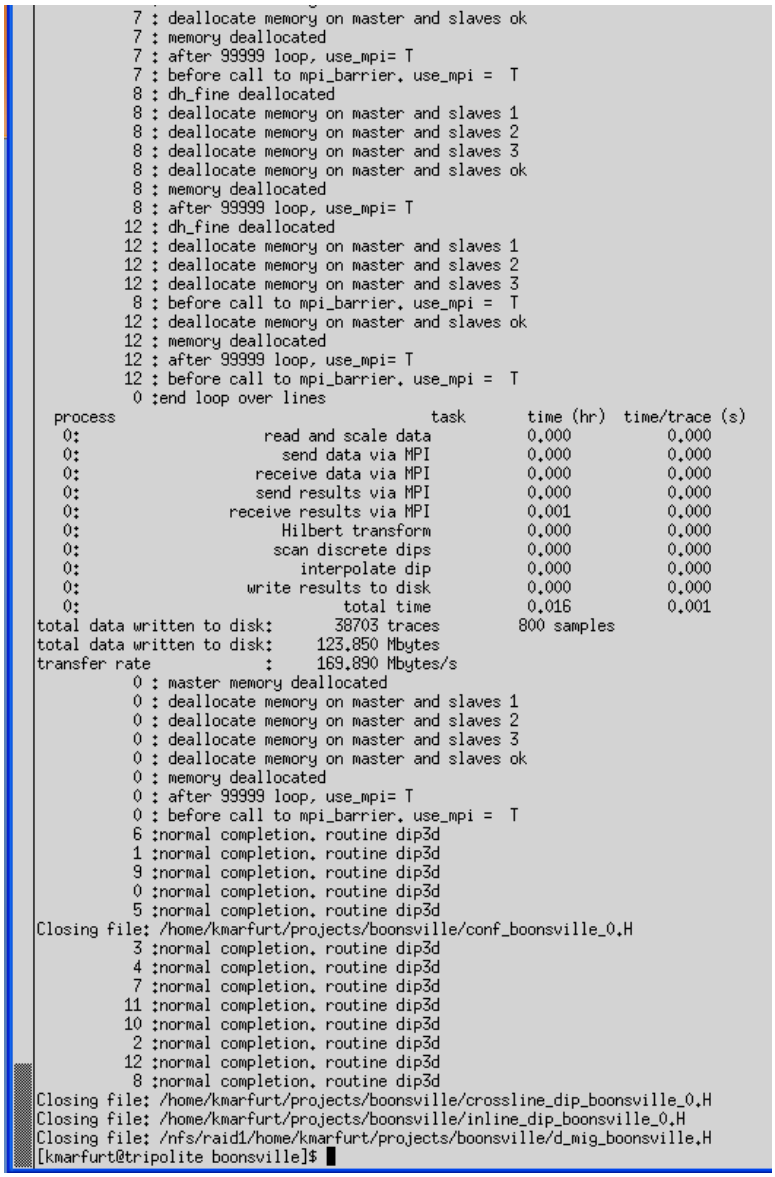
Execution

After selecting all your parameters, type '*Execute*'. Intermediate output comes to the xterm from which you launched **aaspi_util**. some of which appears below:



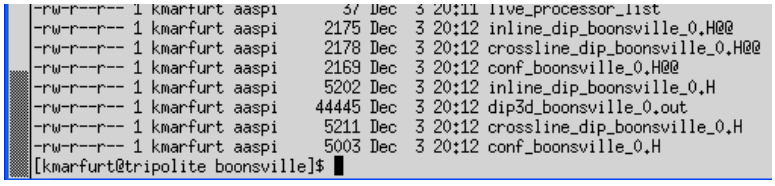
```
clockwise =  F
     inline_azimuth       91.00
  crossline_azimuth        1.00
         clockwise           F
in routine init_3d_history_file
in routine init_3d_history_file
in routine init_3d_history_file
 0 master_process      start_cdp       end_cdp       num_cdp
         0 :           0          1         133
 0  slave_process      start_cdp       end_cdp       num_cdp
 0:             0             1           133           133
 0:             1             1            12            12
 0:             2            13            23            11
 0:             3            24            34            11
 0:             4            35            45            11
 0:             5            46            56            11
 0:             6            57            67            11
 0:             7            68            78            11
 0:             8            79            89            11
 0:             9            90           100            11
 0:            10           101           111            11
 0:            11           112           122            11
 0:            12           123           133            11
  0:data preloaded
 0: first_line_out,current_line,last_line_out,ETA      105     105     201     0.000 h
 0: first_line_out,current_line,last_line_out,ETA      105     106     201     0.028 h
 0: first_line_out,current_line,last_line_out,ETA      105     107     201     0.021 h
 0: first_line_out,current_line,last_line_out,ETA      105     108     201     0.019 h
 0: first_line_out,current_line,last_line_out,ETA      105     109     201     0.017 h
 0: first_line_out,current_line,last_line_out,ETA      105     110     201     0.017 h
 0: first_line_out,current_line,last_line_out,ETA      105     111     201     0.016 h
 0: first_line_out,current_line,last_line_out,ETA      105     112     201     0.016 h
 0: first_line_out,current_line,last_line_out,ETA      105     113     201     0.015 h
 0: first_line_out,current_line,last_line_out,ETA      105     114     201     0.015 h
12: left_cdp_index,cdp_index,right_cdp_index      196     200     206
 3: left_cdp_index,cdp_index,right_cdp_index       97     100     107
 0: first_line_out,current_line,last_line_out,ETA      105     115     201     0.015 h
 0: first_line_out,current_line,last_line_out,ETA      105     116     201     0.014 h
 0: first_line_out,current_line,last_line_out,ETA      105     117     201     0.014 h
 0: first_line_out,current_line,last_line_out,ETA      105     118     201     0.014 h
 0: first_line_out,current_line,last_line_out,ETA      105     119     201     0.014 h
 0: first_line_out,current_line,last_line_out,ETA      105     120     201     0.013 h
 0: first_line_out,current_line,last_line_out,ETA      105     121     201     0.013 h
 0: first_line_out,current_line,last_line_out,ETA      105     122     201     0.013 h
 0: first_line_out,current_line,last_line_out,ETA      105     123     201     0.013 h
 0: first_line_out,current_line,last_line_out,ETA      105     124     201     0.013 h
12: left_cdp_index,cdp_index,right_cdp_index      196     200     206
 3: left_cdp_index,cdp_index,right_cdp_index       97     100     107
 0: first_line_out,current_line,last_line_out,ETA      105     125     201     0.012 h
 0: first_line_out,current_line,last_line_out,ETA      105     126     201     0.012 h
 0: first_line_out,current_line,last_line_out,ETA      105     127     201     0.012 h
 0: first_line_out,current_line,last_line_out,ETA      105     128     201     0.012 h
 0: first_line_out,current_line,last_line_out,ETA      105     129     201     0.012 h
 0: first_line_out,current_line,last_line_out,ETA      105     130     201     0.011 h
 0: first_line_out,current_line,last_line_out,ETA      105     131     201     0.011 h
 0: first_line_out,current_line,last_line_out,ETA      105     132     201     0.011 h
 0: first_line_out,current_line,last_line_out,ETA      105     133     201     0.011 h
 0: first_line_out,current_line,last_line_out,ETA      105     134     201     0.011 h
12: left_cdp_index,cdp_index,right_cdp_index      196     200     206
```

dip3d echoes out the current line it is working on, and an Estimated Time of Arrival (ETA) or expected time to completion. At the end, you will see the following:



which give some computational statistics for each process and shows that each of the 12 processes (number 0 is the master, while 1-12 are slaves) completed successfully. In this case, slave 12 was the last of the 12 to complete. The total elapsed time was 0.016 h.

So far, we have created the following files:



The file, *dip3d_boonsville_0.out* is a copy of the results that came to the screen. Program **dip3d** has generated three SEP-format output files: *conf_boonsville_0.H, crossline_dip_boonsville.H, dip_azimuth_boonsville.H, dip_magnitude_boonsville.H, inline_dip_boonsville.H,* (the SEP history files) and corresponding *.H@@ (header file format). The corresponding *.H@ (binary results), and *.H@@@ (binary trace headers) will reside in the directory you defined earlier in the *.datapath* file that resides in your home directory.

The file *live_processor_list* contains just that:

```
[kmarfurt@tripolite boonsville]$ cat live_processor_list
localhost slots=1
tripolite slots=12
[kmarfurt@tripolite boonsville]$
```

The master processor which does minimal computation but all of the input/output is called *localhost* with one 'slot' or CPU. *tripolite* does all of the computation and has 12 slots or CPUs. If we had used more processors, we would have used more slots. The software is clever enough to not use dead or misspelled processors (by first *pinging* them). Program **ping** is one of those Linux survival commands. Here at OU let's ping three different computers

```
[kmarfurt@tripolite boonsville]$ ping opal.ou.edu
PING opal.fwsmnnsvr.ou.edu (10.254.2.7) 56(84) bytes of data.
64 bytes from opal.sec-deptservers.nor.ou.edu (10.254.2.7): icmp_seq=1 ttl=64 time=0.118 ms
64 bytes from opal.sec-deptservers.nor.ou.edu (10.254.2.7): icmp_seq=2 ttl=64 time=0.128 ms
64 bytes from opal.sec-deptservers.nor.ou.edu (10.254.2.7): icmp_seq=3 ttl=64 time=0.208 ms
64 bytes from opal.sec-deptservers.nor.ou.edu (10.254.2.7): icmp_seq=4 ttl=64 time=0.161 ms

--- opal.fwsmnnsvr.ou.edu ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3000ms
rtt min/avg/max/mdev = 0.118/0.153/0.208/0.038 ms
[kmarfurt@tripolite boonsville]$ ping jade.ou.edu
PING jade.fwsmnnsvr.ou.edu (10.254.2.14) 56(84) bytes of data.
64 bytes from jade.sec-deptservers.nor.ou.edu (10.254.2.14): icmp_seq=1 ttl=64 time=1.23 ms
64 bytes from jade.sec-deptservers.nor.ou.edu (10.254.2.14): icmp_seq=2 ttl=64 time=0.194 ms
64 bytes from jade.sec-deptservers.nor.ou.edu (10.254.2.14): icmp_seq=3 ttl=64 time=0.163 ms
64 bytes from jade.sec-deptservers.nor.ou.edu (10.254.2.14): icmp_seq=4 ttl=64 time=0.148 ms

--- jade.fwsmnnsvr.ou.edu ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3000ms
rtt min/avg/max/mdev = 0.148/0.434/1.233/0.461 ms
[kmarfurt@tripolite boonsville]$ ping hematite.ou.edu
PING hematite.sec.ou.edu (10.254.2.21) 56(84) bytes of data.
64 bytes from hematite.sec-deptservers.nor.ou.edu (10.254.2.21): icmp_seq=1 ttl=64 time=1000 ms
64 bytes from hematite.sec-deptservers.nor.ou.edu (10.254.2.21): icmp_seq=2 ttl=64 time=1.22 ms
64 bytes from hematite.sec-deptservers.nor.ou.edu (10.254.2.21): icmp_seq=3 ttl=64 time=0.088 ms
64 bytes from hematite.sec-deptservers.nor.ou.edu (10.254.2.21): icmp_seq=4 ttl=64 time=0.088 ms

--- hematite.sec.ou.edu ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 2999ms
```

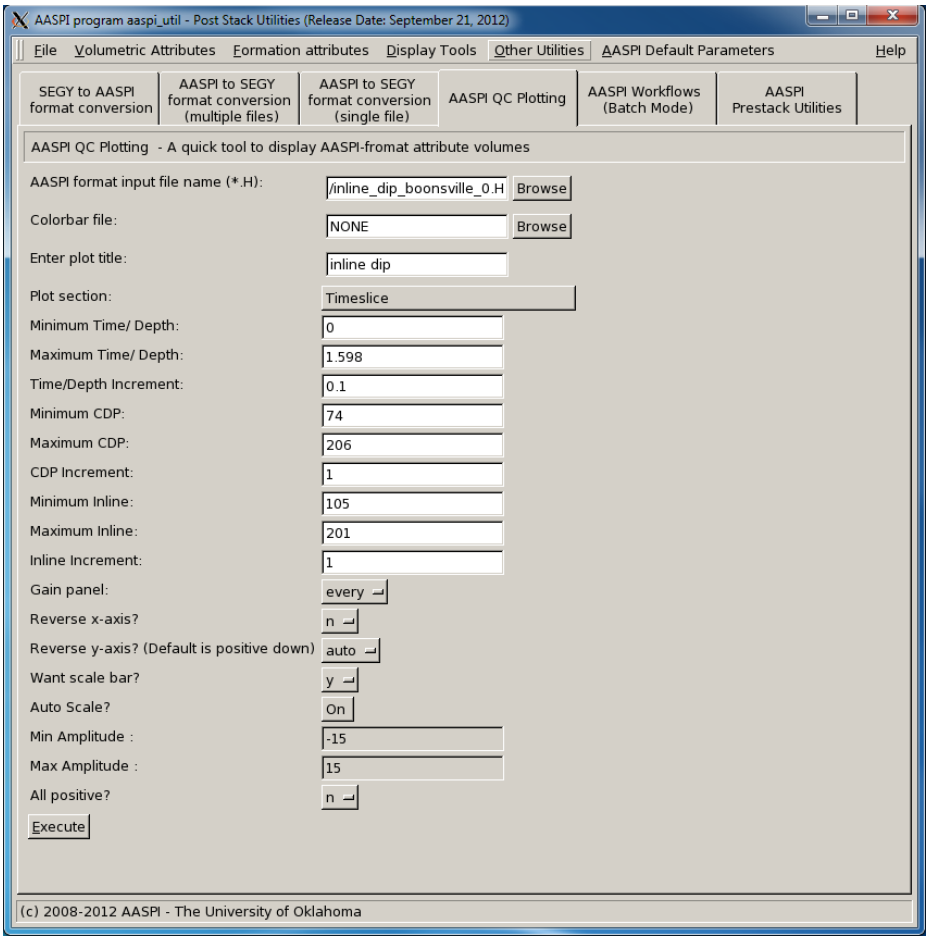
You will need to click *Ctrl-C* to kill the ping job. In the first case, *ping opal.ou.edu* ping, keeps repeating until we hit Ctrl-C. We have full (passwordless ssh) to this computer from the machine I am on *(tripolite.ou.edu)*.

It's easy to mistype a computer name. Here I mistakenly type *jadexx.ou.edu* rather than *jade.ou.edu*. **ping** cannot find this machine.

```
[kmarfurt@tripolite boonsville]$ ping jadexxx.ou.edu
ping: unknown host jadexxx.ou.edu
```

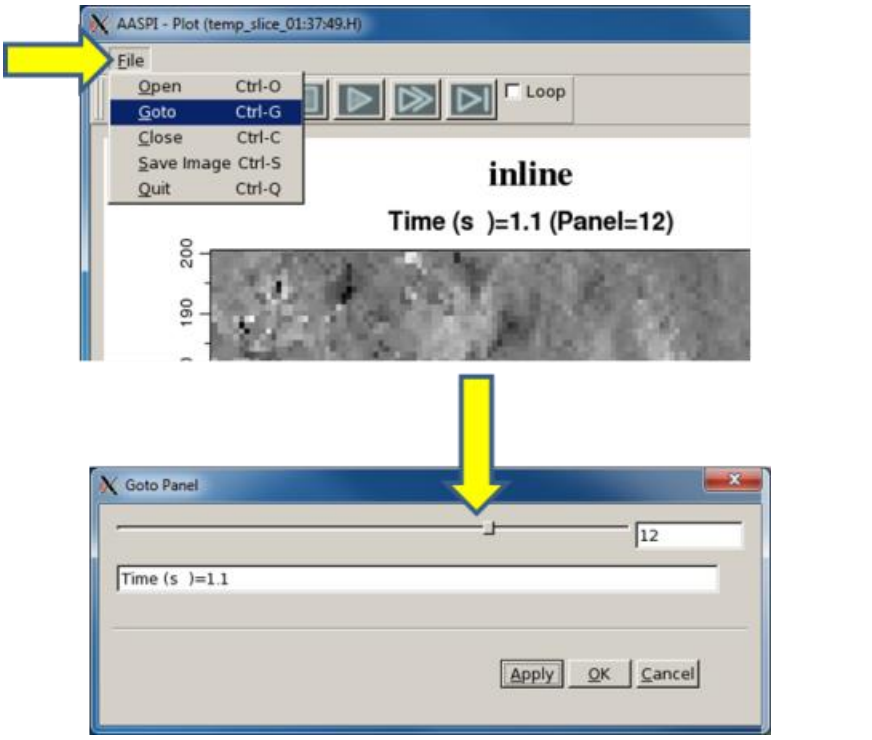
If you do not have access to a machine, or if it is offline for maintenance, you will obtain a similar error message. Again we type *Ctrl-C* to kill the job. If you are not on the OU

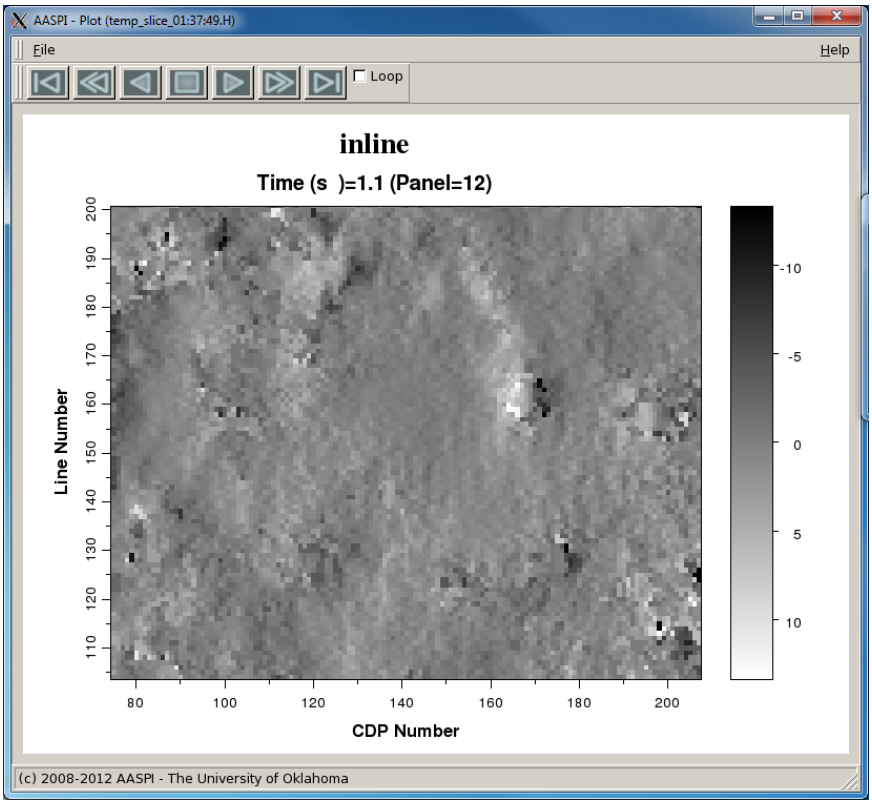
system (at your desk in an oil company), you will be able to *ping opal.ou.edu,* but not obtain access to it.
We can plot our dip components to QC the results by clicking the AASPI QC Plotting tab in the **aaspi_util** GUI:



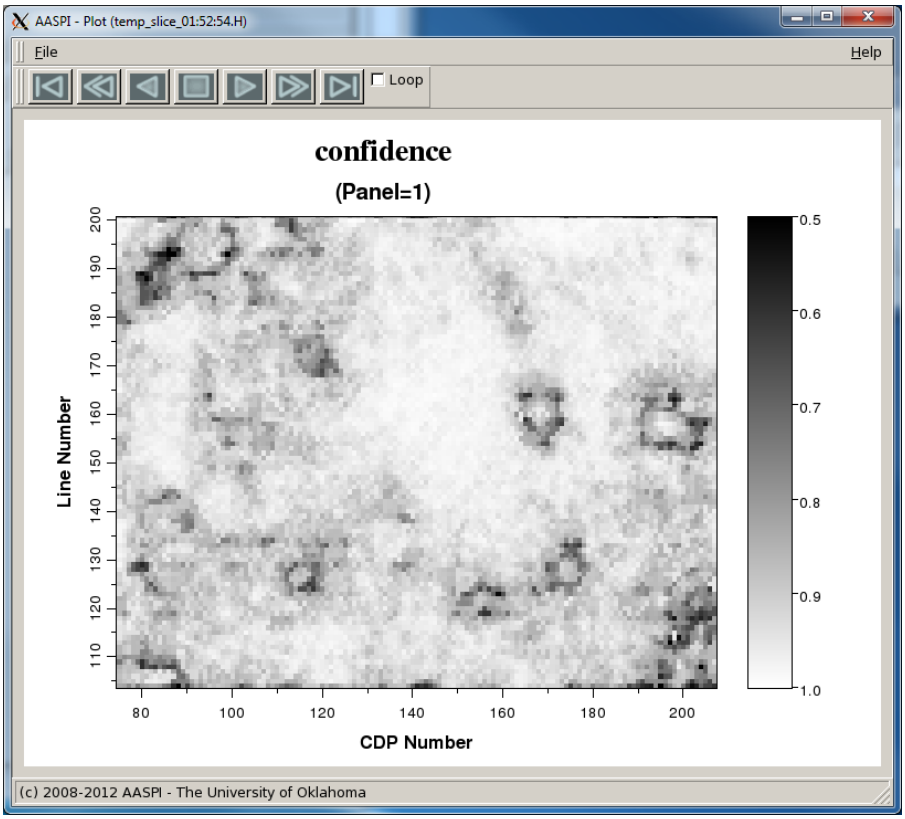
1.1

Browse and choose 'inline_dip_boonsville_0.H' as the file to be plotted, set the Time/Depth Increment to be 0.1 s, use the default Min Amplitude and Max Amplitude values, and click *Execute*. The SEP window pops up and we capture the slice at t=1.1 s and display it below.





Notice, there are some small glitches in my data. We also plot the confidence that we have in each of our dip estimates. Since program **dip3d** uses a Kuwahara multiple overlapping window dip search, the confidence is simply the coherence (in this case semblance of the analytic data) of the window used.

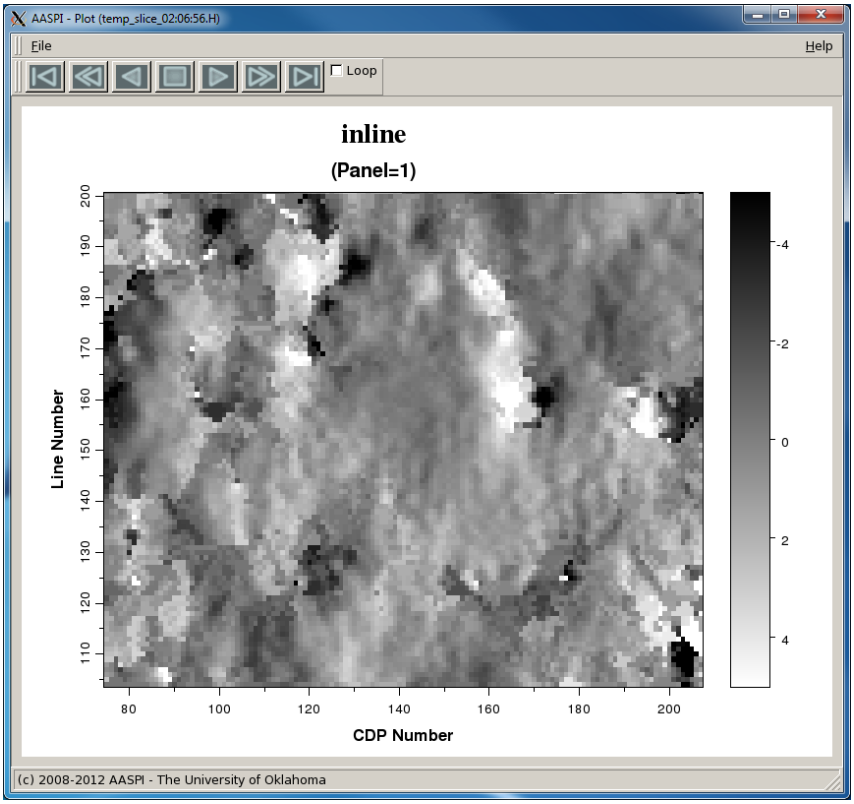


The low semblance areas correspond to the edges of the karst collapse features known to occur at this (Barnett Shale) level.

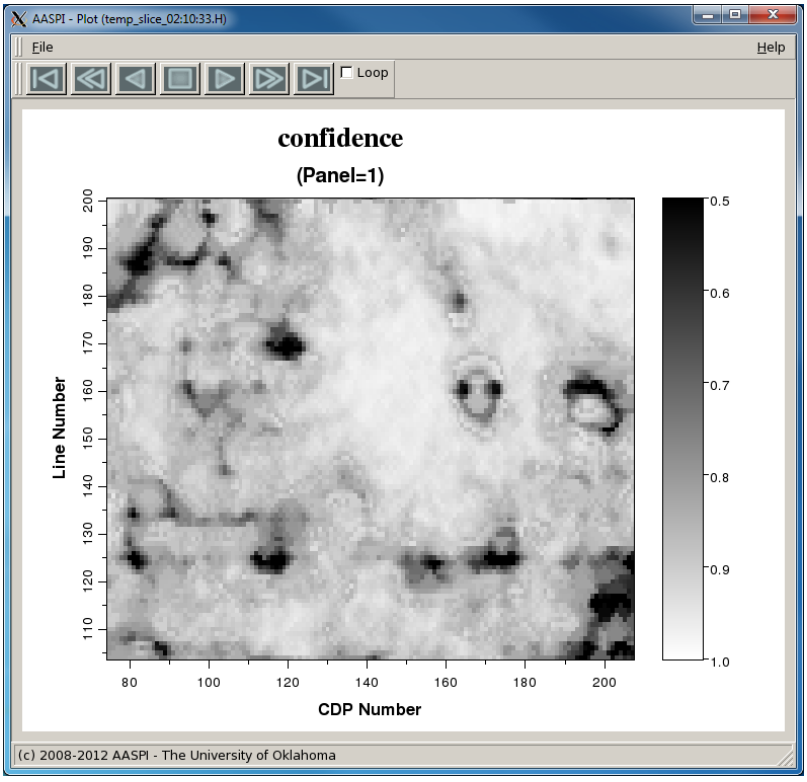
Next, let's evaluate the impact of using a larger analysis window. I set the inline and crossline radii to be 220 ft and set my *Suffix*=220x220ft to give the output files a different name. The run time of this 13-trace window is about three times longer than that for the smaller 5-trace window:

```
          0 :end loop over lines
 process                                  task       time (hr)  time/trace (s)
  0:                    read and scale data          0.000          0.000
  0:                      send data via MPI          0.000          0.000
  0:                   receive data via MPI          0.000          0.000
  0:                     send results via MPI        0.000          0.000
  0:                  receive results via MPI        0.002          0.000
  0:                       Hilbert transform         0.000          0.000
  0:                     scan discrete dips          0.000          0.000
  0:                        interpolate dip          0.000          0.000
  0:                    write results to disk        0.000          0.000
  0:                              total time         0.042          0.004
total data written to disk:       38703 traces      800 samples
total data written to disk:     123.850 Mbytes
transfer rate             :     212.435 Mbytes/s
```

The inline dip component at t=1.1 s looks like this:

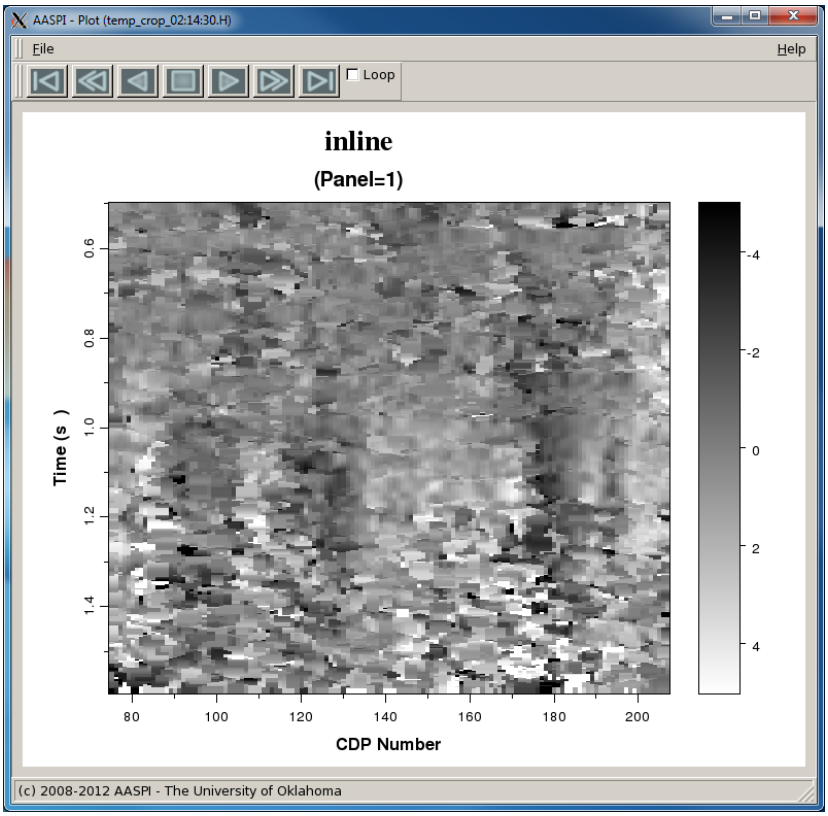


Note that it significantly smoother than that computed using the previous 5-trace window. The confidence is displayed below:

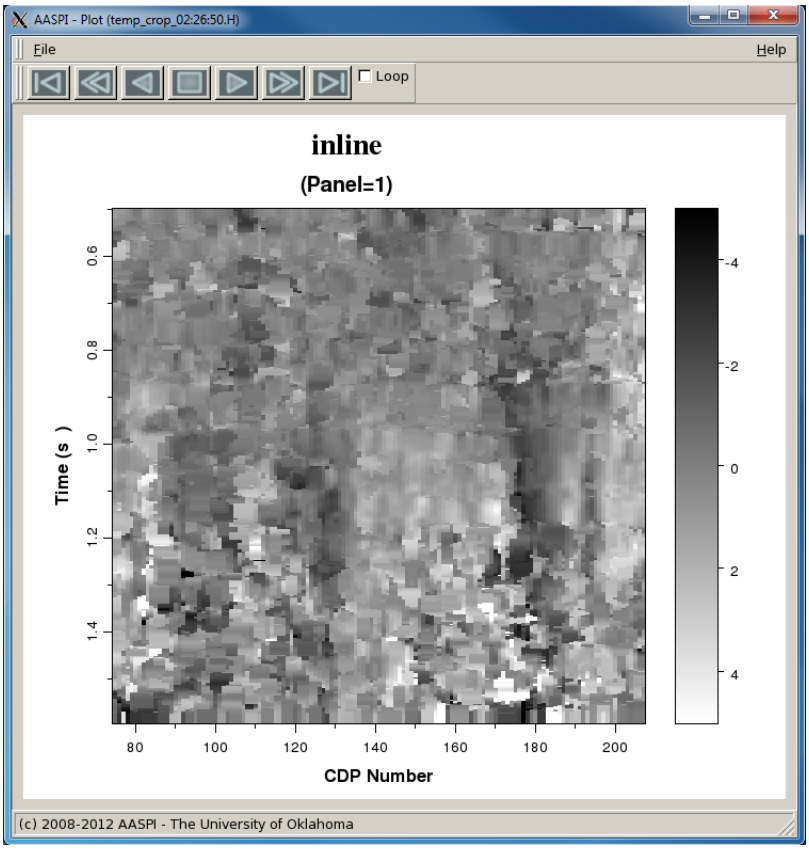


Careful inspection will show that the discontinuities in dip occur at areas where the confidence falls below *s*=0.85, as set by the parameters in the GUI.

Plotting a vertical slice through the inline dip (line 175) gives the following image:



We note the (dark) collapse feature under CDP 100, but also note many horizontal 'stripes' which correlate to the stronger wavelets. Increasing the *Dip Window Height* = 0.02 s, gives the following image



As anticipated, the computation time is almost identical due to the add-drop semblance computation scheme.

process	task	time (hr)	time/trace (s)
0:	read and scale data	0.000	0.000
0:	send data via MPI	0.000	0.000
0:	receive data via MPI	0.000	0.000
0:	send results via MPI	0.000	0.000
0:	receive results via MPI	0.002	0.000
0:	Hilbert transform	0.000	0.000
0:	scan discrete dips	0.000	0.000
0:	interpolate dip	0.000	0.000
0:	write results to disk	0.000	0.000
0:	total time	0.041	0.004
total data written to disk:	38703 traces	800 samples	
total data written to disk:	123.850 Mbytes		
transfer rate :	213.534 Mbytes/s		

We will try to further improve our dip estimates with some simple median filtering using program **image_filt3d.**

References

Kuwahara, M., K. Hachimura, S. Eiho, and M. Kinoshita, 1976, Digital processing of biomedical images: Plenum Press, 187–203.

Luo, Y., S. al-Dossary, and M. Marhoon, 2002, Edge-preserving smoothing and applications: The Leading Edge, **21**, 136–158.

Marfurt, K. J., 2006, Robust estimates of reflector dip and azimuth: Geophysics, **71**, 29–40.